fast.ai: Deep Learning Part 2 (v2) (2018)

7 Video lesson transcripts based on youtube.com autogenerated subtitles.

Discussion forum: http://forums.fast.ai/.

Version 0.1 (2018-06-28) (proofreading progress: 0%).

If you'd like to contribute with proofreading, please see:
https://github.com/stas00/fastai-transcript/.

Credits: Jeremy Howard (fast.ai Teacher/Founder), Eric Perbos-Brinck (Video Timelines), Ottokar Tilk (Punctuator), Stas Bekman (Transcript making).

Table of Contents

Lesson 8: Object Detection

Outline

Welcome to Cutting Edge Deep Learning for Coders, part 2 of fast.ai's deep learning course. This part covers lessons 8 to 14, and assumes you have already completed lessons 1 to 7 from part 1. If you're already a deep learning practitioner, feel free to try starting with this part—if you find that you're not familiar with the topics discussed, you can always return to part 1. In particular, we assume that you're familiar with:

The Pytorch and fastai libraries

CNNs and RNNs

Modern techniques for training and regularizing neural networks.

Lesson 8 starts with a quick recap of what we've learned so far, and introduces the new focus of this part of the course: cutting edge research. We talk about how to read papers, and what you'll need to build your own deep learning box to run your experiments. Even if you've never read an academic paper before, we'll show you how to do so in a way that you don't get overwhelmed by the notation and writing style. Another difference in this part is that we'll be digging deeply into the source code of the fastai and Pytorch libraries: in this lesson we'll show you how to quickly navigate and build an understanding of the code. And we'll see how to use python's debugger to deepen your understand of what's going on, as well as to fix bugs.

The main topic of this lesson is object detection, which means getting a model to draw a box around every key object in an image, and label each one correctly. You may be surprised to discover that we can use transfer learning from an Imagenet classifier that was never even trained to do detection! There are two main tasks: find and localize the objects, and classify them; we'll use a single model to do both these at the same time. Such multi-task learning generally works better than creating different models for each task—which many people find rather counter-intuitive. To create this custom network whilst leveraging a pre-trained model, we'll use fastai's flexible custom head architecture.

Video Timelines and Transcript

1. 00:00:00

- **Intro and review of Part 1**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Okay, well welcome to part two of deep learning for coders part. One was practical, deep learning for coders. That too, is not impractical, deep learning for coders, but it is a little

different as well discuss. This is probably a really dumb idea that last year I started like not starting part, two with part two lesson, one but part two lesson: 8, because it's kind of out of the same sequence. So I've done that again, but sometimes I'll probably forget and call things lesson, one so clap to lesson one in part, two less than eight or the same thing. If you ever make that mistake, so we're going to be talking about object. Detection today, which refers to not just blending out what a picture is a picture of, but also where abouts the thing is, but in general, the idea of each lesson in this part is not so much because I particularly want you to care about, say, object, rotation, But rather because I'm trying to pick topics which allow me to teach you some foundational skills that you haven't got yet right. So, for example, object detection is going to be all about creating much richer convolutional network structures which have kind of a lot more interesting stuff. Going on and a lot more stuff going on in the first day, I library that we have to customize to get there so like at the end of these seven weeks. I can't possibly cover the hundreds of interesting things that people are doing with deep learning.

Right now, but the good news is that all of those hundreds of things are you'll see once you later read the papers like minor tweaks on a reasonably small number of concepts, and so we covered a bunch of those concepts in part one and we're going to Go a lot deeper into those concepts and build on them to get just in deeper concepts in part two. So in terms of what we covered in part, one there's a few key takeaways we'll go through each of these takeaways into it. One is the idea - and you might have seen recently Young Buck Owens - been promoting the idea that we don't call this deep learning but differentiable programming and the idea is that you've noticed all the stuff we did in part. One was really about setting up a differentiable function and a loss function that describes how good the parameters are and then pressing go and it kind of makes it work. You know - and so this is kind of I think it's quite a good way of thinking about it - differentiable programming, this idea that if you can configure a loss function that does but you know describes you have scores how good something is it doing your task and You have a reasonably flexible neural network architecture, you're kind of done. Okay, so that's one key way of thinking about this. This example here comes from playground, tensorflow dot org, which is a cool website where you can play interactively with creating your own little differentiable functions manually.

The second thing, then, we learnt is about transfer learning and it's basically that transfer learning is like the most important single thing to be able to do to use deep wound effectively. Nearly all courses nearly all papers, nearly everything in deep learning, education and research focuses on starting with random words, which is ridiculous because you almost never would want to or need to do, that. You would only want to or need to do that if nobody had ever trained and model on a vaguely similar set of data with an even remotely connected kind of problem to solve. As what you're doing now, you know which almost so this is, where kind of the faster library and the stuff we talk about in this class, is vastly different too any other library, or course, is that it's all focused on transfer learning, and it turns out that You do a lot of things quite differently, so the basic idea of transfer learning is here's a network that does thing a remove the last layer or so replace it with a few random layers at the end, fine-tune those layers to do things be taking advantage of The features at the original network word and then optionally, find you in the whole thing into end and you've now got something which probably uses orders of magnitude less data. And if you start with random weights, it's probably a lot more accurate and probably trained a lot faster. You know we didn't talk a hell of a lot about architecture, design in part one and that's because kind of architecture design is getting less and less interesting.

There's a pretty small range of architectures that generally works pretty well. Quite a lot of the time we've been focusing on using CNN's for generally fixed size, somehow ordered data era tins for sequences at some state fiddling around a tiny bit with activation functions. I soft maps if you've got a single, categorical outcome or sigmoid if you've got multiple outcomes and so forth. Some of the architecture, design we'll be doing in this part, gets kind of more interesting, particularly this first session about object detection. But you know, on the whole, I think we probably spend less time talking about architecture design than most courses or papers, because it's it's not. It's you know, it's generally, not too happy okay. So the third thing is we looked at was out of word overfitting, and so did. The general idea that I tried to explain is is the way I like to build. A model is to of all create something. That's definitely terribly over parameterised will massively overfit for sure trainer and make sure it does over the it factors. At that point, you know: okay, I've got a model that is capable of reflecting the training set, and then it's as simple as doing these things to then reduce that overfitting and if you can't start, if you don't start with something's overfitting, then you you're kind of Lost right, so you start with something else: overfitting and then to make it over fit less.

You can add more data, you can add more data augmentation, you can do things like more batch norm, layers or dense Nets, or you know various things that can handle those a few nice data. You can add regularization like weight, decay and drop out, and then finally, this is often the thing people do first, but this should be. The thing you do last is reduce the complexity of your architecture, have less layers or less activations. We talked quite a bit about betting's both for NLP and the general idea of any kind of categorical data as being something you can now model with neural nets. And it's been interesting to see how, since part one came out at which

2. 00:08:00

- **Moving to Python 3**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Point there were almost no examples of kind of papers or blogs or anything about using kind of tabular data at categorical data in deep learning. Suddenly, it's kind of taken off and it's it's kind of everywhere, so this is becoming a more and more popular approach. It's it's still little enough known that when I say to people like, oh, you know, we use neural nets for time series and tabular data analysis is often like wait really, but it's definitely not such a far-out idea, yeah and there's more and more resources available, including Casual competition, recent capital, competition, winning approaches using this technique; okay, so the part one you know, which kind of particularly had those five messages really was all about, introducing you to best practices in deep learning, and so it's like trying to show you techniques which were mature Enough that they definitely work reasonably reliably for practical, real world problems and that I had researched and tuned enough over quite a long period of time that I could kind of say: okay, here's, the sequence of steps and architectures and whatever that, if you use this you'll, Almost certainly get pretty good results and then had kind of put that into the first day. I bring into a way that you could do that pretty quickly and easily, so that's kind of what practical, deep learning pakodas was designed to do so.

This part two is cutting edge, deep learning for coders, and what that means is, I often don't know the exact best parameters, architecture, details and so forth to solve Euler problem. We

don't necessarily know if it's going to solve a problem well enough to be practically useful. It almost certainly won't be integrated well enough in too fastai or any library that you can just press a few buttons. It'll start working, it's it's all about stuff which I'm not going to teach it unless I'm very confident.

3. 00:10:30

- **Moving to Tensorflow and TF Dev Summit videos**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

That you know there is now or will be soon, very practically useful technique. So, like I don't kind of take stuff which just appeared - and I don't know enough about it to kind of know like what's a trajectory gon na be so if I'm teaching it in this course and saying, like you know, this is, you know either works well In the research literature now and it's gon na be well worth learning about or we're pretty close to being there. It's got to take a lot of freaking, often and experimenting to get it to work on your particular problem, because we don't know you know the details. Well enough to know how to kind of make it work for every dataset or every example, so it's kind of exciting to be working at this point. It means that, rather than fastai and pytorch being obscure black boxes, which you just know these recipes for you're, going to learn the details of them. Well enough that you can customize them exactly the way you want that you can debug them, that you can read the source code or to see what's happening, and so, if you're, not pretty confident of you know object-oriented Python and stuff like that, then that's something you Don't want to focus on studying during this course, because we, we assume that I'm not going to spending time on that, but I will be trying to introduce you to some some tools that I think are particularly helpful.

Like the Python debugger like how to use your editor to kind of jump through the code stuff like that, and in fact in general, there'll, be a lot more detailed. Specific code, walkthroughs coding, technique, discussions and stuff like that, as well as more detailed walkthroughs of papers and stuff, and so anytime. We cover one of these things if you notice something where you're like you know, this is assuming some knowledge that I don't have. That's fine. You know it just means like that's something you could ask from the forum and say: hey you know, Jeremy kind of was talking about whatever static methods in Python. I don't really know what a static method is or like he was using it here, because only some resources like you know these are kind of things at all: they're, not rocket science. It's just just because you don't happen to have come across it yet doesn't mean it ha. It's just something you learn. I will mention that, as I cover these research level topics and develop these courses, I often refer to code that academics have put up. You know to go along with it papers or kind of example, code that somebody else is written on github. I nearly always find that there's some massive critical flaw so be careful of like taking code from you, know, online resources and just assuming that, if it doesn't work for you that you've made a mistake or something you know this kind of like research level code.

It's it's just good enough that they were able to run their particular experiments. You know every second Tuesday, so you should you know you should be ready to kind of do some debugging. So on that, that's, since I just wanted to remind you about something from our old course wiki that we sometimes talk about, which is like people often ask what should I do after the less and like had away? How do I know if I've got it right and we basically have this

thing called how to use the provided notebooks and the idea is this part, don't open up the notebook, and I know I said this in part - one as well, but I'll say it again And go shift and a shift and a shift enter until a bag appears and then go to the forums and say it notebooks broken right. The idea of the notebook is to kind of be like a little crutch to help you get through step. The idea is that you start with like an empty notebook and think like okay, I now want to complete this process right and, and that might be initially require you op tabbing, to to the notebook and reading it figuring it out what it says, but whatever you Do don't copy and paste it your notebook type it out yourself right back, so try to make sure you can repeat the process and as you're typing it out and you'd be thinking like well. What am i typing? Why don't i? Okay? So if you can get to the point where you can, you know solve an object, detection problem yourself in a new empty notebook, even if it's using the exact same data set, we used in the course that's a great sign that you're getting it right and that'll. Take a while, but the idea is that by practicing you know the second time you try to do it.

The third time you try to do it. Your check, the notebook glass! Yes right and if there's anything in the notebook where you think, if you think I don't know what it's doing, I hope to teach you enough techniques in this course in this past that you'll know how to experiment to find out what it's doing right. So you shouldn't have to ask that, but you may well want to ask like why is it doing that you know that's the conceptual bit and that's something which you may need to go to the forums and say, like you know, before this tip jeremy had done This after this tip jeremy, had done that this is bit in the middle way. Who does this other thing? I don't quite know why you know so, then you can try and say like here my policies as to why like try and work through it as much as possible and that way you'll both be helping yourself and other people will help. You fill in the gaps right if you wish, and you have the financial resources now is a good time to build a deep learning box for yourself, when I say a good time, I don't mean a good time in the history of the pricing of GPUs. Gpus are currently by far the most expensive they've ever been. As I say this because of the cryptocurrency mining boom, I mean it's a good time in your study cycle. I mean the fact is, if you're paying somewhere between 60 cents and 90 cents an hour for doing your deep learning on a cloud provider, particularly if you're still on a k80 like an Amazon, p2 or google collab.

Actually I haven't come across it now. Let's you train on a kad for free, but those are very slow jeez. You know you can buy one. It's gon na be like three times faster for maybe six hundred seven hundred dollars. You need a box to put it in, of course, but you know the example in the bottom right here from the forum was something that somebody put together in last year's cost. So, like a year ago, they were able to put together a decent box referred over five hundred dollars. Generally speaking, your problem, I created a veneer forum thread where you can talk about. You know, options and parts and ask questions and so forth, afford it right now that gtx 980ti is almost certainly what you want in terms of the best price performance MIPS if you can't afford it at ten. Seventy is fine. If you can't afford that, you should probably be looking for a secondhand 980 or a second Amazonia, something like that. If you can afford to spend more money, it's worth getting a second GPU, so you can do what I do, which is to have one GPU training and another GPU which I'm running an interactive Cupid in the clock session. Ram is very useful. Try and get 32 gig if you can Ram, is not terribly expensive. A lot of people find that their vendor or person to buy one of these business classes they on CPUs. That's a total waste of time. You can get one of them into our eye.

Five or I seven consumers to CPUs far far cheaper, but actually a lot of them are faster, often you're here, CPU speed doesn't matter, that's if you're doing computer vision. That's definitely

not true. It's very common now with these, like 1080 TRS and so forth, to find that the speed of the data augmentation is actually this mo bit. That's happening on the CPU, so it's worth getting a decency to you, your again, your GPU. If it's running quickly, but the hard drives got fast enough to give it data, then that's a waste as well. So if you can afford an nvme drive, they're super super fast. You don't have to get a big one. You can just get a little one. You just copy your current set of data onto and have some big raid array that sits there for the rest of your data when you're not using it. There's a slightly arcane thing about PCI lanes, which is basically like they're kind of the size of the highway. That connects your GPU to your computer and a lot of people claim that you need to have a 16 lanes to feed your GPU. It actually turns out, based on some analysis, that I've seen recently that that's not true, you need you need eight lanes GPU. So again, so like hopefully help you save some money on your motherboard if you've never heard of PCI lanes before trust me by the end of putting together this bus, you'll be sick of hearing about, you can buy all the parts and put it together yourself.

It's not that hard can be a useful learning experience. It can also be kind of frustrating and annoying, so you can always go to like central computers and they'll put it together, for you there's lots of online and vendors. That will do the same thing now generally like make sure it turns on and runs properly generally, not much of a markup. So it's not a bad idea. We're going to be doing a lot of reading papers, basically, each week we'll be implementing a paper or a few papers, and if you haven't looked at papers before they look. Something like on the left. That thing on the left is an extract from the paper that implements atom. You may also have seen atom as a single excel formula on the spreadsheet that are in there the same thing. Okay, the difference is in academic papers. People love to use Greek letters, they also hate to refactor. So you'll often see like like a page long formula where

4. 00:22:15

- **Moving to PyTorch**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

When you actually look at it carefully, you'll realize like the same and a sub equation of his eight times. You know they didn't know if they didn't think to say about it like that T equal like this, that equation announced one. I don't know why. This is a thing, but I guess all this is to say like once: you've read and understood a paper. You then go back to it and you look at it. You just like wow. How did they make such a simple feeling so complicated, like Adam Wright, is like momentum and momentum on this momentum on the gradient and momentum on the squared already? That's it right and and speak long things and the other reason it's a big long thing is because they have things like this, where they have like theorems and calories and stuff, where they're kind of saying, like here's or our theoretical reasoning behind why this ought to Work or whatever, and for whatever reason, you know a lot of conferences and journals, don't like to accept papers that don't have a lot of this theoretical justification. Geoffrey Hinton's talked about this a bit. How particularly you know a decade or two ago, when no conferences would really accept any neural network papers. Then there was this like one, abstract, theoretical result that came out where suddenly they could show this.

You know I don't like practically unimportant, but theoretically interesting thing and then suddenly they could then start submitting things to journals because they have this like theoretical justification. So it's kind of yeah academic papers are a bit weird, but in the end it's

the way that the research community communicates their their findings. And so we need to learn to read them. But something that could be a great thing to do is to take a paper put in the effort to understand it and then write a blog where you explain it in you know: code in normal, English and lots of people who do that. You know end up getting quite a following end up getting some pretty great job offers and so forth, because you know it's such a useful skill to show like okay, I can I can understand these papers, I can implement code, I can explain them in English. One thing I will mention is it's very hard to read or understand something which you can't localize, which means, if you don't know the names of the Greek letters like it sounds weird, but it's actually very difficult to understand. Remember take in a formula that appears again and again that's got like squiggle right. You need to know that that squiggle is called Delta or that screw or the Sigma or whatever. So, like just spending some time learning the names of the Greek letters is like sounds like a strange thing to do.

But suddenly you don't look at these things anymore and go like squiggle iovers people beat us other weird squiggle looks like a. Why thing right, they've! All got notes, okay, so now that we're kind of at the cutting edge stage, a lot of the stuff we'll be learning, this class is stuff that almost nobody else knows about. So that's a great opportunity for you to be kind of like it's the first person to create an understandable and generalizable code, library that implements it or the first person to write a blog post explains it in clear English for the first person to try applying it To this slightly different area, but it's obviously going to work just as well or so so when we say cutting-edge research, that doesn't mean you have to come up with like the next batch norm or the next atom or the next five liter convoluted convolution. It can mean, like okay, take this thing that was used for translation and apply it to this very similar other parallel and LP task, or take this thing that was tested on skin lesions and trusted on this data set of this other clarifications. That kind of stuff is super great learning, experience and incredibly useful, because the vast majority of the world that knows nothing about this whole field. It just looks like magic. You know you're, like a I've for the first time shown greater than 90 percent accuracy at you know finding this kind of lesion in this kind of data . So you know when I say here, experiment in your area of expertise. You know one of the things we particularly look for in this class is to kind of bring in people who are pretty good at something else.

You know pretty good at meteorology or pretty good at denovo drug design or pretty good at goat, dairy farming or

5. 00:27:30

- **From Part 1 “best practices” to Part 2 “new directions”**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Whatever you know examples people we had in the class, so probably the thing you can do the best would be to take that thing, you're already pretty good at and that on these new skills right because otherwise, if you try to go into some different domain, you're Gon na have to figure out how do I get data for that domain? Problems are solved in that domain. Where else often it'll seem pretty trivial to you to take this technique apply to this data set that you've already got sitting on your hard drive, but that's often going to be that super interesting thing. You know for the rest of the world to see like. Oh, that's interesting, you know when you apply it to meteorology data and you use this eridan or whatever, and allows you to forecast

over larger areas or a longer time periods. So communicating what you're doing is super helpful. We've we've talked about that before, but I know something that a lot of people in the forum's ask people who have already written is that, when somebody's written, a blog, often people on the forum will be like. How did you get up the guts to do that or what did you? What up the process you got to before you decided to start publishing something or whatever, and the answer is always the same. It's always just you know. I was sure I wasn't good enough to do it. I felt terrified and intimidated of doing it, but I wrote it and posted it anyway, but you just like never a time.

I think any of us actually feel like we're not total frauds and imposters, but we know more about what we're doing then us of six months ago right and there's somebody else in the world who knows as much as you did six months ago. So if you act something now that would have helped you at six months ago, you're helping some people and honestly, if you wait another six months, then the you of 12 months ago, probably won't even understand that any more offices to advanced it know so Mike. It's great to communicate wherever you're up to in a way that you think could be helpful to the person you were before. You knew that thing, um and, of course, something that the forum's have been useful for is getting feedback about drafts. You know, and if you post a draft of something that you're thinking of releasing and the folks here, can point out things that they find clear or they think need some Corrections whatever. So the kind of overarching theme with part two I've described as generative models, but unfortunately, then Rachel asked me this afternoon exactly what I meant by generative models, and I realize I don't really know so. What I really mean is, in part one that the output of our neural networks was generally like. Like a number, you know our category, where else the outputs, a lot of the stuff in part, two are going to be like a whole lot of things.

You know like the top left and bottom right location of every object in an image along with what the object is or a complete picture, with the class of every single pixel in that picture or an enhanced super resolution. Version of the input image or the entire original input, paragraph translated into fish - or you know it's kind of like often it just requires some different ways of thinking about things and some kind of different, architectures and and so forth, and so that's kind of like. I guess the main theme of the kind of techniques we'll be looking at the vast majority, possibly

6. 00:31:40

- **Time to build your own box**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

All of the data we'll be looking at will be either text or image data the it would be fairly trivial to do most of these things with audio as well. It's just not something. I've spent much time on myself, yet somebody asked on the forum about like welcoming doing more stuff with time-series and tabular data, and my answer was like I've already taught you everything I know about that, and I'm not sure there's much else to say, particularly if you Check out the machine learning course, which goes into a lot of that in a lot more detail, so I don't feel that there's more stuff to tell you, I think, that's a super important area, but I think we're done we're done with that, we'll be looking at Some larger data sets both in terms of the number of objects in the data search and the size of each of those objects. For those of you that are

working with limited computational resources, please don't let that put you off feel free to replace it with something small at the simpler. In fact when I was designing this course, so did quite a lot of it in Australia. When I went to visit my mom and my mom decided to book a nice holiday house for us with fast Wi-Fi and we turned up to the holiday house with fast Wi-Fi, and indeed it did have Wi-Fi that was fast, but the Wi-Fi was not the internet.

So I caught up the agent - and I said, like I, found thee a DSL router and it's got an ADSL thing plugged in and I followed the cable down and the other end of the cable has nothing to plug into so she called the she called the People, you know renting the house ona and called me back the next day and she said actually the tambaran is quite quickly. I actually point leo: has known internet wait what, and so the good old Australian government had decided to replace ADSL in point Leo with a new National Broadband Network and therefore they had disconnected ADSL that had not yet connected and after brought that down. So we had fast Wi-Fi which we could use to Skype chat from one side of the house to the other, but I had no internet. Luckily, I did have a news surface book fifteen-inch, which has a gtx 1070 in it, and so I wrote a live German of this course entirely on my laptop, which means I had to practice with relatively small resources. I mean not tiny, like 16 gig ram and six gig GP here, so I can definitely you know III, definitely, and it was all in Windows. So I can tell you that most of this you know much one where this course works well on Windows on a laptop, so you can always use smaller batch sizes. You could use a cut-down version of the data set whatever, but if you have the resources, you'll get better results. If you can use the bigger data sets when they're available.

Okay, now it's a good time, I think, to take a somewhat early break, so we can fix the floor so the forum's chill down okay, so it was it okay, let's come back at 7:25. So let's start talking about object, detection and so here is an example. Object, detection and so hopefully, you'll see two main differences from what we're used to when it comes to classification. The first is that we have multiple things that were classifying, which is not unheard of. We did that in the planets satellite data, for example. But what is kind of unheard of is that, as well as saying what we see, we've also got: what's called bounding boxes around. What we see a bounding box has a very specific definition, which is it's a box all right. It's a rectangle and the rectangle has the object.

7. 00:36:20

- **Time to start reading papers**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Entirely fitting within it, but it's no bigger than it has to be okay. You'll see this bounding box is perhaps for the horse at least slightly imperfect, in that this looks like there's a bit of tail here, so it probably should be a bit wider and maybe there's leaving a little bit of hoof here. Maybe there should be a bit longer so, like the bounding box, this won't be perfect, but they're generally pretty good in most data, so our job will be to take data that has been labeled in this way and on data that was unlabeled to generate their classes Of the objects and each one of those their bounding losses, one thing I'll note to start with is that labeling, this kind of data is generally more expensive, it's generally quicker to say horse person, person horse car dog jumbo jet than it is to say you know, If there's a whole like horse race, going on to label the exact location of every rider and a very horse, and then of course, it also depends like what

classes do you want to label? You know if you want to label everything fence, post or whatever? So, generally always just like in like imagenet. It's not like tell me any object. You see in this picture. It's an image notice like here are the thousand classes that we asked you to look for, tell us which one of those thousand classes you find just tell me one thing for these object: detection, datasets! It's here's! A list of object classes that we want you to tell us about.

You know, find every single one of them at any time in the picture, along with where they are so in this case, why isn't there tree or jump labeled? That's because for this particular data set, they weren't one of the classes that the annotators were asked to find and therefore not part of this particular problem. Okay, so that's kind of the specification of the object detection problem, so let me describe stage 1 and stage. 1. Is actually going to be surprisingly straightforward and we're going to start at the top and work down we're going to start out by classifying the largest object in each image, so we're going to try and say the person actually, this one is wrong. Dog is not the largest object. Sofa is the largest object, all right, so here's an example of a misclassified one bird, correct person, correct okay, that'll be the first thing we try to do. That's not going to require anything new, so it'll just be a bit of a warm-up for us. The second thing will be to tell us the location of the largest object at each image again here. This is actually incorrect. It should have labeled the sofa, but you can see where it's coming from and then finally, we will try and do

8. 00:39:30

- **Time to start writing about your work in this course**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Both at the same time, which is to label what it is and where it is for the largest thing - and this is going to be obviously straightforward - actually so to be kind of good warm-up to get us going again. But what I'm going to do is I'm going to use it as an opportunity to show you some useful coding techniques and a couple of little faster. I Andy details before we then get on to multi-label classification and then object specification. So, let's start here, the logbook that we're using is Pascal notebook and it's all of the notebooks are in the DL too. One thing you'll see in some of my notebooks is torch, Dakota dot set device. You may have even seen it in the last part. Just in case you're wondering why that's there, I have four GPUs on the university server that I use, and so I can put a number from North to three in here to pick one. This is how I prefer to use multiple GPUs, rather than run a model on multiple GPUs, which doesn't always beat it up that much and it's kind of awkward. I generally like to have different GPUs running different things, so I, in this case I was running something in this on device one and doing something else, another booking device to. Obviously, if you see this in a notebook left behind, that was a mistake. If you don't have more than one GPU you're going to get an error done, so you just change it to zero or delete that line entire. So there's a number of standard object.

Detection data sets just like imagenet kind of a standard object. Classification data set and kind of the the old classic kind of image net equivalent, if you like, is Pascal vo, see

9. 00:41:30

- **What we'll study in Part 2**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

10. 00:40:40

- **Artistic style (or neural style) transfer**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Visual object class and it's something like that: yeah, the actual main website for it is like I don't know it's running on somebody's coffee, warmer or something it goes down all the time. Every time he makes coffee. I don't know so, some folks with m[?]rida. It's very kind of thin, so you might find it easier to grab from the mirror you'll see when you downloaded that there's a 2007 dataset of 2012 data set that there basically were like academic competitions in those different years. Just like the Internet data set, we tend to use, is like actually the image net 2012 competition producer now we'll be using the 2007 version in this particular notebook feel free to use of 2012 and stared. It's a bit bigger. You might get better results. A lot of people, in fact, most people now in research papers, actually combine the two you do have to be careful because there's some leakage between the validation sets between the two. So if you do decide to do that, make sure you do some reading about the data set to make sure you know how to combine them correctly. The first thing you'll notice in terms of coding here is this. We haven't used this before I'm going to be using this all the time now. This is part of the Python 3 standard library called path Lib, and it's super handy. It's basically gives you an object-oriented access to directory or a file, so you can see if I go path, dot, something it there's lots of things I can do.

One of them is iterative directory. However, path iterate directory returns that hopefully you've come across generators by now, because we do quite a lot of stuff that use them behind the scenes without talking about them too much. But basically, a generator is something in in Python 3, which you can iterate over right. So basically, you go for oh in that grid, Oh for instance, right, ok or of course you could do the same thing as a list. Comprehension right or you can just stick the word list around it to turn a generator into the list. Ok, so anytime, you see me put mist around something, that's normally because it pretend a generator. It's not particularly interests, ding. The reason that things generally return generators is that, like what, if the directory had 10 million items in you, don't necessarily 1 to 10 million long list, so we were the for loop, just grep 1. Do the thing thrown over wait a second throw it away and lets you do things lazily you'll see that the things that's returning aren't actually strings, but they're some kind of object. If you're using Windows it'll be a Windows path of Linux would be a POSIX path. Most of the time you can use them as if they're strings so most like. If you pass it, you know any of the OS path dot. Whatever functions in Python, it'll just work at some external libraries.

It won't work. So that's fine! If you grab one of these - let's say, let's say, o equals: let's just grab one of these, so in general you can change data types in Python, just by naming the data type that you want and treating it like a function and that will cast it. I had so many time you try to use one of these paths, Lib objects and you pass it to something which says like I was expecting a stream. This is not a string. That's okay, so you'll see there's quite a lot of convenient things. You can do one kind of fun thing is the slash operator is not divided by, but its path, slash, so like they've overwritten, the slash operator in Python, so that it works,

so you can say path, slash whatever and that gets you you'll see like see how That's not inside a stream right, so this is actually applying, not the division operator, but the overridden slash operator, which means get the child thing in that path. That makes sense and you'll see. If you run that it doesn't return a string, it returns, a pathway, object. Okay and so part one of the things the path the object can do is it has an open method right. So this it's it's kind of it's actually pretty cool once you start getting. The hang of it and you'll also find that, like the open method takes all the kind of arguments you're familiar with, it's a right for binary your encoding order.

So in this case, I want to load up these these JSON files, which contain not the images but the bounding boxes and the classes of the objects and so in Python. The easiest way to do that is with the JSON library or there's some faster API, equivalent versions. But this is pretty small, so you won't need them and you go json dot load and you pass it and open file object, and so the easy way to do that since we're using path live, is just go path open. So these JSON files that we're going to look inside in a moment if you haven't used it before JSON, is JavaScript object, notation it's kind of the most standard way to pass around hierarchical structure. Don't yet know, obviously not just with JavaScript you'll see. I've got some JSON files in here. They actually did not come from the mirror. I mentioned the the original pascal annotations were an xml format, but cool kids club uses email anymore. We have to use JSON, so somebody's converted them all to JSON and so you'll find. The second link here has all the JSON files. So if you just pop them in the same location that I've put them here, everything will so these annotation files jaison's basically contain a dictionary. Once you open up the JSON, it becomes a Python dictionary and they've got a few different things in the first is we can look at images, it's got a list of all of the images how big they are and the unique ID for each one.

One thing you'll notice here is taken the word images and put it inside a constant court images that may seem kind of weird, but if you're using you can a notebook or any kind of IDE or whatever this down means, I can tap complete all of my Strings and I won't accidentally type it slightly wrong, so it's just a handy trick: okay, so here's the contents first few things and the images more interestingly here are some of the annotations right. So you'll see basically an annotation contains a bounding box and the bounding box tells you the column and row if the top left and it's height and width, and that it tells you that that particular bounding box is for this particular image. So you'd have to join that up to over here to find it sexually, Oh to top jpg, okay and it's of category ID 7 um it also some of them at least has a polygon segmentation, not just a bounding box. We're not going to be using that some of them have an ignore flag, so we'll ignore the ignore flags, and some of them have something telling you it's a crowd of that object, not just one of them right. So that's that's what these annotations look like. So then you saw here there's a category ID so then the categories for examples they're, basically each ID. He has a name here we go okay. So what I did then was turned the his categories list into a dictionary from ID to name.

I created a dictionary from ID to name of the image file names and I created a list all of the image IDs just to make life easier. So you know generally like when you're working with a new data set, at least when I work with a new dataset. I try to make it look the way I would want it to if I designed that data set so kind of do a quick manipulation, and so, like the the steps you see here and you'll, see an h-class. Basically, like the sequence of steps, I talk as I started. Working with, is this bead onus it except like without the thousands of screw-ups that I did. I find like the the one thing people most comment on when they see me working in real time. Having seen my classes is like wow, you actually don't know what you're doing it's like

99. Some of the things I do don't work more percentage of the things that do work end up here so like this is like I mentioned that, because machine learning and particularly deep learning, is kind of incredibly frustrating, because you know in theory you just to find the Correct must function and flexible enough architecture and you press train and you don't all right, but if that was actually all a talk, then like nothing would take any time. The problem is that all the

11. 00:52:10

- **Neural style notebook**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Steps along the way until what works it doesn't work. You know like it. It goes straight to infinity or crashes, with an incorrect answer size or whatever, and I will endeavour to show you some kind of debugging techniques as we go, but it's one of the hardest things to teach because, like I don't know, maybe I just have quite a Fewget it out yet, but it's like the main thing it requires is tenacity. I find like the biggest difference between the people. I've worked with who are super effective and the ones who don't seem to go very far has never been about intellect it's always been about. You know, sticking with it, basically never never giving up. So it's particularly important with this kind of deep learning stuff, because you don't get that continuous reward cycle like with normal programming. You've got like 12 things to do until you've got your Flash endpoints staged up. You know in at each stage. It's like. Okay, we have successfully processing the JSON, and now we successfully you know, I've got the callback from that promise and now I successfully created the authentication system. Like you know, it's this constant sequence of like stuff that works where else generally with training the model. It's a constant stream of life, it doesn't work, it doesn't work, it does.

Okay. So let's see a look at the images so you'll find inside the GOC dev kit, there's 20 toy 2007 and 2012 directories and in there there's a bunch of stuff. That's mainly these XML files, the one we care about the JPEG images, and so again here you've got path, tips, slash operator and inside there's a few examples of the images okay. So what I wanted to do was

12. 00:54:15

- **Mendeley Desktop, an app to track research papers**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

To create a dictionary where the key was the image ID and the value was a list of all of its annotations. So basically, what I wanted to do was go through each of the annotations that doesn't say to ignore it and append it. The bounding box and the class to the appropriate dictionary item where that dictionary item is a list, but the annoying thing is, of course, is that if that dictionary item doesn't exist yet then there's no list to the pen too, so one super handy trick in Python Is that there's a class called collections default depth, which is just like a dictionary, but if you try and access a key that doesn't exist, it magically makes itself exist and it sets itself equal to the return value this function now. This could be the name of some function that you've defined or it can be a lambda function. A lambda function simply means

it's a function that you define in place, we'll be seeing lots of them. So here's an example of a function. All the arguments to the function are listed on the left, so there's no arguments to the function and lambda functions. A special you don't have to write return as there a return is assumed. So, in this case, this is a lambda function that takes no arguments and returns an empty list, so in other words, every time I try and access something in train annotations that doesn't exist now does exist it as an empty list, which means I can go into It okay, one comment on variable: naming is

13. 00:56:15

- **arXiv-Sanity.com**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

When I read through these notebooks I'll generally try and like speak out, the English words that the variable name is a limit for a reasonable question would be well. Why didn't I write the full name of the variable in English rather than using a short demonic. It's a personal preference. I have based on a number of programming communities where the basic kind of thesis is that the more that you can see in a single kind of I grab of the screen, the more you can like understand intuitively. That won't go every time. You have to your eye has to jump around it's kind of like a context, change that reduces your understanding, it's a style of programming. I found super helpful and so generally speaking, I try to. I particularly try to reduce the vertical height, so things don't scroll off the screen, but I also try to reduce the size of things so that there's a mnemonic there, which, if you know it's training annotations, it doesn't take long view to see the patient's. You know throughout the whole thing yet, so I'm not saying you have to do it this way. I'm just saying there's some very light programming communities, some of which have been around for 50 or 60 years which refused this approach, and it's interesting to compare like. I guess my philosophy is somewhere between math and Java. You know like in math, everything is a single character.

The same single character can be used in the same paper for five different things and depending on whether it's in italics or bold, faced with capitals, another fire in Java, you know variable names, sometimes require a few pages. Well. So for me, I personally like names, which are you know short enough to not take too much of my. You know perception to see it once, but long enough to have a mnemonic. Also. However, a lot of the time the variable will be describing a mathematical object as it exists in the paper and there isn't really an English name for it, and so, in those cases I will use the same like often single letter that the paper uses right and So if you see something called

14. 00:59:00

- **Jeremy on twitter.com and reddit.com/r/MachineLearning/**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Delta or a or something, and it's like something inside an equation from a paper, I generally try to use the same thing just to explain that yeah and by no means do you have to do the same thing. I will say, however, if you contribute to first day I I'm not particularly fastidious

about coding style or whatever, but if you write things more like the way I do than the wage are, but people do okay. So by the end of this, we now have a dictionary from file names to at Apple and so here's an example of looking up that dictionary and we get back a bounding box and a-plus you'll see when I create the bounding box. I've done a couple of things: the first years I've switched the X & amp Y, coordinates and the reason for this I think we mentioned this briefly in the last course. The kind of computer vision world when you say like. Oh, my screen is 640 by 480. That's width by height, or else the math world. When you say my array is 640 by 480. It's rows by colors, so you'll see that a lot of things like pil, pillow image library in Python tend to do things in this kind of width by height or columns by rows way. Numpy is the opposite way around. So I again, my view is: don't put up with it's kind of incredibly annoying inconsistency fix it right, so I've decided fastai is you know the lump I pytorch way is the right way, so I'm always rows by cons so you'll see here. I sketched my rows of columns.

I've also decided that we're going to do things by describing the top left, XY coordinate and the bottom right XY coordinate the bounding box rather than the XY and the eye width. Okay, so you'll see here, I was converting the the

15. 01:01:15

- **Neural style notebook (continued)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Height and width to the top, so you know again, it's kind of like I often find dealing with junior programmers and particular junior data scientists that they kind of get given datasets that are in shitty formats or happy api's, and they just act as if everything has To be that way, but your life would be much easier if you take a couple of moments to make things consistent, make them the way you want to be okay, so earlier on, I took all of our classes and created a categories list, and so, if we Look up category number 7, which is what this year's veteran on the 7 is car. Let's have a look at another example. Image number 17 has two bounding boxes. One of them is of type 15, one some type 13, that is a person and a horse. So this would be much easier to understand if we can see a picture of these things. So let's create some pictures so having just turned our height width stuff into top left bottom right stuff. We're now going to create a method to do the exact opposite, because anytime, I want to call some library that expects the opposite, I'm going to need to pass it in the opposite. So here is something that converts a bounding box to a hiking, with B bhw. The bounding box, okay, so it's again reversing the order and credit and giving us the height width. So we can now open an image in order to display it and where we going to get to is we're going to get it to show that sets that car? We just sorts it out right, so one thing that I often get asked on the forums or through github is like well.

How do I find out about this open image thing? Where did it come from? What does it mean who uses it? And so I wanted to just to take a moment, because what other things are going to be doing a lot? And although a lot of you aren't professional coders, you have backgrounds in statistics or you know, meteorology your physics or whatever, and I apologize for those of you that are professional coders. You know this already. You need because we're gon na be a lot doing about a stuck with the fastai library and other libraries. You need to go to navigate very quickly through them, okay, and so let me give you a quick overview of how to navigate

through code and for those of you that haven't used an editor properly before this is going to blow your months right.

16. 01:04:05

- **Broadcasting, APL as "A Programming Language", and Jsoftware**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

For those of you that have you're going to be like, so for the demo, I'm going to show you in Visual Studio code personally, my view is that on pretty much every platform, unless you're prepared to put in the decades of your life, to learn beer more In apps, well, Visual Studio code is probably the best editor out there. It's free it's open source. There are other perfectly good ones as well. Okay, also, if you download a recent version of anaconda, it will offer to install Visual Studio code for you. It integrates with anaconda sets it up with your Python interpreter and comes with the Python extensions and everything. So it's it's a it's a good choice, if you're not sure if you've got some other editor, you, like you, know, search for the right keywords in the health. So if I fire up Visual Studio code, the first thing to do, of course, is do a git loan of the faster I library to your laptop you'll, find in the root of the repo, as well as the environment, yml file that sets up a condor environment. 52, you one of the students has been kind enough to create an environment, CPU, yml file, and perhaps one of you that knows how to do. This can add some notes to the wiki, but basically you can use that to create a local CPU, only fastai installation and the reason you might want to do. That is so that, as you navigate the code, you know you'll be able to navigate into pytorch you'll, see all the status is there anyway.

So I open up visual studio code and it's as simple as saying open, folder right and then you can just point it out the faster I get hub, folder that you just downloaded, and so the next thing you need to do is to set up visual studio Code to say, I want to use the fastai, Condor environment place. So the way you do, that is with the select interpreter, command and there's a really nice idea, which is kind of like the best of both worlds, between a command-line interface and a GUI which is you hit. This is the only command in each mode. Ctrl shift P, you hit ctrl shift P, and then you start typing what you want to do and watch what happens. Joseph P, I want to changed my interpreter in okay and it appears if you're, not sure you can kind of try a few different things right. So here we are Python, select interpreter and you can see generally, you can type stuff in it'll. Give you a list of things if it can and so here's a list of all of the environments interpreters. I have set up and here's my fastai environment, okay, so that's basically the only setup that you have to do. The only other thing you might want to do is to know there's an integrated terminal, and so, if you hit ctrl backtick, it brings up the terminal and you can the first time you do it it'll ask you: what terminal do you want if you're in Windows It'll be like PowerShell or command prompt or if you're on Linux, you've got more shells installed and asked so

17. 01:07:15

- **Broadcasting with Keras**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

As you can see, I've got it set up to use. Okay and you'll see it automatically goes to the directory. Alright, so the main thing we want to do right now, let's find out what I couldn't understand is so the only thing you need to know to do that is control teeth. If you hit ctrl T, you can now type the name of a class function. Pretty much anything and you find out about it so open image you can see it appears and it's kind of cool if there's something that's got like camelcase capitalized or something that underscore you can just type the first few letters of each pitch. So I could be like open image. For example, look I do that and it's found the function. It's also found some other things that match. Oh there, it is okay, so that's kind of a good way. You can see exactly where it's come from when you can find out exactly what it is and then the next thing I guess would be like. Well, what's it used for so? If it's used inside fastai, you could say find references which is shift. O smoke set up should say, shift open image shift f12 and it brings up something saying: oh, it's used twice in this codebase and I can go and I can have a look at each of those examples. Okay, if it's used in multiple different files, it'll tell you the different files that it's used in it. Another thing, that's really handy, then, is, as you look at the code, you'll find that certain bits of the code call other parts of the code.

So, for example, if you're inside files data set and you're like oh, this is calling something called open image. What is that? Well, you can wave your pointer over it and it'll give you the doc string or you can hit f12, and it jumps straight to its definition. Right. So like often it's easy to get a bit lost in, like things call things cool things and if you have to manually go to each bit, it's if you're ready for us this way. It's always one button way right, ctrl T to go to something that you're specific. You know the name of or f12 to jump to the name, the definition of something that you're clicking on and when you're done, you probably want to go back where you came from so alt left takes you back to where you were okay, so whatever you use Bm Emacs Adam whatever they all have this functionality. As long as you have an appropriate extension installed, if you use pycharm, you can get that for free. That doesn't need any exchange. It's Python! You know whatever you're using. You want to know how to do this stuff. Finally, I mentioned there's a nice thing, called sin mode, ctrl, K, Z, which basically gets rid of everything else, so you can focus, but it does keep this nice little thing on the right hand, side, which kind of shows you whele, okay, so that's something that you Should practice if you haven't played around with it before during the week, because we're increasingly going to be, you know, digging deeper and deeper into faster iron, pipe or fibers? As I say, if you're already a professional coders know all this stuff - apologies for telling you stuff - you know okay, so we're going to well.

Actually, since we did that, let's just talk about open image, you'll see that we're using cv 2 cv 2 is the library is actually the opencv library. You might wonder why we're using open CV - and I want to explain some of the units of fastai to you, because some of them are kind of interesting and might be helpful to the torch vision like the standard kind of pytorch vision. Library actually uses apply torch tensors for all of its. You know: data augmentation and stuff, like that, a lot of people use pillow Pio a standard of Python, imaging library I found I did like a lot of testing of all. Of these I found open CV was about 5 to 10 times faster than to watch vision, so early on actually teamed up with one of the students from an earlier class to do the planetlab satellite competition back

18. 01:12:00

- **Recreate input with a VGG model**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to

When that was on and we used to watch vision and because it was so slow, we could only get like 25 % GPU utilization, because we were doing a lot of data augmentation. And so then I use the profiler to find out what's going on and realized. It was all in in to watch, vision, pillow or PIL is quite a bit faster, but it's not as fast as open CV. It also is not nearly as threadsafe, so I actually talked to the guy who developed the the thing that python has. This thing called the global interpreter, lock this before the GI L, which basically means that true fred's can't do do pythonic things at the same time. Yes, but it makes python a really shitty language, actually the modern programming, but they're stuck with it. So I spoke to the guy on Twitter who actually made it so that open CV releases the GIM, so one of the reasons the faster your library is so amazingly fast is because we don't use multiple processors, like every other library does for our data organization. We actually do multiple threats and the reason we can do multiple threads is because we use it and see bit. Unfortunately, OpenCV is like a really shitty API, it's kind of inscrutable a lot of stuff. It does this point documented as they're poorly documented. It's documented but like in really obtuse kind of ways. So that's why I try to make it so like no one using fastai needs to know that it's using a CD you know like.

If you want to open an image, do you really need to know that you have to pass these flags to open to actually make it work? Do you actually need to know that if the reading fails, it doesn't show an exception? It just silently returns. Now you know it's these kinds of things that we try to do to actually make it work. Lastly right, but as you start to dig into it, you'll find yourself in these places and you're kind of want to know. You want to know what - and I mentioned this in particular to say: don't start using you know height or your data orientation, don't start bringing in pillow you'll find suddenly things slow down horribly or the body threatening won't work anymore or whatever. I try to stick to using OpenCV for your processing, okay, so so we've got our image, we're just going to use it to to demonstrate the pascal library, and so the next thing I wanted to show you in terms of like important coding, stuff we're going to Be using throughout this course is is using matplotlib a lot better, so matplotlib is so named because it was a rich, a clone of matlab's flooding. Later, unfortunately, MATLAB matlab's plotting library is awful, but at the time it was what everybody knew. So at some point the matplotlib folks realized - or they probably always view that the MATLAB plotting library is lawful, so they added a second API to it, which was an object-oriented API. Unfortunately, because nobody who originally learned that plot, let let the OO API, they then taught the next generation of people, the old MATLAB style, API and now there's basically no examples or tutorials online.

I'm aware of that use the much much better easier to understand simpler ago. So one of the things are going to try and show you, because plotting is so important in deep learning is how to use this API and I've discovered some simple little tricks. One simple little trick is plot. Subplots is just a super handy wrapper. I'm going to use it lots right and what it does is it returns two things. One of the things you probably won't care about. The other thing is an axes, object and basically anywhere where you used to say, PLT dot, something you now say: ax dot something, and it will now do that plotting to that particular sub bike. So a lot of the time you'll use this or I'll use this during this course to kind of plot, multiple plots that we can compare next to each other, but even in this case, I'm I'm creating a single plot, alright, but it's just it's just nice to Only know one thing rather than lots of things so, regardless of whether you doing one plot and lots of plots, I always start now with with this, that I was right, and the nice thing is that this way I can pass in an access object. If I want to plot it into a figure I've already created or if it hasn't been passed, you know I can create so this is also a nice way

to make your matplotlib functions like really versatile and you're kind of see this used throughout this course.

So now, rather than plot that I am show it's a yesterday on show okay and then, rather than kind of weird stateful setting things in in the old-style API, you can now use ooohs. You know, get access that returns an object except visible, that's a property! It's all pretty normal, straightforward stuff. So once you start getting the hang of a small number of these oo, matplotlib things hopefully you'll find life a little easier. So I'm going to show you a few right now. Actually, so let me show you a cool example. What I think is a cool example, so one thing that kind of drives me crazy, with people putting text on images, whether it be subtitles on TV or people doing stuff with computer vision. Is that it's like white text on a black background or black text on a black background? You can't read it, and so a really simple thing that I like to do. Every time I draw on an image is to either make my text in boxes white with a little black border, or vice versa, and so here's a like cool little thing you can do in matplotlib is you can take a matplotlib plotting object and you can go Set path, effects and say add a black stroke around it, and you can see that then, when you draw that, like it doesn't matter that here, it's white on a white background right or here at some black background, it's equal and like it's just.

I know it's a simple little thing, but it kind of just makes life so much better when you can actually see your bounding boxes and actually read the text, so you can see, rather than just saying add a rectangle. I get the object that it creates and then pass that object to draw outline now everything I do that again, this nice path effect runner, you can see. Matplotlib is perfectly convenient way of drawing stuff alright. So when I want to draw a rectangle matplotlib calls that a patch and then you can pass it all different kinds of patches. So here's again, you know, rather than having to remember all that every time please take another function. Alright, now you can use that function. Every time you know you don't have to put it in a library somewhere. I always put lots of functions inside my notebook. If I use it in like three notebooks, then I know it's useful enough that I'll stick it in a separate library. You can draw text and notice. All of these take an axis object right, so this is always going to be added to whatever thing I want to add it to right, so I can add text and outline around it. So, having done all that, I can now take my show image, which and notice here the show image. If you didn't pass it an axis, it returns, the axis it created right so show image returns returns the axis that image is on. I then turn my bounding box into height width for this particular images, bounding box. I can then draw the rectangle.

I can then draw the text in the topple in the top left corner. So remember, the bounding box x and y are the first two coordinates right. So the column to the top left - this is the remember. The top all contains two things: the bounding box and then the class. So this is the class and then to get the text of it. I just pass it into my categories list and there we go. Okay, so now that I've kind of got all that set up, I can use that for all of my object, detection stuff from here all right. What I really want to do, though, is to kind of package all that up, so here it is packaging it all. It up so here's something that draws an image with some annotations right, so it shows the image that goes through each annotation turns it into height and width draws the rectangle Roza test. Okay, if you haven't seen this before each annotation, remember, contains a bounding box and a class so rather than going for o in a and n and going o 0 or 1, I can D structure it. Okay, this is a D structuring assignment. So if you put something on there, something on the left, then that's going to put the two parts of a top-off or a list into those two things to bandy. So for the bounding box and the class in the annotations go ahead and do that and so then I can then say: ok draw a image of particular index by grabbing the image ID

opening it up and then calling that draw. And so let's test it out and there it is okay.

So you know that kind of seems like quite a few steps, but to me, when you're working with a new data set like getting to the point that you can rapidly explore it, it pays

19. 01:22:45

- **Optimize the loss function with a deterministic approach**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Off right, you'll see, as we start building our model, we're going to keep using these functions now to kind of see how things go alright, so step, one from our presentation is to do a classifier, okay, and so I think it's always good like for me. I didn't really have much experience before I started preparing this course a few months ago in doing this kind of object, detection stuff, so I was like alright, I want I want to get this feeling of, even though it's deep learning of continual progress all right so, Like what could I make work all right? Well, why don't? I find the biggest object in each image and classifier. I know how to do that all right, so it's like this is one of the biggest problems I find today with the younger students. If they figure out the whole big solution, they want generally, which involves a whole lot of new speculative ideas. I tried before, and they spend six months doing it and then the day before the presentation, none of it works and this roof right. Where else like I've talked about my approach to Kabul, competitions before I was just like half an hour, if you go at the end of that, half an hour, submit something right and try and make it a little bit better than yesterday's. So I kind of tried to do the same thing in preparing this lesson right, which is try to create something that's bit better than lasting okay.

So the first thing was like the easiest thing I could come up with was my largest item classifier. So the first thing I needed to do was to go through each of those each of the bounding boxes in an image and get the largest one right. So I actually didn't write that first. I actually wrote this first right. So normally I like pretend that somebody else has created the exact API. I want and then go back and write right. So I kind of I wrote this phone first and it's like okay. I need something which takes all of the bounding boxes for a particular image and finds the largest and well that's pretty straightforward. I can just sort the bounding boxes and here again we've got a lambda function. So again, if you haven't used lambda functions before this is something you should study during the week right, they're used all over the place to quickly define a function or like a once-off function, and in this case the pythons dead. The Python built-in sorted function lets you pass in a function to say how do you decide whether something's earlier or later, in the sort order? And so in this case I took the product of the last two items of my bounding box list. Ie the bottom right hand corner the two items of my bounding box: vest a the top left corner. So bottom right couple left is the size, the two sizes, and if you take the product of those two things you get the size of any boss and so then that's the function do that in descending order I mean often um often you can take something.

It's gon na be a few lines of code and turn it into one line of code, and sometimes you can take that too far. But for me I like to do that. You know where I reasonably can, because again it means like, rather than having to understand a whole big chain of things. My brain can just say like. I can just look at that at once and say: okay, there. It is, and also I find it

over time. My brain kind of builds up this little library of idioms. You know and like more and more things, I can look at a single line and know what's going on okay, so this bill is a dictionary and it's a dictionary, because this is a dictionary comprehension. A dictionary comprehension is just like a list. Comprehension, I'm gon na use it a lot in this part of the course except it goes inside curly brackets and it's got a key colon value all right. So here the key is going to be the image ID and the value is the largest angles. Okay. So now that we've got that we can look at an example and here's an example of the largest bounding box for this image. Okay, so obviously there's a lot of objects here, there's three bicycles and three people: okay, but here's the largest bus - and I feel like this - ought to go without saying. But it definitely needs to be said because so many people don't do it. You need to look at every stage when you've got any kind of processing pipeline if, if you're as bad at coding, as I am everything you do, will be wrong the first time you do it right, but like there's lots of people that are as bad as Be according and yet lots of people write lines minds of code, assuming they're all correct and then at the very end, they've got a mistake and they don't know where it came from right. So particularly when you're working with images, write or text like things that humans can look at and understand, keep looking at it right so here I have it yep.

That looks like the biggest thing, and that certainly looks like this. So, let's move on here's another nice thing in path: Lib make directory okay method, so I'm going to create a path called CSB, which is a path to my large objects. Csv file. Why am I going to create a CSV file? Pure laziness? Right? We have an image classifier dot from CSV, but I could go through a whole lot of work to create a custom data set and blah blah blah to use this particular format. I have but why you know it's so easy to create the CSB check it inside a temporary folder and then use something they already have right. So this is kind of a something I've seen a lot of times on the forum is people will say like how do I convert this weird structure into a way that first day I can accept it, and then normally somebody on the forum will say like print It to a CSV file, so that's a good, simple tip and the easiest way to create a CSV file is to create a pandas data frame all right. So here's my pandas data frame - I can just give it a dictionary with the name of a column and the list of things in that column. So there's the file name, there's the category and then you'll see here. Why do I have this? I've already named the columns in the dictionary. Why is it here? Because the order of columns matters all right and a dictionary does not have an order? Okay, so this says the file name comes first in the category list, all right, so that's a good trick to creating a CSV. So now it's just dogs and cats.

Right I have a CSV file. It contains a bunch of file names and for each one it contains the plus of that object. So this is the same two lines of code 15,000 times. What we will do, though, is to like take a look at this. The one thing that's different is crop type, so you might remember, the default strategy for creating whatsis is here to 24, a to 24 by to 24 image in fastai is to first of all resize it so the largest side. Sorry, the smallest side is to 24 and then to take a random crop, assuming it's rectangular, a random square run during training and then during validation. We take the center crop unless we use data augmentation, in which case we do a few ran across for bounding boxes. We don't want to do that because, unlike an image net, where the thing we care about is pretty much in the middle and it's pretty big a lot of the stuff in object. Detection is quite small and close to the edge, so we could crop it out, and that would be bad. So when you create your transforms, you can choose crop type, equals crop type, got no and no means don't crop and therefore to make it square. Instead, it squishes it so you'll see this guy now looks kind of a bit strangely wide right, and that's because he's been squished like this. Okay and generally speaking, a lot of computer vision models work a little bit better if you crop rather than squish, but they still work pretty well. If you squish right - and in this case we definitely don't want to crop.

So this is perfectly fine right, so we you know if you had like very long or very or images that you know such that if a human looked at the squashed version, if you like that, looks really weird, then that difficult, but in this case we're just Like so the computer whoa okay, so I'm going to kind of quite often just dig a little bit into some more depths of fastai and pipe torch. In this case, I want to just look at data loaders a little bit more, so you already know that, let's just make sure this is all run so

20. 01:33:25

- **Visualize the iterations through a short video**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You already know that inside a model data object when there's lots of model data subclasses like image, classifier data. We have a bunch of things which include training, data, loader and the training data set all right and we'll talk much more about this. So, but the main thing to know about training about a data loader is that it's an iterator that each time you grab the next iteration of stuff from it, you get a mini batch. Okay and the mini that you get is of whatever size you asked for, and by default, the batch size is 64. Okay, you can pass below um, however, so in Python, the way you grab the next thing from an iterator is with next right. You can't just do that right and why can't you just do that? The reason you can't do that is because you need to say, like start a new epoch now right in general, like this, isn't just in pi choice but for any Python, iterator, you're kind of need to say start at the beginning of the sequence. Please all right, and so the way you do that - and this is a general Python concept. Is you write it up and it says please grab an iterator out of this object right specifically, as we will learn later, it means this class has to have to find an underscore underscore header underscore underscore method, which returns some different object, which then has an underscore Underscore next underscore underscore yes right! So that's how I do that right, and so, if you want to grab just a single batch, this is how you do it X, comma y, equals next in a data load that Y X, comma Y, because our our lab data loader is our data.

Sets behind the daily loaders always have an X. You know the independent in the Y, the dependent variable, so here we can grab a mini batch of X's and Y's, and now I'm going to pass that to that show image command we had earlier, but we can't send that straight to show image. For example, here it is for one thing: it's not an umpire right, it's not on the CPU and its shape is all wrong. It's not to 24 by 2. 24. 3, it's 3 by 2. 3. 4. 30. 24. Furthermore, these are not numbers between 0 & amp. 1, why not, because remember all of the standard, imagenet pre-trained models expect our data to have been normalized to have a 0 mean and a 1 standard deviation. So if you look inside see, let's use Visual Studio code for this, that's what we've been doing. So, if you look inside transform the strong model, so ctrl T transforms from model TFM egg, alright, which in turn calls transforms. 12 ashle transports model calls transport stats, and here you can see normalize and it normalizes with some set of image statistics and the set of image statistics they're, basically hard-coded. This is the image snap statistics: this is statistics, user insertion models right so there's a whole bunch of stuff. That's been done to

21. 01:37:30

- **Recreate a style**

The input to get it ready to be passed to a pre train model, so we have a function called denim for denormalize. It doesn't only do normalize. It also fixes up the dimension order and all that stuff right and they look. The denormalization depends on the transport. Okay and the data set knows what transform was used to create it. So that's why you have to go model data dot and then some data set dot d norm and that's a function that it's stored for you. That will undo that all right and then you can pass that a mini batch, but you have to turn it into non bio, first, okay. So this is like all the stuff that you need to be able to do to kind of grab, batches and unlock them, and so, after you've done all that, you can show the image and we've got that catalyst. So that's looking good. So, in the end, we've just got this to end of four lines of code. We've got our transforms: we've got our model data come learn about pre-trained we're using your resna 34 here, I'm gon na add accuracy as a metric fix some optimization function to an LR find and that looks kind of weird, not particularly helpful. Normally, we would expect to see a uptick on the right. The reason we don't see it is because we intentionally remove the first few points in the last few points. The reason is that often the last few points shoots so high up towards infinity that you basically can't see anything so the vast majority of the time. Removing the last few points is a good idea.

However, when you've got very few mini batches. Sometimes it's not a good idea, and so a lot of people asked us on the forum. Here's how you fix it all right, just say: skip by default. It skips 10 at the start. So in this case we just say 5 by default. It's gives 5 at the end, we'll just say 1, and so now we can see that the shape properly um. If your data sets really tiny, he made it he's a smaller batch size like if you only have like three or four batches worth this. One is not going to see that in this case, so it's it's fine. We just have to plot a little bit more okay, so we pick a learning rate. We say fit after 1 Apoc just training the last player. It's a descent. Let's unfreeze a couple of players: do another epoch, 2 % and freeze the whole thing not really improving. Why are we stuck at 80 % kind of makes sense right, like unlike imagenet or dogs versus cats, where each image has one major thing they were kicked because they have one major thing and the one major thing is what they're asked to look for a lot Of the Pascal data set has lots of little things, and so a largest classifier is not necessarily going to do great. But of course, we really need to be able to see the results to kind of see like whether it makes sense so we're going to write something that creates this, and in this case I'm kind of like I. After working with this a while, I know what the 20 Pascal classes are, so I know there's a person and a bicycle class.

I know there's a dog and I so for class. So I know this is wrong. It should be so forever. That's correct! Yes! Yes, chair, that's wrong. I think the tables bigger motorbikes correct, because there's no cactus there should be a bus person's correct. That's correct, Kasper if plants great cars correct. So that's looking pretty good all right so um when you see a piece of code like this, if you're not familiar with all the steps to get there, it can be a little overwhelming alright and I feel the same way when I see a few lines of Code in something I'm not familiar with, I feel like a 1 as well, but it turns out there's two ways to make. It super super simple to understand the code or there's one high level, where the high level way is run. Each line of code step yeah, step printout the inputs print out the efforts most of the time that'll be enough. If there's a line of code where you don't understand how the outputs relate to the inputs go and have a look for the sauce. So now all you need to know is what are the two ways you can step through the lines of code, one at a time um. The way I use paths the most often is to take the contents of the loop copy it create a cell. Above it paste it out, dent it

right, I equals naught and then put them all in separate cells and then run each one one at a time printing out the inputs. Now I mean - I know that's obvious, but the number of times I actually see people do that when they asked me for help is basically zero, because if they had done that they wouldn't be asking for help another method.

That's super handy and there's particular situations where a super super super handy is to use the Python debugger, who here is used a debugger before so after two thirds so for the other half of here this would be life-changing. Actually, a guy. I know this morning is actually a deep learning researcher wrote on Twitter and his his message on Twitter was how come nobody told me about the Python debugger before my life has changed and like this guy's, an expert but because, like nobody, teaches basic software engineering skills In academic courses, you know nobody thought to say to him: hey Mark, you know what there's something that shows you everything your code does one stair at a time. So I replied on Twitter and I said good news mark not only that every single language in existence in every single operating system also has a

22. 01:44:05

- **Transfer a style**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Debugger and if you google, for language named debugger, it will tell you Harry's right so there's a metal piece of information point in Python. The standard debugger is called PDB, ok and there's two main ways to use it. The first is to go into your code and the reason I'm mentioning this now is because during the next few weeks, if you're anything like me, 99 % at the time you'll be in a situation where your codes not working right and very often it all have Been on the fourteenth mini-batch inside the forward method of your custom, module that it's like. What do you do right and the answer? Is you go inside your module and you wrap that right and if you know it was only happening on the 14th iteration, you type. If I equals 13 right, so you can set a conditional breakpoint, that's put a breakpoint PDB. Is the Python debugger fastai imports it for you? If you get the message that PDB spots there, then you can just say import PD Lee, ok! So, let's try that and justly it's not the most user-friendly experience it just pops up a boss right, but the first cool thing to notice is holders. Should the debugger even works in a notebook all right, so that's pretty nifty. You can also work in the terminal plus, and so what can you do? You can type a trip right and there are plenty of tutorials here, and the main thing to know is this is one of these situations where you definitely want to know the one letter mnemonics right, so you could type next, but you definitely want to talk right.

You could type continue you're. Definitely less. I've listed the main ones you need. So what I can do now that I'm sitting here is like it shows me the line, I'm Kara it's about to run. Okay, so one thing I might want to do is to print out something and I can write any Python expression and hit them up and find it okay. So that's that's a useful thing to do a might want to find out like more about like well. Where am I in the code? More generally, I just want to see this line, but what's the before it and after it, okay, so I want a whole forest right, and so you can see I'm about to run that line. These are the lines above it in the blower, okay um, so I might be like okay, let's run this line and see what happens so go to. The next line is ten okay and you can see now it's about to run the next one. One handy tip you don't even have

to type n if you just hit enter it repeats. The last thing you did so that's okay, so I now should have a thing called beep right. Unfortunately, single letters are often used for debugger commands. So if I just type B, it'll run the big man rather than print B. For me, that's so to force it to print use, P print, okay, so there's bird all right! Fine! Let's do next again right at this point. If I hit next it'll draw the text, but I don't want to just draw the text, I want to know how it's going to draw the text, so I don't put no next over it. I want to ask step into it.

So if I now hit s to step into it, I'm now inside draw test, and I now hit n. I concede your text and so forth. Okay and then I'm like okay. I know everything I want to know about this. I will continue until I hit the next breakpoint, so C will continue. What if I was zipping along this happens quite often that, like let's step into Dean on here, I am inside Dean on and what will often happen is if you're debugging, something in your pytorch module and it's hidden exception and you're. Trying to debug you'll find yourself like six layers deep inside pytorch, but you want to actually see backup what's happening when you called it from right. So in this case I'm inside this property, but I actually want to know what was going on up. The call stack I just hit you and that doesn't actually run in a thing. It just changes the context of the debugger to show me what called it, and now I can type. You know things to find out about that: environment, okay and then, if I'm gon na go down again, it's deep okay, so, like I'm, not gon na show you everything about the debugger, but I just showed you all of those commands right. Yes, there's a Oh something that we found helpful as we've been doing. This is using from ipython court a debugger imports a trace, and then you get a all prettily colored. It's usually excellent tip. Let's learn about some of our students here. Is it tell us? I know you were doing an interesting project. Can you tell us about it? Okay, hello.

Everyone I mean is a here with my uh, my collaborator Britt and we're using this kind of stuff to try to build a Google Translate for animal communication yeah. So that involves playing around a lot with like unsupervised, machine, neural translation and doing it on top of audio. Where do you get data for that from ah that's sort of the hard problem, so there you have to go and like we're talking to a number of researchers to try to collect and collate large data sets, but if we can't get it that way, we're thinking About building a living library of the audio of the species of Earth that involves going out and like collecting a hundred thousand hours of like gelada monkey vocalization, so all right, that's great here! Okay, so let's get rid of that set trace um the other place that the debugger comes in particularly handy is, as I say, if you've got an exception all right, particularly if it's deep inside pipes watch. So if I like, when I times 100 here, obviously that's gon na in exception, I've got rid of the set trace. So if I run this now, okay something's wrong. Now, in this case, it's easy to see what's wrong right, but like often it's not so what do I do? Percent debug pops open the debugger at the point, the exception that okay, so now I can check like okay, creds Len crits, 64. 5 times 100. I've got a print that size and 100, oh, no one, okay and you can go down the list. Okay, so I do all of my development, both with the library end of the lessons in G but a notebook.

I do it all interactively and I use you know - percent debug. You know all the time, along with this idea of, like copying stuff out of a function of putting in a desert of cells, running it step by step. There are similar things you can do inside, for example, Visual Studio code, there's actually Jupiter extension, which lets you select any line of code inside Visual Studio code and it'll and say run in Jupiter, and it will run it in Jupiter and create a little window showing You, the output, there's neat little stuff like that. Personally, I think Jupiter notebook is better and perhaps by the time you watch this on the video you know the lab or me the main thing give it a lab selection in the next version of Jupiter notebook pretty similar Wow. I just broke it totally: okay! Well, we know exactly how to fix it, so we were worried about that. Another

time - hey debug it this evening, okay, so to kind of do the next stage we want to create the bounding box. Okay and now creating the bounding box around the largest object may seem like something you haven't done before, but actually it's totally something you've done before. Okay, and the reason is something you've done before is we know that we can create a regression rather than a classification. Here all right, in other words a classification year on there, is just one that has a sigmoid or soft mapped out port and that we use across entropy or binary cross entropy loss function like that's. Basically, what makes it if we don't have the softmax it boys at the end, and we use means Guidera as a loss function. It's now our regression model right, and so we can now use it to predict a continuous number rather than the category.

We also know that we can have multiple outputs like in the planet, competition. We did a multiple object classification. What if we combine the two ideas and to a multiple column regression? So in this case, we've got four numbers top left out and why bottom-right X & amp Y yeah and we could create a neural net with four activations. We could have no softmax or sigmoid and use a mean squared error loss function, and this is kind of like where you're thinking about it like differentiable programming, it's not like how do I create a bounding box model? It's like all right. What do I need? I need four numbers. Therefore, I need a neural network with four activations. Okay, that's traffic. What I need to know, the other half I need to know is a loss function, in other words, what's a function that, when it is lower, means that the four numbers are better, because if I can do those two things, I'm going okay. Well, if the X is close to the first activation and the wires close to the second so forth, then I'm done so that's it. I just need to create a model with four activations with a mean squared error loss function, and that should be it right like we don't need anything new. So, let's try it so again: we'll use a CSV right and if you remember from part one to do a multiple label classification, your multiple labels have to be spaced, separated, okay and then your file name is comma separated. So I'll. Take my largest item.

Dictionary create a bunch of bounding boxes for each one, separated by a space. No use. You know this comprehension I'll then create a data frame like I did before I'll turn that into a CSV and now I've got something. That's got the file name and the four bounding box corners. I will then pass that to from CSV again I will use crop type equals crop type dot. No real next week, we'll look at transform type dot, coordinate for now. Just realize that when we're doing scaling and data augmentation that needs to happen to the bounding boxes, not just images image, classifier data dot, CSV gets us to a situation where we can now grab one mini batch of data. We can do normalize it. We can turn the bounding box back into a height width so that we can show it, and here it is okay, remember we're not doing classification, so I don't know what kind of thing this is it's just a thing, but there is the thing okay. So I now to create a comic debt based on President 34, but I don't want to add the standard, a set of fully connected layers that create a classifier. I want to just add a single linear layer with four outputs. So fastai has this concept of a custom head. If you say my model has a custom head, the head being the thing that's added to the top of the model, then it's not going to create any of that fully connected Network for you, it's not going to add the adaptive average pooling for you, but instead It'll add whatever model you asked for so in this case I've created a tiny model.

It's a model that flattens out the previous layer. So remember, I'm normally would have a seven by seven by I think 512 previous layer in risen at 34. So it has flattens that out into a single vector of length, 2508 fat and then I just add a linear layer that goes from 2508, eight to four there's my four yeah. So, like that's the simplest possible kind of final layer, you could

add. I stick that on top of my pre-trained risen at 34 model, so this is exactly the same as usual, except I've just got this custom here, all right, optimize it with atom user criteria, I'm actually not going to use MSC, I'm going to use l1 loss. So I can't remember recover this last week. We can revise it next week if we did it, but l1 loss means rather than adding up the squared errors, add up the absolute values of years. So it's like it's. It's normally. Actually what you want, adding up. The squared errors really penalize -- is bad misses by too much so l1 loss is generally better to work with okay I'll come back to this next week, but basically you can see what we do now is we do our ela find find our learning rate learn For a while freeze to learn a bit more freeze, three learn a bit more and you can see this validation loss, which remember, is the absolute value mean of absolute value with pixels were off by gets, lower and lower, and then, when we're done, we can print Out the bounding boxes, and lo and behold, it's done a damn good job.

Okay, so well revise this a bit more next week, but, like you can see this idea of like if I said to you before this class, do you know how to create a bounding box model? You might have said no nobody's taught me that all right, but the question actually is: can you create a model with for continuous outputs? Yes, can you create a loss function that is lower if those poor outputs are near to four other numbers? Yes, then you're done. Okay, now you'll see if I scroll a bit further down, it starts looking a bit crappy anytime. We've got more than one object, and that's not surprising right because, like how the hell do you decide which birds? So it's just said I'll just pick the middle, which cow I'll pick the middle. How much of this is actually potted plant right this one? It could probably improve, but you know it's got close to the car, but it's pretty weird right, but nonetheless you know for the ones that are reasonably clear. I would say it's done a pretty good job. Okay, all right! So that's time for this week I think J. You know it's been a kind of gentle introduction for the first lesson, if you're, a professional coder, there's, probably like not heaps of new stuff here for you, and so you know. In that case, I would suggest like practicing learning. You know about bounding boxes and stuff if you answer experienced with things like debuggers and that flat live api and stuff like that, there's gon na be a lot for you to practice because we're going to be really assuming you know well from next week.

Okay, thanks: everybody see you next Monday, [Applause,]

Lesson 9: Object Detection

Outline

In today's lesson we'll move from single object to multi-object detection. It turns out that this slight difference makes things much more challenging. In fact, most students found this the most challenging lesson in the whole course. Not because any one piece is highly complex, but because there's a lot of pieces, so it really tests your understanding of the foundations we've learnt so far. So don't worry if a lot of details are unclear on first viewing – come back to this lesson from time to time as you complete the rest of the course, and you should find more and more of it making sense!

Today's focus is on the single shot multibox detector (SSD). This is a way to handle multi-object detection by using a loss function that can combine losses from multiple objects, across both localization and classification. It also uses a custom architecture that takes advantage of the difference receptive fields of different layers of a CNN. And we'll see how to handle data augmentation in situations like this one where the dependent variable requires augmentation too. Finally, we'll discuss and simple but powerful trick called focal loss which is used to get state of the art results in this field.

Video Timelines and Transcript

1. 00:00:30

- **Contribute to, and use Lesson 8 Wiki**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Okay, so today we're going to continue working on object detection, which means that, for every object in a photo with one of 20 classes, we're going to try and figure out what the object is and what its bounding box is. Hi that model to a new data set of unlabeled data and add those medals to the general approach. We're going to use this to start simple and gradually make it more complicated. So we started last week with a simple classifier, the three month of classifier. We then made it slightly more complex to turn it into a bounding box without a classifier today, we're going to put those two pieces together to make a classifier plus a bounding box. All of these are just for a single object, the largest object and then from there we'll roll it up to something which is closer to final goal. You should go back and make sure that you're understanding all of these concepts from last week before you move on. If you don't go back and really go through the locals carefully, I won't read them all to you, because you can see in the video needs to be enough. That practice is the most important knowing how to jump around source code in whatever, but the lambda functions. Lambda function is also particularly important. They come up everywhere, and this idea of a custom head is also gon na, come up in pretty much every lesson. I've also added here a a reminder of what you should know from part one of the course, because, quite often, I see questions on the forum

asking. Basically, why isn't my model working like? Why doesn't it start training or having trained? Why doesn't it seem to be any use and nearly always the answer to the question is: did you print out the inputs to it from a data

2. 00:02:00

- **Experiments on Image/Neural Style Transfer**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Loader, did you print out the outputs from a after evaluating and normally the answer is no and I try printing it and it turns out all the inputs, 0 or all of the appellants negative reports when it was really obvious. So that's just part of something wanted to remind you about this. You need to know how to do these two things. If you can't do that, that's gon na be very hard to debug models and B. If you can do that, if you're not doing it, then it's going to be very happy to debug models. You could have debug models by staring at a source code, hoping your error, pops out your debug models by checking all the intermediate steps. Looking in the data printing, it out plotting its histogram, making sure it says okay, so we were working through Pascal notebook and we just quickly zipped through the bounding box of the largest object without a classifier, and there was one bit that I skipped over and said. I've come back to so. Let's do that now, which is to talk about augmentations data augmentations of the the Y of the dependent variable before I do. I just mention something pretty awkward in all this, which is I've got here image classifier data continuous equals true. This makes no sense whatsoever. A classifier is anything where the dependent variable is categorical or binomial, as opposed to regression, which is anything with the pen and variable is continuous, and yet this parameter here continuously was true, says that the dependent variable is continuous. So this claims to be creating data for a classifier where the dependent is continuous.

This is the kind of awkward rough edge that you see when we're kind of at this. Like you know, at the edge of the pasta, our code is not quite solidified. Yes, so probably by the time you watch this in the MOOC, this will be sorted out. This is before even you regress it or something like that, but you know I just wanted to kind of point out this this issue and also because sometimes people were getting confused between regression, business classification, and this is okay. So let's create some data augmentations right there. Normally, when we create data augmentations, we tend to type in like transform, sidon or transform spa gym. But if you look inside that fast, they are transforms. Module you'll, see that they are simply defined as a list. So this one called transforms basic, which is 10 degree, rotations, plus 0.05 brightness and contrast, and then sidon adds to that and random horizontal flips or else top down as to that random dihedral group of symmetry flips, which basically means, if we possible 90 degree rotation optionally. So like these are just a little shortcuts that I added because time, but you can always create your own list of augmentations right and, if you're not sure what augmentations are there, you can often see check the past AI source or, if you just start typing random. They all start with, so you can see.

So, let's take a look at what

3. 00:05:45

- **Advanced tips from Keras on Neural Style Transfer**

Happens if we create some data, augmentations create a model data object and let's just go through and rerun the iterator a bunch of times, and we all do two things we'll print out the bounding boxes. And so you can see the value box is the same time and we will also draw the pictures so you'll see this lady is, as we would expect, flipping around and spinning around getting darker and lighter, but the bounding box, a is not moving and B, is In the wrong spot, so this is the problem with data augmentation, when your dependent variable, you is pixel values or is in some way connected to your independent variable, its who need to be augmented together and in fact you can see that from the printout. These numbers are bigger than two to four, but these images are of size, two to four that what we requested in this in this transpose, and so it's not even being like scaled or cropped or anything right. So you can see that how the pendant variable needs to go through all of the same geometric transformations without independent variable. So to do that every transformation has an optional transform Y parameter. It takes a transform type-in um. The transform type-in um has a few options. All of which we'll cover in this course, the co ward up ssin, says that the Y values represent coordinates in this case.

Bounding box coordinates okay, and so, therefore, if you flip, you need to change the coordinate derivative represent that flip or if you rotate you to change the coordinate, represent a rotation, so I can add, transform type a chord to all of my augmentations. I also have to add the exact same thing to my transforms from model function, because that's the thing that does the cropping, and/or, zooming and or padding and or resizing, and all of those things need to happen to the dependent variable. So if we add all of those together and rerun, this you'll see the bounding box changes each time and you'll see us in the right spot now, you'll see. Sometimes it looks a little odd like here. Why is that bounding box there and the problem is. This is just a constraint that the information we have right, the bounding box does not tell us that actually, her head isn't way over here in the top left corner. Alright. But actually, if you do a thirty degree, rotation in her head was over here, the top left corner, then the new bounding box would need would go really high right. So this is actually the correct bounding box based on the information and has available, which is to say this is this is how higher Mobe so basically, you've got to be careful of not doing to higher rotations with bounding boxes, because there's not enough information for them To stay totally accurate, just fundamental limitation of the information we're given if we were doing like polygons or segmentations or whatever we wouldn't have this problem.

Okay, so I'm gon na do a maximum of three degree rotations, I'm also going to only rotate half the time. My random flip, my brightness/contrast changing and so there's my set of transformations that I can use. So we briefly looked at this custom head idea, but basically, if you look at dot summary dot summary does something pretty cool, which is. It basically runs a small batch of data through a model and prints out how big it is at every every layer, and we can see that at the end of the convolutional section before we get the flatten it's 512 by 7, by 7, okay and so 512. By 7, 5 7 10 serve breakfast reach answer at that size. If we flatten it out into a single rank, one tensor into a vector, it's going to be 225 thousand a ninety eight long right,

4. 00:10:15

- **More tips to read research papers &**
- **“A Neural Algorithm of Artistic Style, Sep-2015”**

So then, that's why we had this linear layout to 500 for the bosses right. So stick that on top of a pre-trained ResNet and trade it for a while, okay. So that's where we got to last time. So, let's now put those two pieces together so that we can get something that classifies and does bounding boxes and there are. There are three things that we need to do basically to train a neural network ever right. We need to provide data. We need to pick some kind of architecture and we did a loss function. Okay, so the loss function says you know something anything that gives a lower number here is a better network using this data in this architecture. So we're going to need to create those three things for our classification plus bounding box regression. So that means we need a model data object which has as the independence the images and as the dependence I want to have a tupple the first time. One of the tuples should be the bounding box coordinates and the second element, as a couple should be there class. Okay, there's lots of different ways. You could do this. The particularly lazy and convenient way I came up with was to create two mobile data objects representing the two different dependent variables I want so one with the bounding box coordinates one of the classes just using the CSVs. We go over four, and now I'm going to merge them together.

So I create a new data set class and a data set class is anything which has a length and an index service or something that lets. You use it as like lists, and so in this case I can have a constructor which takes an existing data set and the second dependent that I want the length then is just obviously the length of the data set. The first data set and then getitem is grab the X and the y from the data set that I passed in and return that X and that Y and the ayth of the second so there's a data set. That basically adds in a second. As I said, there's lots of ways you could do this, it's kind of convenient, because now what I could do is I can create training data set in the validation data set. Based on that, so here's an example: you can see. It's got a couple of the bounding box coordinates in the class. We can then take the existing training and validation data loaders. Now so you replace their data sets with these and unknown okay. So we can now test it by grabbing a mini batch of data and checking it says: okay, so there's one way to customize data set. So what we're going to do this time now is: we've got the data, so now we need an architecture, so the architecture is going to be the same as the architectures that were used for the classifier and for the bounding box, aggression, but we're just going to Combine them so, in other words, if there are C classes, then the number of activations we need in the final layer is 4 plus C for panama's coordinates and the C probabilities one per class.

So this is the final layer, a linear layer that has four plus men of categories divisions. The first player is before is a flattened. We could just join those up together, but in general I want my my custom head to like, hopefully be capable of solving the problem that I give it on its own. If the pre-trained backbone is connected to is, you know, is appropriate, and so in this case I'm thinking. Okay, I'm trying to do quite a bit here, two different things: there classifier and balanced regression. So just a single linear layer doesn't sound like enough. So I put in a second millionaire okay, and so you can see we basically go over Al. You drop out Lydia, rarely veteran or dropout yeah. If you're wondering why there's no better on that here at the resonant backbone, it already has a match norm as it's. Finally, out okay, so this is basically nearly the same custom headers before it's just. It's got two linear layers, one and nonlinearities. Okay. So that's piece to but data we've got architecture. Now we need a loss function, so the loss function needs to look at these four plus C activations and aside. Are they good right? Are these numbers accurately reflecting the position and class of the largest

object in this image? We we know how to do that. For the last, for the first four, we use l1 loss, just like we did in the bounding box regression before remember l1 loss is like mean squared error.

Rather than sum of squares is some of our values and then for the rest of the activations. We can use cross-entropy loss, so let's go ahead and do that so we're going to create something called detection, loss and loss functions, always take an input and a target. That's what pytorch always calls them. So this is the activations. This is the ground truth. So remember that our our date, custom data set returns. A tuple containing the bounding box coordinates in the classes. So we can D structure that use D, structuring assignment, to grab the bounding boxes and the classes of the target, and then the bounding boxes and the classes of the input are simply the first four elements of the input and the four onwards. Elements of the universe and remember, we've also got a batch dimension. So that's it. We've now got the bounding box target bounding box import class target plus input further bounding boxes. We know that they're going to be between 0 & amp 2 to 4. The coordinates because that's how they got images - okay, so let's grab a sigmoid to force it between 0 & amp, 1, multiply it by 2 to 4 and that's just helping our neural net. You knowget close to what we you know be in the range we know it has to be as a general rule. Is it better to put bash norm before or after RL you? I would suggest that you should put it after a value, because batch norm is meant to move towards a computer, one random variable and if you put value after it, then you're truncating it 0. So there's no way to create negative numbers about a huge port value.

Lembeck norm, having said that, and I think that that way of doing it gives slightly better results. Having said that, it's not too big a deal either way and you'll see during this part of the course most of the time. I go well you and then vet norm, but sometimes they go vetch moment in value. If I'm consistent, so I think originally the batch normal is put it off to the activation, so, okay, so so this is kind of to help our data or force our data into the right range, which you know if you can do stuff like that, it makes It easier to Train yes, Rachel, one more question: what's the intuition behind using dropout with P equals 0.5 after a batch norm, does a batch alarm already do a good job of regularizing fessional doesn't okay Java vaporizing and if you think that to part one we have Had that list of things we do to avoid overfitting and adding batch long as one of them is david augmentation, but it's perfectly possible that you'll still be okay. So one nice thing about dropout is that it has a parameter to say how much to drop out, and so that, like parameters, are great like well specifically, parameters that decide how much to regularize. Because it lets you build a. I speak over priced model and then decide on interiorize it so yeah I tend to always drop out and then, if it turns out, I'm you know I'll start with P equals zero and then, as you know, I can just change my grammar without worrying about you Know if I saved a model, I'm gon na be able to load it back, but if I have dropout ladies and one and not in another or load me more, this way it stays consistent.

Okay, so now that I've got my inputs and targets, I can just go: hey calculate the l1 loss and add to it the cross-entropy okay. So that's our that's! Our loss function. It's a surprisingly easy brass. Now, of course, the cross entropy and the l1 lost. Maybe of wildly different scales, in which case in the loss function, the larger one is going to dominate, and so I just ran this in a debugger checked what you don't? You just use print check how big each of the two things were and found if they multiply by 20. That makes them about the same about the same scale as your training, it's nice to print out information as you go, so I also grabbed the l1 part of this and put it in our in a function, and I also created a function for accuracy so that I could make the metrics and so they're down alright, so we can how about something which is

printing out our object, detection loss, detection, accuracy and detection l1 and so chain it for a while and it's looking good a detection. Accuracy is in the low 80s, which is the same as what it was before. That doesn't surprise me because, like ResNet was designed to do classification, so I wouldn't expect us to be able to improve things in such a simple way, but it certainly wasn't designed to do bounding box regression. It was explicitly actually designed in such a way is to be as to kind of not care about geometry. Rather it takes that last seven, most seven creative activations and averages them all together and throws away all of the information that is going wrong.

So so you can see that the when we only train the last layer. The detection l1 is pretty bad running poor and it really improves a lot, but where else the accuracy doesn't improve, it stays exactly the same. Interestingly, the l1, when we do accuracy and bowing boss at the same time, 8.5 seems like it's a little bit better than when we just do bounding box regression.

5. 00:23:00

- **From Style Transfer to Generative Models**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

And if that's counterintuitive to you, then this would be one of the main things to think about. After this lessons, it's a really important idea and the idea is this figuring out figuring out what the main object in an image is, is kind of the the eye part and then figuring out like exactly where the bounding box is and what class it is, is Kind of the easy part in a way, and so when you've got a single network, that's about saying what is the object and where is the object? It's going to share all of the computation about like finding the object, and so all that shared information. All that shared computation is very efficient, and so, when we vet propagate the errors in you know the class and in the place, that's all information. That's going to help the computation around like finding the biggest object. So anytime, you've got multiple tasks, which kind of share some some concept of what those tasks would need to do to complete their work. It's very likely they should share at least some layers of were together and we'll look later today, a place where most of the layers are shared, but are just the last one, isn't? Okay, so you can see. This is some doing a good job as before of anytime. There is just a single major object, sometimes it's getting a little confused. It thinks the main object here is the dog and it's kind of served with the dog, although it's kind of recognize that actually, the main object is us so far, and so the classifier is doing the right thing with the bounding boxes labeling the wrong thing, which Is yours when there are two birds you can only pick one, so it's just kind of hitching in the middle get over these lots of cows and so forth, doing good job with this car all right.

So that's! So! That's that all right there's not much new. There, although in that last bit, we did learn about you know some simple custom data sets and simple custom Lots functions. Hopefully you can see now how easy that is to do so next stage for me would be to do multi-label classification. So this is this idea that I just want to keep building models that are slightly more complex than the last model, but hopefully don't require too much extra concepts, so I can kind of keep seeing things working and if something stops working, I know that wherever I Find building everything at the same time so model label classification is so easy, is there's not much to mention so we've moved to a Pascal multi. Now that's what we're going to do. The multi object stuff so for the multi object, stuff, I've just copied and pasted the

functions from the previous notebook that we used so they're all at the top. So we can create now a multi class, a multi class CSV file, using the same basic approach that we did last time and I'll mention by the way one of our students actually who's visiting from India. Funny pointed out to me that all this stuff we're doing with default, dicks and stuff like that, he actually showed her a way of doing that which was much simpler, using pandas and he shared that on the forum. So I totally bow to his much better approach. Simpler and more concise approach, yeah, it's definitely true.

Like the more you get to know handers, the more often you realize is a good way to solve lots of different problems, so definitely check that out when you're building out the smaller models and you're iterating. Do you reuse those models as pre-trained waits for this link larger one, or do you just toss it all away and then retrain from scratch with when I'm kind of like figuring stuff out? As I go like this, I would generally towards tossing away because they're kind of reusing, pre-trained weights introduces complexities that are not really to think about. However, if I'm trying to get to a point where I can run something on really big images, doing these much smaller ones and often I don't reuse those ways, okay, so in this case, what we're doing is we are just joining up all of the classes with A space which gives us a CSV in the normal format and once we've got the CSV in a normal format. It's the usual three lines of code and we train it and we've checked the results. So, there's literally nothing to show you yeah and, as you can see, it's done a great job. The only mistake I think it made was it called this dog for us it should have been dog and so far okay, so Maury class classification is, is pretty straightforward. One minor tweak here is to note that I used a set here because I don't want to list all of the objects.

I only want each object type type here once and so the set plus is a way of deem cute. Locating so that's why I don't have person person, person, person, person just appears so yeah. These sum. These object classification, pre-trained networks. We have a really pretty good at recognizing multiple objects as long as you only have to mention each one once so. That works pretty. Well. All right, so we've got this idea that we've got an input image that goes through a con ComNet. You know which was kind of treated the black box and it spits out a tensor vector of size, 4 plus C. Right C is the number of classes, and so that's what we've got and that gives us a an object detector for a single object. The largest object Americus. So let's now create one which doesn't find a single object, but that finds 16 objects. Okay, so an obvious way to do that would be to take this last. This is just a n n dot, minion right, which has got, however many inputs and 4 plus C outputs. We could take that linear layer and rather than having 4 plus C outputs, we could have 16 times 4 plus C outputs. So it's now spitting out enough things to give us 16 sets of class probabilities and 16 sets of bounding box coordinates, and then we would just need a loss function that will check whether those 16 sets of bounding boxes correctly represented the up to 16 objects that Were represented in the image now, there's a lot of hand waving about the loss function, we're going to it later as to what that is.

But let's pretend we have one: okay, assuming we had a reasonable loss function, that's totally going to work right that that is an architecture which has the necessary output activations. But with the correct loss function, we should be at a trainer to do what we wanted to do. Okay, but that's just one way to do it, there's a second way. We could do it rather than having a n n dot linear what if, instead, we took from our ResNet convolutional background backbone, not in linear, but instead we added a an end, come to D with stride to right, so the final layer of resin app is gets. You a seven by seven by 512 as all right, so this would give us a four by four by whatever number of filters result, maybe for the number of filters that say we picked

256 okay, so it would be four four four by four by 256 has Well, actually,

6. 00:32:50

- **“Perpetual Losses for Real-Time Style Transfer**
- **& Super-Resolution, Mar-2016”**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Look, let's change that, let's not make it four by four by 256; better still, let's do it all in one step, let's make it four by four by four plus C, because now we've got a tensor, where the number of elements is exactly equal to the number Of elements we wanted, so in other words we could we could now. This would well to if we created a loss function that took a four by four by four plus c tensor and mapped it to sixteen objects in the image and checked whether each one was correctly represented by those four plus c activations. That would work like these are two exactly equivalent sets of activations, because they've got the same number of elements they're just reshaped yeah, so it turns out that both of these approaches are actually used. The approach where you basically just spit out one big long vector from fully connected linear layer, is used by a class of models known as yellow. Where else the approach of the convolutional activations is used by models which started with something called SSD single. What I will say is that, since these things came out of very similar times in late 2015, things are very much moved towards here. So the point where this morning, Yolo version 3 came out and is now doing it, the SSD. Ok, so that's what we're going to do right, we're gon na do this and we're gon na learn about why this makes more sense as well, and so the basic idea is this: let's imagine that, on top of underneath this we had another another come to D Stripe 2 and we'd have something which was to buy to buy again less sales for plus C, alright, that's nice and simple, and so basically, it's creating a grid.

That looks something like this one, two, three four okay, so that would be like how the activations are. You know the geometry of the activations of that second extra convolutional straight laughs. Remember it's fair to convolution does the same thing to the geometry of the activations. As a stripe, one convolution below biomass pulling okay. So let's talk about what we might do here, because the basic idea is like we want to kind of say: alright, this top left grid cell is responsible for identifying any object. That's in the top of left this one in the top right is responsible for identifying something in the top right. This one bottom left this one at the bottom right, okay, so in this case you can actually see it started. It said: okay, this one is going to try and find the chair this one, that's actually made a mistake introducing a table, but there are actually one two three chairs here as well right, so basically, each of these grid cells, it's going to be told in the Loss function, your job is to find the object. You know the big object is in that part of the image. So what so? For a multi-label classification, i saw you had a threshold on there, which i guess we're getting your relative. Let's, let's work through it. Okay right, so why do we care about the idea that we would like this convolutional grid cell to be responsible for finding things that were in this part of the image? And the reason is because of something called the receptive field of that car and the basic idea is that, through actual convolutional layers, every every piece of those tensors has a receptive field, which means which part of the input image was responsible for calculating that cell right And, like all things in life, the easiest way to see this is with Microsoft Excel. So do you remember our convolutional neural net and this was emili�s and we had the number 7 and it went through a a two channel filter channel 1 channel 2, which therefore created a 2 channel output.

Okay and then the next layer was another convolution. So this tensor is now a 3d tensor, okay, which then creates saved again to channel output, and then, after that, we had our max pooling layer. Okay. So let's look at this part of this output and the fact this is common, followed by max pool. Let's just pretend as a stripe to come. That's basically the same! So, let's see where this number 27 came from. So if you've got Excel, you can go formulas trace precedents right, and so you can see this came from these 4 okay. Now, where did those 4 come from? Those 4 came from obviously the convolutional filter, Colonel kernels and from these 4 parts of time, one right

7. 00:39:30

- **Implementation notebook w/ re-use of 'bcolz' arrays from Part 1.**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Because we've got four things here, each one of which has a 3 by 3 filter, and so we have 3, 3, 3. 3, the other boy, oh, where did this four come from? Those four came from obviously our filter and this entire part of the input image. Okay and what's more, you can see - and it also comes through this whole duration as well, and you can see that these bits in the middle have lots of weights coming out batgirls. These bits on the outside only have one weight coming out. So we call this here, the receptive field of this activation right, but note that the receptive field is not just saying it's this here box, but also that the center of the box has more dependencies. So this is a critically important concept when it comes to kind of understanding, architectures and understanding why confidence work yeah of the receptive field, and there are some great articles. If you just Google for convolution receptive field, you can find lots of through the articles. I'm sure some of you will write much better ones during the week as well. So that's the basic idea there right is that the receptive field of this convolutional activation is generally centered around this part of the input image. So it should be responsible for finding objects that here, so that's the architecture. The architecture is that we're going to have a resonant backbone, followed by one or more 2d convolutions, and for now we're just going to do one right, which is going to give us a four by four grid. So, let's take a look at that, so here it is, we start with our Lu and drop out.

We then do at the start at the output. Well, I just go through and see what we've got here. This one is not being used. We start with a straight one convolution and the reason we start with a straight one. Convolution is because that doesn't change the geometry at all. It just lets us add an extra layer of calculations right, let's create you know not just a linear layer, but now we have like a little mini neural network in our custom here, all right, so we start with this. Dr1 convolution and standard college is just something i defined up here, which does convolution value vaginal. Like most research code, you see, won't define a class like this. Instead, they'll write the entire thing again and again and again, convolution don't be like that right, like that kind of typical code, leads to errors and leads to poor understanding, and I mentioned that also because this week I released

8. 00:43:00

- **Digress: how "practical" are the tools learnt in Part 2, vs. Part 1 ?**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to

improve - remove this note once proofread)

The first draft of the FASTA, a style guide and the faster I style guide, is very heavily Orion towards the idea of expository programming, which is the idea that programming code should be something that you can use to explain an idea, ideally as readily as mathematical notation To somebody that understands your your coding method, and so the idea actually goes back a very long way, but it was best described in the Turing award like Genesis like the Nobel in computer science, the cheering award lecture of 1979 by probably my greatest computer science hero Can I was he had been working on a since well well before in 1964, but 1964 was the first example of this approach to programming. He released something called APL and then, 25 years later he won the Turing award. He then passed on the baton to his son, Eric Iverson and there's been basically 50 or 60 years now of continuous development of this idea of like what does programming look like when it's designed to to be a notation notation as a tool for thought for expository Programming and so I've made a very shoddy attempt at taking some of these ideas and thinking about how can they be applied to programming, with all the limitations, by comparison that python has anyway so, but you know, here's a very simple example is that if you write All of these things again and again and again, then it really hides the fact that you've got.

You know two convolutional layers, one of this dried one, one of this dried, so my default for standard canvas tried to as a straight one. This is a straight two and then at the end, so this the output of this is going to be four by four okay. I've got a outcomes and an outcome. Vis interesting you can see. It's got two separate convolutional layers, each of which is straight one. So it's not changing the geometry of the input. Okay, one of them is of length of the number of classes. Just ignore K for now K is equal to K is equal to 1 at this point in the code. So it's not doing anything. So what is equal to the length of the number of classes? One is equal to four, and so this is this idea of, rather than having a single comp layer that outputs 4 plus C, let's have two complex, one of which outputs for one of which outputs C and then I will just return them as a list of Two items: that's nearly the same thing: it's nearly the same thing as having a single column player that outputs 4 + C, but let's it lets these layers specialize just a little bit alright. So, like we talked about this idea that when you've got kind of multiple tasks, they can share layers, but they don't have to share all the layers. So in this case our two tasks, which is fine, create a classifier and create down paths. Regression share every single layer is set the very last one okay, and so this is going to spit out two separate tensors and activations one of the classes and one of the coordinates.

Why am i adding one? That's because I'm going to have one more class for background right, so if there aren't actually sixteen objects to detect or if there is an object in this corner represented by this population or grid cell, then I want you to predict background. So that's the entirety. That's the entirety of our architecture. It's incredibly simple right, but the point is now that we, you know we have this convolutional air at the end. One thing I do do is that I at the very end I flatten out the convolution. Basically because I wrote the loss function to expect flattened out tensor, but I could totally try doing that during the week and see which one was easier to understand. Okay, so we've got our data, we've got our architecture, so now all we need is a loss function. Okay, so the loss function needs to look at each of these 16 sets of activations, each of which you're going to have four bounding box, coordinates and C plus one class probabilities and aside, are those activations close or far away from the object, which is kind of Closest to this this this grid cell, in the image and if nothing's there, then you know, are you predicting background correctly, so that turns out to be very hard to do, because let's go back to the 2x2 example to keep it simple, the loss function actually needs To

take each of the objects in the image and match them to one of these convolutional grid cells. To say, like this grid cell is responsible for this particular object.

This grid cell is responsible for this particular object. So then it can go ahead and say like okay, how close are the four coordinates and how close are the master probabilities right? So this is called the matching problem and in order to explain it, I'm going to show it to you. But what I'm going to do first, is I'm going to take a break okay and we're going to come back and understand the maxing map? The matching problem so during the break have a think about how would you design a loss function here? How would you design a function which has a lower value if these 16 times 4 plus K activations, you know somehow better reflect the up to 16 objects which are actually in the ground. Truth image and we'll come back at 7:40. So here's our goal - our dependent variable, basically looks like that, and those are just an extract from our CSV file trapped in dependence and our final convolutional layer is going to be a bunch of numbers which initially is a four by four. I, in this case I think C, is equal to twenty plus we've got one for background right, so four plus 21 equals 26 all right four by four okay and then we we flatten that out into vector we flatten that out into a vector, and so basically Our goal, then, is to say to some particular set of activations that ended up coming out of this model for some let's, let's pick some particular dependent variable.

We need some function that takes in that and that right and where, if it feeds back a higher number, if these activations aren't a good reflection of the ground, truth bounding boxes or a lower number. If it is a good reflection of the ground through the bounding boxes, that's how cool we need to create that much, and so the general approach to creating that function will be to first of all, to simplify it. Down with the two-by-two version will be the first of all. Well, actually I'll show you here's a model I trained earlier. Okay and let's run through I've, taken the loss function and I've split it line by line so that you can see every line that goes into menu. Okay, so so, let's grab our

9. 00:52:10

- **Two approaches to up-sampling: Deconvolution & Resizing**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Validation set data loader grab a batch from it turn them into variables, so we can stick them into a model, put the model in evaluation mode. Stick that data into. We don't stick that data into our model to grab a batch of activations and remember that the final output convolution returned two items that the classes and the bounding boxes. So we can do destructuring assignment to grab the two pieces, the batch of classes and outputs and the batch of bounding box. Okay, and so, as expected, the batch of class outputs is batch, size 64 by 16 grid cells by 21 classes and then 64 by sixteen. By four through the bounding box coordinates: okay, hopefully that all makes sense and after class go back into spec sure. It's not obvious why? These are the shapes, make sure you get to the point where you understand where they are. Let's now go back and look at the ground truth, so the ground truth is is in this Y variable. So let's grab the bounding box part and the plas part and put them into these two Python variables and print them out and so there's our ground truth. Bounding boxes and there's our ground truth classes. So this this image apparently has three objects. You know. So, let's draw a picture of the three objects and there they are okay. We already have a show ground. Truth function, the torch ground, truth function, simply converts the tensors into numpy and passes

them on so that we can print them out. So here we've got.

The bounding box coordinates you'll notice that they've all been scaled to zero to what between 0 and 1 okay. So, basically we're treating the image as being like 1 by 1, so these are all relative to the size of the image. There's our three classes - and so here they are chair - is 0 dining table is 1 and 2 is so far. This is not a model. This is the ground truth. Great here is our 4 by 4 grid cells from our final convolutional letter. So each of these square boxes different papers call them different things. The three terms you're here are anchor boxes prior boxes or default boxes. Okay and through this explanation, you'll get a sense of what they are, but for now think of them as just these 16 squares, I'm going to stick with the term anchor boxes. Ok, these 16 squares on our end devices. So what we're going to do for this loss function is we're going to go through a matching problem where we're going to take every one of these 16 boxes and we're going to see which one of these three round truth objects has the highest amount of overlap With this square, okay, so to do that, we're going to need to know we really have to have some way of measuring amount of overload and there's a standard function for this, which is called the jacquard in this, and the jacquard index is very simple I'll. Do it through example? Let's take this sofa okay.

So if we take this sofa and let's take the jacquard index of this sofa with this grid cell here all right, what we do is we find the area of their intersection. So here is the area of their intersection: okay, and then we find the area of their union. So here is the area of their unit, smart phone, here's, the area of their union, okay and then we say, take the intersection divided by the union. Okay, and so that's jacquard index, also known as iou intersection over all right so of two things overlap by more compared to their total sizes. Together they have a higher jacquard, alright, so we're going to go through and find the jacquard overlap for each one of these. Three objects versus each of these 16 anchor boxes, and so that's going to give us a 3 by 16 matrix, but for every ground truth object and above every anchor box. How much overlap is there? So here are the coordinates of all of our anchor boxes. In this case, they're printed as Center and height width, and so here is the amount of overlap between and, as you can see, it's 3 by 16 right so for each of the three ground truth objects each of these 16 anchor boxes. How much do they overlap? Right, so you can see here: 0. 1. 2. 3. 4. 5. 6. 7. 8. The 8 anchor box overlaps a little bit with the second ground. Truth object, okay, so what we could do now is we could take the max of dimension 1 right. So the max of each row and that will tell us for each ground truth object. What's the maximum amount that overlaps with some grit zone - and it also tells us - remember pytorch when you say Mets, two things, it says what is the max and what is the index both amass? So, for each of these things, the 14th grid cell is the largest is the largest overlap for the first round truth: thirteen for the second and 11 okay.

So that tells us. You know a pretty good way of assigning each of these ground. Truth objects to a grid cell what what the matches is, which one is the highest overlap, but we're going to do a second thing. We're also going to look at max over dimensions, zero and Max over dimension. Zero is going to tell us what's the maximum amount of overlap for each grid cell, so across all of the ground. Truth objects right, and so particularly interesting here - tells us for every grid cell of sixteen. What's the index of the ground truth object which overlaps with it? The most zero is a bit overloaded here, zero could either mean the amount of overlap was zero or it could mean its largest overlap, is with object, index zero. It's going to turn out not to matter so there's a function called map to ground truth which I'm not going to worry about for now. It's it's super simple code, but it's slightly awkward to to think about, but basically what it does is it combines these two sets of overlaps in a way described in the SSD paper, to assign every anchor box to a ground truth object and basically the way of The signs that is

each of these ones, each of these three gets assigned in this way right. So this one, this object is assigned to bound to anchor box 14 this one to 13 and this one through 11 and then of the rest of the anchor boxes.

They get assigned to anything which they have an overlap of at least point 5 with, if anything that doesn't, which isn't in either of those criteria, ie either isn't a maximum or doesn't have a greater than 0.5 overlap is considered to be a cell which contains background. Okay, so that's all the map to ground truth motion does, and so after we go through it, you can see now a list of all of the assignments, and you can also see anywhere that there's a zero here. It means it was assigned a background, in fact anywhere. It's less than point five here, give us a sign in the background, so you can see those three which kind of forced assignments that puts a high number in just to make sure all of their assigned all right. So we can now go ahead and convert those two classes, and then we can make sure we just grab those which are at least point five in size, and so finally, that allows us to spit out the three pluses that are being predicted. We can then put that back into the bounding boxes, and so here are what each of those bounding boxes is sorry, what each of those anchor boxes is meant to be predicting okay, so you can see sofa dining room table chair, which makes perfect sense. If we go back to here, this is meant to be predicting so far. This is over. This is meant to be predicting dining room table. This has been to be predicting chair and everything else has been to be predicting background. So that's the matching stage.

So once we've done the matching stage, we're basically done we can take the activations just grab those which which matched that's. What this positive indexes are subtract from those the ground. Truth bounding boxes just for those which matched positive ones, take the absolute value of the difference. Take the mean of that and that's not one loss and then for the classifications. We can just do a cross entropy and then before we can add them together. Okay, so that's the basic idea, there's a few, and so this is. This is what's going to happen right, we're going to end up with 16 recommended, you know predicted bounding boxes coming out. Most of them will be background, see all these ones that say BG, but from time to time. They'll say this is a cow. This is potted plant. This is par okay, if you're wondering like what does it predict in terms of the bounding box of background, the answer is a totally ignores it right. That's why we had this. Only positive indexes thing here right. So if it's background, there's no, you know sense of like where's the correct bounding box in background. That's totally meaningless, so the only ones where the bounding box makes sense out of all these are the ones that there are some important literal tweets. What is that the? How do we interpret the activations, and so the way we interpret the activations is defined here in activation, two bounding box, and so basically we grab the activations.

We stick them throughthe an and so remember. Fan is the same as sigmoid shape, except it's scaled to be between negative 1 and 1, but read zero okay. So it's basically a sigmoid function that goes between negative one and one, and so that forces it to be within that range and we then say: okay, let's grab the the actual position of the anchor boxes and we will move them around according to the value of The activations divided by two, so in other words each each each activate each edge, as predicted bounding box, can be moved by up to fifty percent of a grid size from where its default position is and ditto for its height and width. It can be up to twice as being all half as big as its default size. So so that's one thing is we have to convert the activations into some kind of way of scaling those default length of box positions. Another thing is, we don't actually use cross. Entropy we actually use binary cross entropy loss. Okay, so remember binary. Cross entropy loss is what we normally use for. Multi-Label classification like in the planet, amazon, satellite competition, each satellite image - you could

have multiple things in it: okay, so if it's got multiple things in it, you can't use soft max because soft max kind of really encourages just one thing to have in our case, each Anchor box can only have one object associated with it, so it's it's not for that reason that we're avoiding song max it's something else, which is it's possible for an anchor box to have nothing associated with it.

So there'd be two ways to handle. That is this either at background. One would be to say you know what backgrounds just a class right. So let's use soft max right and just treat background as one of the classes that the soft max could could predict a lot of people have done it. This way, I don't like that, though, right, because that's a really hard thing to ask of your network, do it's basically to say: can you tell whether this grid cell doesn't have any of the 20 objects that I'm interested with a jacquard overlap of more than 0.5? Now that's a really hard thing to put into a single computation. On the other hand, what if we just had for each class, you know, is it a motorbike? Is it a bus? Is it a person, it's the bird? Is it a dining room table right and then it can check each of those would be no? No, no, no and it snowed all of them, and it's like. Oh it's background, all right, so that's that's the way. I'm doing it is it's not that we could have multiple, true labels, but we can have 0, and so that's what's going on here we take our target and we do a one hot, embedding with number of classes plus 1. So this stage we do have the idea of background, but then we remove the last problem. So the Batman columns now gone right, and so now this vectors either of all zeros. Basically meaning there's nothing here or it has at most one one. And so then we can use binary cross-entropy predictions with that.

That is a minor tweak right, but like it's the kind of minor tweak that I I want you to think about and understand, because it's a really like it makes a. It makes a really big difference in practice to your training and it's the kind of thing that you'll see a lot of papers talk about like often when there's some increment over some previous paper it'll. Be something like this that we somebody to realize this, like Oh trying to predict a background category using a soft mass, is really hard to do what if we use the binary cross-entropy instead, you know - and so it's kind of like if you understand what this is Doing and more importantly, why yeah that's a really good test of your understanding of the material, and, if you don't that's fine Brad just shows you. This is something that you need to. Let me go back and re-watch this part of the video and talk to some of your classmates and, if necessary, ask for the forum and sure you understand what are we doing? Okay? So that's what this that's!

10. 01:09:30

- **TQDM library: add a progress meter to your loops**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

What this binary cross, entropy loss loss function is doing so, basically, in this part of the code, we've got this custom loss function. We've got the thing that calculates the declared index. We've got the thing that converts activations to bounding balls. We've got the thing that does map to ground truth. Okay and that's it all that's left is the SSD loss function, so the SSD loss function. This is actually what we set yeah as they are crit, as our criterion is SSD, so what SSD lost class? Is it it loops through each image in the minivan and it calls SSD one loss, so Assisting loss for one image, so this function is really where it's all happening. This is calculating the SSD loss for one image, so we D structure our bounding box in class and

basically, there's a what is this doing here? Actually, this is worth mentioning a lot of code you find out there on the Internet doesn't work with mini-batches. You know it only does like one thing at a time. So in this case we you know all this stuff is working. It's not exactly a mini batch at a time. It's on a whole bunch of ground, truth objects at a time, and the data loader is being fed a mini batch at a time to do or the convolutional layers, because we could have different numbers of ground. Truth objects in each image, but a tensor has to be the strict rectangular shape fastai automatically with zeros anything. That's not the same life thing I fairly recently added, but it's super handy.

Almost no other libraries do that, but that does mean that you then have to make sure that you get rid of those zeros right. So you can see here I'm checking to find all of the all of the non zeros and I'm only keeping those. This is just getting rid of any of the bounding boxes that are actually I'm just padding, yeah, okay, so get rid of the padding turn the activations bounding boxes do the jacquard doing about this is all the stuff we just went through it's on line by line Underneath right check that there's an overlap greater than something around 0.4 0.5 different papers use different values. For this find the things that match, but the class put the background class for those and then finally get the l1 loss for the localization part. Get the binary. Cross-Entropy boss, for the classification part, turn those two pieces and then finally add them together. So that's a lot going on and it might take a few watches of the video to in the code to fully understand it. But the basic idea now is that we now have the things we need. We have the data, we have the architecture and we have the loss function. So now we've got those three things. We can train my finder and train for a bit and we get down to 25 and the end. We can see how we went so. Obviously, this isn't quite what we want when we do practice we'd kind of remove the background ones or some threshold, but it's like it's on the right track.

There's a dog in the middle: let's go to 0.34 there's a bird here in the middle of 0.94. You know something's working, okay, yeah, I've got a few concerns. I don't think it's. I don't see anything saying motorcycle here. It says bicycle, which is in great there's nothing for the potted plant. That's big enough, but that's not surprising, because all of our anchor boxes were small. They were 4x4 grid so to go from here to something that's going to be more accurate. What we're gon na do is to create way more and the buses okay. So there's a couple of ways: we can create quick question and I'm just getting lost in the fact that the anchor boxes in the bounding boxes are. How are they not the same? I must be missing something. Anchor boxes are the square, the fixed square grid cells. These are the anchor boxes they're in an exact, specific unmoving location. The bounding boxes are, these are three things: the bounding boxes, these 16 things at anchor boxes; okay, so we're going to create lots more anchor bosses. So there's three ways to do that and I've kind of drawn some of them printed. Some of them here one is to create anchor boxes of different sizes and orientations. So here you can see you know, there's a upright rectangle, there's a line down rectangle and there's a square. It's a question for the multi-label classification.

Why aren't we multiplying the categorical loss by a constant like we did before? That's a great question because later on it'll turn out, we don't need to so yeah. So you can see here like this square, and so I don't know if you can see this blue shield up. You basically got one two: three squares of different sizes and for each of those three squares. You've also got a line down rectangle and up rectangle. That's we've got three aspect ratios at three zoom levels. That's why don't you do? We can do this and this is for the one by one grid. So, in other words, if we added two more strata, convolutional layers, you eventually get to a one by one grid, and so this is for the one by one include. Another thing we could do is to use more

compositional layers as sources of anchor boxes so as well as our and I've I've randomly gated these a little bit. So it's easy to see right so, as well as our 16 by 16 grid cells. These cells we've also got two by two grid cells, and we've also got the one by one itself, alright. So, in other words, if we add three straight hours prior to convolutions to the to the end, we'll have 4 by 4, 2 by 2 and 1 by 1 sets of grid cells, all of which have anchor boxes and then for every one of those we can Have all of these different shapes and sizes? Okay, so obviously those two combined with each other to create lots of anchor boxes and if I try to print that on the screen is just one big toss. So that's what this code is right.

It says all right. What are all the grid cell sizes? I have for the anchor boxes, what are all the zoom levels I have for the anchor boxes and what are all the

11. 01:17:30

- **Fast Style Transfer w/ "Supplementary Material, Mar-2016"**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Aspect ratios, I have for the anchor boxes and the rest of this code then just goes away and creates the top left and bottom right corners inside anchor corner and the middle and height width in mecca's. So that's all this does and you can go through it and print out the anchor in a corner. So the key. The key is to remember this basic idea that we have a vector of ground truth, stuff right where that stuff is like sets of four bounding boxes. This is what we were given. It was in the JSON files that it's the ground truth. It's the dependent variable sets of four bounding boxes and for each one, also a class right. So this is a person in this location. This is a dog in this location. That's the ground! Truth that we've given yes, yeah exactly left X, Y bottom right, X Y. So that's what we printed here right, we print it out. This is what we call the ground truth, there's no model! This is what we're told is what we this is, what the answer is meant to be, and so remember any time we train an ear on that we have a dependent variable and then we have a neuro nap from blackbox neural net. That takes some input and spits out some output activations and we take those activations and we compare them to the ground truth. We calculate a loss. We find the derivative of that and adjust the weights according to the derivative times a learning rate. Okay, so the loss is calculated using a must function. Something I wanted to say is, I think them.

One of the challenges with this problem is part of. What's going on here is we're having to come up with an architecture. That's letting us predict this ground truth like it's, not because you can have you know any number of objects in your picture. It's not an you know immediately obvious, like oh, what's the correct architecture, that's gon na, let us predict that sort of ground truth. That's so I'm gon na kind of make this plain, as we saw when we looked at the kind of Yolo versus SSD, that, like there are only two possible architectures, the last layer is fully connected or the last layer is convolutional and both of them work perfectly. Well, I'm sorry, I meant, in terms of by creating this idea of anchor boxes and anchor boxes with different locations, locations and sizes. That's giving you a format that kind of lets you get to the activations you're right like high level. Is that you see okay? So that's that's really entirely in the loss function, not in the architecture like and if we use the Yolo architecture where we had a fully connected layer like literally there would be no concept of geometry at all right, so I would suggest, like kind of forgetting the Architecture and just like treated as just a given it's a theme that is spitting out 16 times 4 plus C activations right and then I would

say our job is to figure out how to take those 16 times, 4 plus C activations and compare them to our Ground truth, which is like 4 plus it's 4 plus 1, but if it was one hot encoded, it would be C and I think that's easier to think about so call it 4 plus C times.

However, many ground truth objects. There are for that particular image. All right, so, let's pour that right, so we need a loss function that can take these two things and spit out a number that says how good are these activations that that's that's what we're trying to do so to do it. We need to take each one of these M ground. Truth objects and decide which set of 4 plus C activations is responsible for that object, which one should we could be comparing and saying like. Yes, the right class will not and yeah it's false or not, and so the way we do, that is basically to say. Okay, let's decide the first for the first 4 plus C activations are going to be responsible for predicting the bounding box of the thing. That's closest to the top left and the last 4 plus C you'll be predicting those the furthest to the bottom right and kind of everything in between. So this is this matching and then, of course, we're not using the yellow approach where we have a single vector. We're using the SSD approach, where we spit out a convolutional output, which means that it's it's not arbitrary as to which we mash up, but actually we want to match up the set of activations, whose receptive field most closely reflects. You know as the maximum density from where this real object is, but that's a that's a minor tweak. You know, I guess like that.

It that's the easy way to have taught this, but if we just start with the Yolo approach, where it's just like an arbitrary vector and we can decide which activations correspond to which agree on trees object as long as it's consistent, it's going to be a consistent Rule because like if, in the first image, the top left object, corresponds with the first four plus C activations and then the second image we threw things around and suddenly it's now going with the last four plus the activations. The neural net doesn't know what to learn about that. The neural net needs to like a loss function needs to be like some consistent task right, which in this case the consistent task is try to make these activations reflect the bounding box in this general area. That's basically what this loss function is trying to do. Is it purely coincident that you know the 4x4 in the kong-kong 2d is the same thing as no you're 16. Not at all coincidence. It's it's because, though, that 4x4 comp is going to give us activations whose receptive field corresponds to those locations in the infinite. So it's it's carefully designed now. Remember I told you before part two that, like the stuff we learn in part, two is going to assume that you are extremely comfortable with everything you learnt in part one and for a lot of you, you might be realizing now.

Maybe I wasn't quite as familiar with the starting part, one as I first thought and that's fun right, but just realize you might just have to go back and really think deeply. It's there more with understanding with life. What are the inputs and outputs to each layer and a convolutional Network? How big are they one of their rank? Is that V? How are they calculated so that you really fully understand the idea receptive field? What's a loss function really, how does back propagation work exactly like these things all need to be like deeply felt intuitions, but you only get through to practice and once they're all deeply felt intuitions, then you can we watch this video and you'll be like? Oh, I see okay, see that you know these activations just need some way of understanding what task they're being given, that is being done by the loss function and the loss function is encoding a task, and so the task of the SSD loss function is basically two Parts part one is figure out which ground truth object is closest to which grid cell, which anchor boss. When we, where we started doing this, the grid cells of the convolution and the anchor boxes were the same right. But now we're starting to introduce the idea that we can have multiple anchor boxes per grid cell okay, a little bit more

complicated, so every ground truth object. We have to figure out which anchor boxes are closest to every anchor box.

We have to decide which ground truth object isn't responsible for if any and once we've done that matching it's trivial now we just basically go through and going back to the single object detection. It now is just this: it's once

12. 01:27:45

- **Ugly artifacts like “checkerboard”: cause and fixes; Keras UpSampling2D**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

We've got every ground truth object mesh to anchor box to a set of activations. We can basically didn't say: okay, what's the cross, entropy loss of the categorical part, what's the r-1 loss? So really it's the matching part which is kind of I don't know kind of slightly surprising bit and then this idea of picking those in a way that the convolutional Network gives it the best opportunity to calculate that part of the space. Is then the final cherry on top and this um I'll tell you something else this class is by is by far, I think, going to be the most conceptually challenging and part of the reason for that is that, after this we're going to go and do some Different stuff and we're going to come back to it in lesson 14, and do it again with some tweaks, alright, we're gon na add in some of the new stuff we learned afterwards, so you're gon na get like a whole second round for this material. Once we add some some extra stuff at the end, so we kind of dinner revise it, as we normally do. Remember. In part 1, we kind of went through computer vision and RP structured data back to NLP back to computer vision. You know so we revised everything from this at the end. It'll be kind of similar yeah, so yeah, so don't worry if it's a bit challenging in fist you'll get there okay, so so for every grid cell there can be different sizes.

We can have different orientations and zooms representing different different anchor boxes, which are just like conceptual ideas that basically every one of these is associated with one set of four plus e activations in a model right. So, however, many of these ground truth boxes, we have. We need to have that x, 4 plus C activations in the model. Now that does not mean that each convolutional layer needs that many filters right because remember the 4x4 convolutional layer already has 16 sets of filters. The 2x2 accomplished layer already has four sets activations and then finally, the one by one has one set, so we basically get 1 plus 4 plus 16 for free. Just because that's how convolution works it that relates things at different locations. So we actually only need to know K where K is the number of zooms by the number of aspect ratios where else the grids we're gon na get for free through our architecture. So, let's check out that architecture, so the model is nearly identical to what we had before all right that we're gon na go we're going to have a number of strive.

13. 01:31:20

- **ImageNet Processing in parallel**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

To convolutions, which is going to take us through to 4 by 4, 2 by 2, 1 by 1, you straight to convolution, perhaps our grid sizes, okay and then after we do our first convolution to get to 4 by 4, we're going to grab a set of Outputs from that, because we want to save away the 4 by 4, which anchors and then once we get to 2 by 2. We grab another set now two by two anchors and then finally, we get to 1 by 1 and we saw we get another set of the Ovilus right. So you can see. We've got like a whole bunch of these outcome. This first one we actually not easy. So at the end of that we can then concatenate torque at concatenate them all together. So we've got the four by four activations, the two by two activations, the one by one and visions. So that's going to give us the correct number of activations to give us one activation for every for every bounding, a periphery anchor boss. So then we just set our criterion as before to SSD loss and we go ahead and train right and the way we go so in this case, I'm just printing out those things with at least probability of point one, and you can see, we've got some things Booked: okay, some things: don't our big

14. 01:33:15

- **DeVISE research paper**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Objects like bird we've got a box here with a point: nine three probability: it's looking to be in about the rest, but our person is looking pretty hopeful, but our motorbike has nothing at all a lot of plants looking pretty horrible. That bus is all the wrong size. What's going on so, but what's going on here, will tell us a lot about the kind of history of object, detection, and so these five papers are key steps in the history recent modern history of our vegetation, and so they go back to a balance of maybe 2013, this paper called scalable object, detection using deep neural networks. This is what basically set everything up and when people refer to the multi box method, they're talking about this paper - and this was the basic one that came up with this idea - that we can have a loss function that has this matching process, and then you can Kind of use that to do object addition, so everything since that time has been trying to figure out basically how to make this better. So, in parallel, as a guy quarter, Oscar shik, who was going down a totally different direction, which was he had these two-stage processes? Where the first stage used like classical computer vision, approaches to like fine kind of edges and changes of gradients and stuff to kind of guess which parts of the image may represent us objects and then fit each of those into a convolutional neural network which was basically Designed to figure out is that actually the kind of object I'm interested in, and so this was the kind of our CNN and fast.

I see a hybrid of traditional computer vision and so what Russ and his team then did was they basically took this multi box. Ide the air and replaced the traditional non deep learning computer vision, part of their two-stage process with a confident, so they now have two conquers one comp net that basically sped out something like this, which we call these region proposals. You know all of the things that might be objects and then the second part was the same as his earlier work, because, basically something the top each of those fitted into a separate part net, which was designed to classify whether or not. That particular thing really isn't. Interesting at a similar time, these two papers came out, Yolo and SSD. Now both of these did something pretty cool, which is they got. The same kind of performance is faster, a CNN, but with once okay, and so they basically took the multi boss idea and they tried to figure out how to deal with this mess. It's done and the basic ideas were to use, for example,

called hard- mining where they would like go through and find all of the matches that didn't. Look that good and further away has some very tricky and complex data. Augmentation methods all kinds of hackery basically, but they got it to work pretty pretty well, but then, through really cool happened. Late last year, which is this thing called focal versus paper philosophy, dense object attention. The network object is called written written where they actually realized.

Why? This messy crap wasn't working and I'll describe why dismissive fat loss by trying to describe why it is that we can't find the motive a so here's the thing when we look at this. We have three different granularities of convolutional route, four by four two by two one by one, the one by one, it's quite likely to have a reasonable overlap with some object, because most photos have sometimes main so. Okay, on the other hand, in the four

15. 01:38:00

- **Digress: Tips on path setup for SSD vs. HD**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

By four those sixteen grid cells unlikely, most of them are not going to have a much of an overlap with anything like in this motor pipe cases guys. So, if somebody was to say to you like, you know, 20-buck bet. What do you reckon this little quick? Is you know and you're not sure you're gon na say background because most of the time in this background right and so here's the thing? Okay, I understand why we have a four by four grid of receptive fields with one anchor box each to coarsely. Localize objects in the image, but I think I'm missing is why we need multiple receptive fields at different sizes. The first version already included 16 receptive fields, each with a single anchor box associated with the addition. There are now many more anger boxes to consider. Is this because you constrain how much a receptive field could move or scale from its original size, or is there another reason it's kind of backwards? The reason I did the constraining was because I knew I was going to be adding more boxes later, but really the reason is that the jacquard overlap between one of those 4x4 grid cells, and you know, a picture, a single object that takes out most of the Image is never going to be 0.5 because, like the intersections much smaller than the Union, because the object is too big.

So for this general idea to work where we're saying, like you're responsible for something that you've got a better than 50 % overlap with, we need anchor boxes which, which will, on a regular basis, have a 50 % or higher overlap. Which means we need to have a variety of sizes and shapes and scales yeah. So this is this this this all happens. This all happens in the last match. Basically, the vast majority of the interesting stuff in all of the object detection stuff is the loss function, because there is only three things last function, architecture, so the this is the focal most paper for kalasa dense, object, detection from August 2017, his Ross Perik still doing This stuff coming her, you might recognize as being the the rezneck guy a bit of an all-star cast here and this. The key thing is this very first picture. The blue line is a picture of binary cross-entropy loss. The x-axis is, what is the probability or what is the activation? What is the probability of the ground truth class? So it's actually a motorbike. I said with point six chance: it's a motorbike or it's actually not a motorbike, and I said with plot point six chance. So this blue line represents the level of the value of cross-entropy loss, so you can draw this in Excel or Python or whatever a simple plot of cross-entropy loss. So the point is: if the answer is because remember, we do in binary custom to be lost. If the answer is not a

motorbike - and I said yeah, I think it's not a motorbike.

I'm point six sure it's not a

16. 01:42:00

- `words, vectors = zip(*w2v_list)`

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Motorbike this blue line is still it like a loss at about 0.5. It's it's it's! It's there's a lot of. It's still pretty bad right, so I actually have to keep getting more and more confident that it's not a motorbike. So if I want to get my loss down, then for all of these things, which are actually background, I have to be saying like I am sure that background you know or I'm sure it's not a motorbike or a bus or a person or a diagram there, Because if I don't say I'm sure it's not any of these things, then I still get plus. So that's why this doesn't work all right. This doesn't work because even when it gets to here and it wants to say, I think it's a motorbike - there's no payoff for it to say so, because if it's wrong right and it kick it, it gets killed and the vast majority of the time. It's not anything the vast majority of time its background, and even if it's not background, it's not enough just to say it's not background you're gon na say which of the 20 things. It is right. So for the really big things. It's fine because that's the one by one grid, you know so it's it generally is a thing, and you just have to figure out which thing it is where else for these small ones and generally it's not anything so generally small ones. We just prefer to be like I've got nothing to say no comment.

Okay, so that's why this is empty and that's why, even when we do have a bus right, it's using a really big grid cell to say it's a bus, because these are the only ones where it's like confident enough to make a call that something right, because The small grid cells it very rarely is something so the trick is to try and find a different loss function instead of binary cross, which we lost. It doesn't look like the blue line, but it looks more like the green or purple line and they actually end up suggesting the purple line, and so it turns out. This is cross-entropy loss. Negative log PT focal loss is simply 1 minus PT to the gamma, where gamma is some parameter right and they recommend using 2 times the cross entropy loss. That's it's literally just a scaling, and so that takes you to if you use camera, that's true that takes you to this purple line. So now, if we say yeah, I'm point 6 sure that it's not a motorbike than the loss function is way good for you. No worries, ok, so that's what we want to do. We want to replace cross entropy loss with vocalist, and I mentioned a couple of things about this fantastic paper. The first is like the actually contour. The actual contribution of this paper is to add Y minus P. To the gamma to the start of this equation, which sounds like nothing right, but actually people have been trying to figure out this down problem for years that and I'm not even sure that realize it's a problem.

There's just this assumption that you know object. Detection is really hard and you have to do all of these complex data, augmentations and have- mining and blah blah blah to get the damn thing to work. So a it's like this recognition of like. But why are we doing all those things? And then this realization is I'm like. Oh, if I do that it goes away, it's fixed alright. So when you come across a paper like this, which is like game-changing, you shouldn't assume you're gon na have to write 100 thousand lines of code. Very often is one line of code or the change of a single, constant or adding log to a single class. So, let's go down to the bit where it all

happens where they describe personal loss, and I just wanted to point out a couple of terrific things about this paper. The first is here is their definition of cross-entropy and if you're not able to write cross-entropy on a piece of paper right now, then you need to go back and study it, because we're going to be assuming that you know what it is, what it means why It's that what the shape of it looks like cross-entropy appears everywhere: binary, cross-entropy and kind of variable, cross, entropy and the softmax that most people, most of the time, we'll see cross entropy written as like an indicator on y Times, log K plus an indicator on Y Minus y times y minus PE, this is like kind of awkward notation, often people. We use that Dirac Delta functions like that, or else this this paper just says.

You know what it's just a conditional. The cross, yet repeat, simply is a lot negative luck. P of Y is 1 negative, 1 1 minus P of us, so this is y is 1. If it's a motorbike 0 or not in this paper, they say what, if it's about by 4 negative 1, and then they do something which mathematicians never do they refactor? All right check us out hey what, if we replace what, if we define a new term called PT, which is equal to the probability if Y is 1 or 1 minus P. Otherwise, if we did that, we could now redefined seee as that we're just super cool. Like it's such an obvious thing to do, but as soon as you do it all of the other equations get simpler as well because later on straight at the variants paragraph, they say, hey one way to deal with class in balance. Ie lots of stuff is background. Would just be to have a different weighting factor, the background. This is not so like for class one. You know we'll have some number alpha. + 4 + 0 will have 1 minus alpha, but then they're like hey, let's define alpha to you the same way and so now our cross entropy. Well, you know with a weighting factor like this, and so then they can wrap their focal loss with the same concept and then eventually they say, hey, let's take focal loss and combine it with class waiting like so yeah. So often when you see in a paper huge big equations, it's just because mathematicians don't know how to be back, though, and you'll see like the same pieces, are kind of repeated all over the place right very, very, very often, and by the time you've turned it Into non pile code suddenly is super simple, so this is a million times better than so great paper to read to understand how papers should be a terrible paper to read to understand what most his long rap okay.

So, let's try this we're going to use this yeah now remember: negative log P is

17. 01:49:30

- **Resize images**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

The cross entropy loss. So therefore, this is just equal to some number times the cross entropy loss and when I defined the binomial cross, entropy loss. If you remember, or if you noticed, I had a weight which by default, was none right and when you call binary cross entropy Swaggart's pipe torch thing, you can optionally pass in the web. This is nothing that's more applied by everything and if it's none, then there's no work so since we're just going to multiply across entropy by something we can just define get away so here's the entirety. This is the thing that, like suddenly made object, detection business. Okay, so this was late last year. Suddenly it got rid of all of the complex messy pattern right and so sigmoid is that PT is W, and here you can see 1 minus P, 2 e to the power of gamma right. So we're gon na hammer of to our first point: 2. 5. If you're wondering why this paper, because they tried lots of different values of gamma and alpha, and they found that 2 and 0.25 work well. Okay, so

there's our new loss function. It derives from our BC loss, adding a weight to it, focal loss other than that there's nothing else to do. We could just train our model again, okay, and so this time things are looking quite a bit better. You know we now have motor bike. Bicycle person motorbike like it's, it's actually having a go at finding something yeah, it's still doing a good job with big ones. In fact, it's looking quite a lot better.

It's finding quite a few people. It's finding a couple of different birds. It's looking pretty good right! So our last step is for now is to basically figure out how to pull out just the interesting stuff like. Let's take this barking, this sofa right. How do we pick out a dog and a sofa, and the answer is

18. 01:52:15

- **Three ways to make an algorithm faster:**
- **memory locality,**
- **simd/vectorization,**
- **parallel processing**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Incredibly simple: all we're going to do is we're going to be going to go through every pair of these bounding boxes and if they overlap by more than some amount say 0.5 using jacquard, and they both predict in the same class. We're going to assume that the same thing and we're just got to pick the one with the higher p-value, and we just keep doing that repeatedly. That's really boring code. I actually didn't write it myself. I copied it on somebody else's code, non maximum suppression. Yes, no reason particularly to go through it, but that's one of us okay, so we can now show the results of the non maximum suppression and yeah. Here's the sofa, here's, the dog yeah. Yes, the bird is the person this person cigarette looks like it's firework. This one it's like it's okay, but not great, like it's found a person and his bicycle of a person and his bicycle with his bicycle in the wrong place place. You know you can also see that, like some of these smaller things that lower p-values the top button, where the back of just point one sticks. This is same time of bus. So there's something still two things here right and the trick will be to use something called featured here and that's what we're going to do in less than 14. What I wanted to do in the last few minutes of class was to talk a little bit more about the papers and specifically to go back to the SSD paper.

Okay, so this is single-shot multipass detector, and when this came out I was very excited because it was kind of you know it and Yolo were like. You know. The first kind of single pass good quality object, detection methods that come along, and so I kind of ignored, object, detection until this time or this to pass stuff with and then fast, our CNN and faster us. You know, because there's been as kind of continuous repetition of history and the deep learning world, which is things that involve multiple passes of multiple different pieces over time. You know, particularly where they involve some long deep learning pieces like over time. They basically always get turned into a single end-to-end deep learning model, so I tend to kind of like ignore them until that happens, because that's the point where it's like okay, now, people have figured out how to show this as a deep learning problem as soon as People do that they generally end up something it's much faster and much more accurate and so SSD and Yolo were really important. So here's the SSD paper, let's go down to the key piece, which is where they describe the model and let's try and understand it. So the model is basically one two three

four paragraphs right, so tape is a really concise mat, which means that you kind of need to read them pretty carefully. Partly though you need to know which bits to read carefully so the bits where they say here, we're going to prove the error bounds on this model, you could ignore that right because you don't care about proving the intervals. But the bit which says here is what the model is careful, so here's the bit called model, and so hopefully you'll find.

We can now read this together and understand it. So SSD is a feed-forward content and it creates a fixed size, collection of bounding boxes and scores for the presence of object class instances in those boxes so fixed size that ie the the convolutional great time. Okay, you know different aspect ratios and stuff and each one of those has 4 plus C activations, followed by a non maximum suppression step to take that massive gump and turn it into. You know just a couple of novel adding different objects. The early layer is based on the standard architecture, so we just use rest man. This is pretty standard. As you know, you can kind of see this consistent theme, particularly in kind of how the pasta you know library tries to do things which is like grab a pre trade network. That already does something pull off the envious, decant, a new enemy, all right, so early network players. We use the standard, classifier truncate, the classification layers, as we've always do that automatically when we use Carmona - and we call this the base Network. Some campers call that the backbone and we then add an auxilary structure. Okay, so the auxiliary structure, which we call the custom head, has multi scale feature nests. So we add compositional layers to the end of this base Network and they decrease in size progressively. So a bunch of side to convice, so that allows predictions of detections and multiple scales. The grid cells are different size. It is.

The model is different for each feature: layer compared to Yolo that operate on a single feature, so you're low, as we discussed dispenser, is one vector for us. We have different harmless H, added feature layer gives you a fixed set of predictions using a bunch of filters for a filter layer where the grid size is n by n 4x4 with pay channels where, in fact, let's take the pick as well. Seven by seven with 512 panels, the basic element is going to be a three by three by P cone, which, in our case is a three by three by four for the shape offset bit or three by three by C for the score for category vision. That's so those are those three. Those are those two pieces at each of those grid cell locations. It's going to produce an output value and the bounding box offsets measured relative to that default. Cost position which we've been calling an anchor box position relative to the feature, mount location. What we've been calling the grid cell? Okay as opposed to Yolo right, which has a fully connected layer? And then they go on to describe the default boxes, what they are for? Each feature now cell organs in grid cell, they Tyrell the picture map in a convolutional manner, so the position of each box relative to its grid cell is best. So hopefully you can see you know we end up with C plus 4 times K filters.

If there are K boxes at each location, okay, so these are similar to the anchor bosses so like. If you jump straight in and read a paper like this, without knowing like only solving and Maya solving, yet when what's the kind of man acted shirt circle, those four paragraphs would probably make almost no sense, but now that we've gone through it, you read those four Paragraphs and hopefully you're thinking - oh that's just what Jeremy said only they said it better than Jerry and less words. Okay. So so I have the same problem when I started reading the SST paper and I read those four paragraphs - and I didn't have before this time much of a background in object notation, because I decided to wait until she passed me more, and so I read this And I was like what the hell right and so the trick is to then start reading back over the citation right. So, for example - and you should go back and read this paper now look here's the matching strategy but and that all matching strategy that I somehow spent like it

now we're talking about. That's just a paragraph, but it really is right for each ground truth. We select from default boxes based on location aspect, ratio and scale. We match each ground truth to the default box, with the best jacquard overlap, and then we met two default boxes, anything with jacquard over that I own printer. That's it that's the one sentence version and then we've got the loss function, which is basically to say, take the average of the last based on classes, plus the lost based on localization, with some weighting factor now with focal loss I found.

I didn't really need the weighting factor anymore, they both, but in this case, as I started reading this, I didn't really understand exactly what LMG and all this stuff was, but it says well, this is derived from the more t-bar subjective. So then, I went back to the paper that defined Modi bas and I found in their proposed approach. They've also got a section called training objective, also known as loss function, and here I can see it's the same. Notation l, GE, and so this is where I can go back and see the detail and after you read a bunch of papers, you'll start to see things very quickly. For example, when you see these double bars, you'll realize every time there's mean squared error. That's how you're right mean square error right. This is actually called the two norm. The two long is just the sum of squared differences right and then there's two up here means normally they take the square root. So we just don't do this, so this is just M SC. All right anytime! You see like. Oh, yes, a lot of C and his see. You know that's basically, a binary cross-entropy right, so it's like you, you're, not actually gon na, have to read every day at every equation. Right. You are kind of a bit at first right, but after a while, your brain just like immediately knows basically what's going on, and then I say: oh I've got a Luxio panel of 1 minus C and as expected I should have my X and here's. My 1 minus X, okay, there's all the pieces there that I would expect to see.

So then, having done that, that then kind of allowed me okay and then they get combined with the two pieces and oh there's the multiplier that I expected, and so now I can kind of come back here understand what's going on, okay, so we're going to be Looking at a lot more papers right, but maybe this week go through the code and go through the paper all right and be like. What's what's going on and remember what I did to make it easier for you was I took that loss function. I copied it into a cell and it has lit it up so that each bit was in a separate cell and then, after every cell I other printed or plotted that value that. So, if I hadn't have done that for you, you should do it yourself right. Like if there's no way, you can understand these functions without trying putting things in single time, yeah, okay, so hopefully this is kind of a good good start. Okay! Well thanks! Everybody have a great week, see you next Monday,

Lesson 10: NLP Classification and Translation

Outline

After reviewing what we've learned about object detection, today we jump into NLP, starting with an introduction to the new fastai.text library. This is a replacement for torchtext which is faster and more flexible in many situations. A lot of today's class will be very familiar—we're covering a lot of the same ground as lesson 4. But today's lesson will show you how to get much more accurate results, by using transfer learning for NLP.

Transfer learning has revolutionized computer vision, but until now it largely has failed to make much of an impact in NLP (and to some extent has been simply ignored). In this class we'll show how pre-training a full language model can greatly surpass previous approaches based on simple word vectors. We'll use this language model to show a new state of the art result in text classification.

Video Timelines and Transcript

1. 00:00:10

- **Picking an optimizer for Style Transfer (student post on Medium)**
- **Plus other student posts and tips on class project.**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

So welcome to lesson 10, or is somebody on the forum described at lesson 10, not seven, which is probably a clearer way to think about this, we're going to be talking about NLP before we do. Let's do a quick review of last week because last week you know, there's quite a few people who have kind of flown here to San Francisco for this inverse. Of course, I'm seeing them pretty much every day, they're working full time on this, and quite a few of them are still struggling to understand the material from last week right. So, if you're, finding it difficult, that's that's fine. One of the reasons I kind of put it up there up front is so that we've got something to cogitate about and think about gradually work towards for the by lesson. 14. Mod 7. Your you'll get a second crack at it. Okay, but it's it's world work that there's so many pieces, and so hopefully you can keep developing a better understanding to understand the pieces. You'll need to understand. You know the shapes of convolutional layer, outputs and receptive fields and loss, functions and and everything right. So it's all stuff that you're going to understand need to understand for all of your deep learning studies anyway right, so everything you do to develop an understanding of last week's lesson is going to help you everything else.

So one key thing I wanted to mention is we started out with something which is really pretty

simple, which is single person, classifier, a single object, class player, single object, bounding box without a classifier, and then single object, classifier and darling box, and anybody who's hopefully spent Some time studying since lesson 8 mod seven has got to the point where they understand this bit right now. The reason I mention this is because the bit where we go to multiple objects is actually almost identical to this, except we first have to solve the matching problem. We ended up creating far more activations than we need for our number of bounding boxes. Ground truth, bounding boxes, and so we match each ground. Truth object to a subset of those activations right and once we've done that the loss function that we then do to which matched pair is almost identical to this loss function right. So if you're feeling stuck go back to lesson eight right and make sure you understand the data set, the data loader and, most importantly, the loss function from the end of lesson, eight or the start of lesson: nine: okay, okay! So once we've got this thing, which can predict the class and bounding box for one object, we went to multiple objects by just creating more activations. We had to then deal with the matching problem.

Having done with the dealt with a matching problem, we then basically moved each of those anchor boxes in and out a little bit and around a little bit, so they tried to line up with a particular ground. Truth objects and we talked about how we took advantage of the convolutional nature of the network to to try to have activations that had a receptive field that was similar to the ground truth object. We were predicting and khlo sultan provided this fantastic picture. I guess for her own notes, but she shared it with everybody which is lovely to talk about like what is SSD mode. He had to forward do line by line, and I apparently wanted to show this to help you with your revision. But I also can't you wanted to show through this show this to kind of say doing this kind of stuff is very useful for you to do like walk through and in whatever way, helps you make sure you understand something, and you can see what Chloe's done Here is she's focused, particularly on the dimensions of the tensor at each point, and the in the in the path has be kind of gradually down sampling, using these tragic convolutions making sure she understands why those grid sizes happen and then understanding how the outputs come out Of those okay, and so one thing you might be wondering, is well how did Chloe calculate these numbers? So I don't know the answer I have spoken to her, but obviously one approach would be like from first principles just thinking through it, but then you want to know what am i right right, and so this is where you've got to remember this pdb dot set Trace idea right so I just went in just before class and went into SSD multi-head dot forward and entered pdb dot set trace, and then I ran a single batch right and so at that.

So I put the trace at the end and then I could just print out the size of all of these guys right so which, by the way, reminds me last week. There may have been a point where I said: 21 plus 4 equals 26, which is not true in most universes and like by the way when I code I do that stuff. Like that's the kind of thing I do all the time. So that's why we have debuggers and know how to check things and do things in small little bits along the way anyway. So this idea of putting a debugger inside your forward function and printing out the sizes is something which is damn super or you could just put a print statement here as well. So I actually don't know if that's how Khloe figured it out, but that's how I would if I was her and then we talked about increasing K, which is the number of anchor boxes for each convolutional grid cell, which we can do with different zooms and different Aspect ratios, and so that gives us a plethora of activations and therefore predicted bounding boxes, which we then went down to a small number using non maximum suppression and I'll. Try and remember, to put a link. There's a really interesting paper that one of our students told me about that. I hadn't heard about which is attempting to like you know: I've mentioned on maximum suppression. It's

like kind of hacky kind of leads totally heuristic.

You know didn't even talk about the code because it seems kind of hideous, so somebody actually came up with a paper recently attempts to do an end-to-end ComNet to replace that NMS piece. So I'll I'll put that paper up nobody's created a torch, a pipe torch implementation yeah, so be an interesting project. If anyone wanted to try that um one thing, I've noticed in our study groups during the week is not enough people reading papers, the the what we are doing in class now is implementing papers. The papers are the real ground. Truth right - and I think you know from talking to people a lot of the reason people aren't reading papers is because a lot of people don't think they're capable of reading papers. They don't think they're the kind of people that read papers, but you are you're here right and like we started looking at a paper last week and we read the words that were in English and we largely understood them right. So it's like if

2. 00:07:30

- **Use Excel to understand Deep Learning concepts**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You actually look through this picture from SSD carefully you'll realize SSD multi-head, but forward is not doing the same as this and then you might think. Oh well, I wonder if this is better, you know - and my answer is probably right, because that's his team, all he had up forward, was like the first thing. I tried just to get something out there. You know, but you know that there are between this and the yellow vert and three paper and stuff they're, probably much better ways. One thing you'll notice in particular: is they use a smaller K, but they have a lot more sets of grids one by one three by three: five by five, ten by ten 19 by 19 and 38 by 38, 8700 per class right. So a lot more, then, the we hats, they've been interesting thing to experiment with. Another thing I noticed is that where else we had four by four two by two one by one, which means there's a lot of overlap like every set fits within every other set in this case, where you've got one three: five: there's that you don't have that Overlap right, so it might actually make it easier to learn so there's lots of interesting things you can play with based on stuff. That's out, you know either trying to make it closer to the paper or think about other things. You could try that aren't in the paper or whatever. Perhaps most important thing I would recommend, is to put the code and the equations next to each other.

Yes, Rachel. There was a question of whether you could speak about the used cyclic learning rate argument and the fit function we broke it there so put the code and the equations from the paper next to each other and draw in one of

3. 00:09:20

- **ImageNet Processing (continued from Lesson 9)**
- **& Tips to speed up your model (simd & parallel processing)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Two groups: you are either a code person like me who's, not that happy about math, in which

case I start with a code, and then I look at the math and I learn about how the math master the code and end up eventually understanding the math. All your PhD in stochastic differential equations like Rachel, whatever that means, in which case you can look at the math and then learn about how the code implements the math right then either way unless you're, one of those rare people who is equally comfortable in either world You'll learn about one or the other. Now learning about code is pretty easy because there's documentation and we know how to it's indexed up and so forth. Sometimes learning the math is hard because the notation might seem how to look up, but there's actually a lot of resources. For example, list of mathematical symbols on Wikipedia is amazingly great. It has examples of the explanations of what they mean and tells you what to search for to find out more about it, really terrific, and if you, google, for math notation cheat sheet you'll find more of these kinds of terrific resources. Ok, so over time you do need to learn the notation, but as you'll see from the Wikipedia page, there's not actually that much division right, obviously, there's a lot of concepts behind it, but once you know the notation, you can then quickly look at the concept as It pertains to a particular thing: you're studying nobody learns all of math and then starts learning machine learning right, everybody, even top researchers. I know when they're reading a new paper will very often come to bits of math they haven't seen before, and they'll have to go away and what that that bit of math.

Another thing you should try doing is to recreate things that you see in the papers. Right so here was the key most important figure, one from the focal loss paper, the retina paper so recreate it right and like very often I put these challenges up on the forums right so like keep an eye on the lessons register either put on the forums And so I put this challenge up there and within about three three minutes, serrata had said done it in Microsoft, Excel naturally along with actually a lot more information within the original paper. The nice thing here is that she was actually able to draw a line showing at a point, five ground, truth probability. What's the loss for different amounts of gamma, I'm just kind of cool, and if you want to cheat she's, also provided Python code to, I did discover a reminder bug in my code last week. The way that I was flattening out, the convolutional activations did not line up with how I was using them in the loss function and fixing that actually made it quite a bit better. So my motorbikes and cows and stuff we're actually in the right place. So when you go back to that, notebook you'll see it's a little less bad than it was okay. So there's some there's some quick coverage of what's gone before yes, quick question: are you gon na put the PowerPoint on github I'll, put a subset of it on? Okay and then, secondly, usually when we down sample, we increase the number of filters or depth when we're doing sampling from 77 to 44. Why are we decreasing the number from 512 to 256? Why not decrease dimension and SSD head? Is it performance-related? 77? Well, seven by seven, two four by four: I guess they've got star, it's it's it's because well my treats because that's kind of what the papers tend to do is is we've got a number of well.

We have a number of our paths and we kind of want each one to be the same, so we don't want each one to have a different number of filters, and also this is what the the papers did. So I was trying to match up with that and having these 256 it's it's a different concept because we're taking advantage of not just the last layer but the layers before that as well life's easier if we make them more consistent, okay, so we're now going to Move to NLP, and so let me kind of lay out where we're going here. We we've seen a couple of times now this idea of lab taking a pre trained model. In fact, we've seen it in every lesson: take a pre trained model whip off some stuff. On the top, replace it with some new stuff, get it to do something similar right, and so what we're going to do so, and so we've kind of dived in a little bit deeper to that to say, like okay, with convolute a dot pre-trained. It had like a standard way of like sticking

stuff on the top, which does a particular thing which, with some classification, okay and then we learn. Actually we have. We can stick any pytorch module. We like on the end and have it do anything we like with a in a custom here and so suddenly you discover wow, there's, there's some really interesting things we can do in fact. That reminds me reminds me, young Lou, said well what, if we did a different kind of custom in here, and so the different custom hit was well, let's take the original pictures and rotate them and then make our dependent variable the op.

You know the opposite of that rotation, basically and see if it can learn to unrotated, and this is like a super useful thing. Obviously, in fact, I think Google photos nowadays has this option that it'll actually automatically rotate your photos for you, but the cool thing is, as young lord shows here. You can build that Network right now by doing exactly the same as our previous lesson, but your custom had is one that spits out a single number, which is how much to rotate by and your data set, has a dependent variable, which is how much did you Rotate, though yeah so like, you suddenly realized with this idea of a backbone plus a custom head, you can do almost anything you can think about. So today we're going to look at the same idea and say like okay: well, how does that apply to NLP and then in the next lesson, we're going to go further and say like well, if NLP in computer vision kind of lets, you do the same basic Ideas, how do we combine the two and we're going to learn about a model that can actually learn to find word structures from images or images from word structures or images from images, and that will form the basis if you wanted to go further of doing things Like going from an image to a sentence, that's called image captioning for going from a sentence to an image which will kind of have to do a phrased image and so from there.

You know we've got to go deeper, then into computer vision to think like okay, what other kinds of things we can do with this idea of a pre-trained network plus a custom head, and so we'll look at various kinds of image enhancement like increasing the resolution of A low-res photo to guess what was missing or adding artistic filters on top of photos or changing photos of horses into photos of zebras and stuff. Like that and then finally, that's gon na bring us all the way back to bounding boxes again and so to get there. We've got a first of all that about segmentation, which is not just learnt figuring out, where a bounding boxes, but figuring out what every single pixel in an image is part of. So this pixel is part of a person. This pixel is part of a car and then we're going to use that idea, particularly an idea court unit. It turns out that this idea from unit we can apply to bounding boxes where it's called feature pyramids. Everything has to have a different name in every slightly different area and we'll use that to hopefully get some better, better results and really good results. Actually it bounding boxes, so that's kind of our path from here um. So it's all gon na kind of build on each other that take us into lots of different areas. Now for NLP last part, we relied on a pretty great library called torch text, but as pretty great as it was, I've since then found the limitations of it too problematic to keep using it. As a lot of you complained on the forums, it's pretty damn slow, partly that that's because it's doing parallel not doing parallel processing and partly it's because it doesn't remember what you did last time and it does it all over again from scratch.

And then it's kind of hard to do fairly, simple things like a lot of you were trying to get into the toxic comment: competition on cattle, which was a multi-label problem and trying to do that with torch text. I eventually got it working, but it took me like a week of hacking away, which is kind of ridiculous. So to fix all these problems, we've created a new library called fastai blood test. fastai dot text is a replacement for the combination of torch text and fastai, NLP yeah. So don't use fast. I don't NLP anymore. Okay, that's like that's obsolete. It's! It's slower! It's more confusing! It's less good in every way. Right but there's a lot of overlaps.

Okay, like intentionally a lot of the classes, have the same names. All of the functions have the same names, but this is the non torch text: okay, okay, so we're going to work with IMDB again, alright. So, for those of you have forgotten, go back and check out lesson 4, that basically this is a data set of movie reviews and you remember, we used it to find out whether we might enjoy some be getting or not, and we thought probably my kind of Thing so we're going to use the same data set and by default it it calls itself a co IMDB. So this is just that the raw data set that you can download and, as you can see, I'm doing from fastai dot text, import, star, there's no torch text and I'm not using faster. I don't an LP I'm going to use path Lea, but, as per usual, we're going to learn about what these tags are later.

So you might remember, the basic paths for NLP is that we have to take sentences and turn them into numbers and there's a couple of steps to get there. Okay, so at the moment, somewhat intentionally fastai dot text doesn't provide that many helper functions. It's really designed more to let you handle things in a fairly flexible way right. So, as you can see here, I wrote something called get texts which goes each thing in classes, and these are the three classes that they have in IMDB negative, positive and then there's another folder unsupervised that stuff they haven't gotten around to labeling. Yet so I'm just going to call that class now, and so I just go through each one of those classes, and then I just find every file in that folder with that name, and I open it up and read it and check it into the end of This all right, okay and as you can see with path lab, it's super easy to grab stuff and pull it in, and then the label is just whatever class or not so right. So I'll go ahead and I'll do that for the Train bit and I'll. Do that for the test fit so there's 70,000 and Tre and 25,000 in test talk about 50,000. If the Train ones are unsupervised, we won't actually be able to use them when we get to the classification piece okay, so I actually find this much easier than the kind of torch text approach of having lots of layers and wrappers and stuff, because in the end, Reading text files is not it's not that hard.

Okay, one thing, that's always good idea is to sort things randomly. It's useful to know this simple trick for sorting things randomly, particularly when you've got multiple things. You have to sort the same way in this case. You've got labels and texts, NP, dot, random block permutation. If you give it an integer, it gives you a random list from 0 up to and not including the number you give it in some random order, and so you can then just pass that in as a indexer right to give you a list. That's sorted in that random order, so in this case it's going to sort train texts and train labels in the same random way. Okay, so that's a useful little idiom to use. So now I've got my texts and my labels sorted. I can go ahead and create a data frame from them. Why am I doing this? Well, the reason I'm doing this is because there is a somewhat standard approach, starting to appear for text classification datasets, which is to have your training set as a CSV file, with the with the labels first and the text of the NLP documents. Second, you know trained on CSV and a test or CSB, so it basically looks like this. You have your labels and your texts and then a file called classes text which just lists the classes. I think somewhat standard is just in a reasonably recent academic paper. Yarn lakorn and a team of researchers looked at quite a few data sets and they use this format for all of them, and so that's what I've started using as well for my recent paper.

So what I've done is you'll find that this notebook, if you put your data into this format, the whole notebook will work every time and so far than having a thousand different classes or formats and readers and writers or whatever have you said. Let's just pick a standard format and your job, your coders, you can do it perfectly well - is to put it in that format, which is the CSV file. Okay, the CSV files have no header okay by default, all right now, you'll notice. At the start here that I had two different paths: one was the classification path. One was the

language model of path in NLP, you'll, see LM all the time. L M means language model, so the classification path is going to contain the information that we're going to use to create a sentiment analysis model. The language model path is going to contain the information we need to create a language model, so they're a little bit different. One thing that's different is that when we create the train, dot CSV in the classification path, we move everything that has a label of two, because label of two is unsupervised okay, so we remove the unsupervised data from the classifier. We cannot use it so

4. 00:26:45

- **Adding Preprocessing to Keras ResNet50**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

That means that this is going to have actually 25,000 positive, 25,000 negative, and the second difference is the labels for the classification path. The labels are the actual labels, but for the language model there are no labels, so we just use a bunch of zeros and that just makes it a little bit easier because we can use a consistent data frame format or CSV format. Okay, now the language model we can create our own validation set and so you've probably come across by now. scikit-learn but model selection, dot, train test split, which is a really simple, little function that grabs a data set and randomly splits it into a training set and the validation set according to whatever proportion you specify, and so in this case I concatenate my classification, training And validation together, so that's got to be a hundred thousand altogether split it by ten percent, and so now I've got 90,000 training. Ten thousand validation for my language model, so I'll go ahead and save that. So that's my basic, you knowget the data in a standard format for my language model in my classifier right. So the next thing we need to do is tokenization, so tokenization means at this stage we've got for a document for a movie review. We've got a big long string and we wanted to pretend it into a list of tokens which are kind of a list of words, but not quite right.

For example, don't we want to be door? Hmm, you probably want full stop to be a token and so forth.

5. 00:28:30

- **Transfer Learning with ResNet in Keras: difficulty #1**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Right, so tokenization is something that we passed off to a terrific library called spacey, partly terrific, because in australian wrote it and partly terrific, because it's good at what it does we put a bit of stuff on top of Spacey, but the vast majority, the works being Done by Spacey before we pass it to Spacey I've written this simple fix-up function, which is basically each time I don't put a different data set and I've looked at about a dozen in building this. Everyone had different, weird things that need to be replaced. So here all the ones I've come up with so far. Hopefully this will help you out as well, so I HTML and escape all the entities and then there's a bunch more things. I replace have a look at the result of running this on text that you put in and make sure there's not more weird tokens in there. It's amazing how many weird

things people do to test. So basically, I've got this function called get all which is going to go ahead and call get texts and texts is going to go ahead and do a few things, one of which is to apply that fix up that we just mentioned. So, let's kind of look through this because there's some interesting things to point out, so I got to use pandas to open our train CSV from the language model path, but I'm passing in an extra parameter.

You may not have seen before called chat, sighs, Python and pandas can both be pretty inefficient when it comes to storing and using text data, and so you'll see that very few people in NLP are working with large corpuses, and I think part of the reason is That traditional tools have just made it really difficult. You ran out of memory all the time, so this process, I'm showing you today, I have used on corpuses of over a billion words successfully using this exact code right and so one of the simple tricks is to use this thing called Chuck size with pandas. What that means is that pandas to not return a data frame, but it returns an iterator that we can iterate through chunks of a data frame, and so that's why, I don't say: truck train equals, get texts because here is a finger texts, but instead I call Get all which loops through the data frame, but actually what it's really doing, is its looping through chunks with the data frame. So each of those chunks is basically a data frame representing a subset of the data. When I'm working with NLP data many times I come across data with foreign text or characters, is it better to discard them or keep them? No, no definitely keep them, and this whole process is is Unicode and I've actually used this on Chinese text. This is designed to work on pretty much anything yeah yeah in general. Most of the time.

It's not a good idea to remove anything like old-fashioned. Nlp approaches tend to do all this like limit ization and all these kind of normalization steps to kind of get rid of. You know that lower case everything blah blah blah, but you know that's throwing away information which you don't know ahead of time, whether it's useful or not. So don't throw away information. Ok, so we go through each chunk, each of which is a data frame, and we call get texts. Get texts is going to grab the labels and makes them into ants it's going to grab. Then the the texts and I'll point out a couple of things. The first is that before we include the text, we have this beginning of stream - token, which you might remember we used way back up here, there's nothing special about these particular strings of letters they're. Just once I figured don't appear in normal text, it's very often, so every text is going to start with X POS now. Why is that? Because it's often really useful for your model to know when a new text is starting, for example, if it's a language model right, we are going to concatenate all the text together and so it'd be really helpful for it to know all this articles finished and a New one started, so I should probably like forget some of their context. Yeah Devo is quite often, texts have multiple fields like a title and abstract and then the main document, and so by the same token, I've got this thing here, which alerts us actually have multiple fields in our CSV okay.

So this really, this process is designed to be

6. 00:33:40

- **Transfer Learning with ResNet in Keras: difficulty #2**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Very flexible and again at the start of each one we put a special field, starts here token, followed by the number of the field. That's starting here for as many fields as we have okay,

then we apply our fix up to it and then, most importantly, we tokenize it and we tokenize it by doing a process or multiprocessor multi processing. I guess I should say, and so tokenizing tends to be pretty slow, but we've all got multiple cores and our machines now and some of the better machines on AWS and stuff can have dozens of course, here on a university computer, we've got 56 cause, so Spacey Is not very amenable to multi processing, but I finally figured out how to get it to work, and the good news is it's all wrapped up in this one function now, and so all you need to pass to that function is a list of things to tokenize, Which each part of that list will be tokenized on a different core, and so i've also created this function called partition by cause which takes a list and splits it into sub lists. The number of sub lists is the number of cores that you have in your computer, ok, so on on my machine without more -- processing. This takes about an hour and a half and with more processing. It takes about two minutes alright. So it's a really handy thing to have and now that this codes here, you know, feel free to look inside it and take advantage of it for your own stuff right. Remember. We all have multiple processes protocol cause, even in our laptops and very few things in place that and take advantage of it unless you make a bit of an effort to make it work.

So, there's a couple of tricks to get things working quickly and reliably as it runs. It prints out how it's going and so here's the result at the end right, beginning of stream token, beginning of field number one token: here's the tokenized text you'll see that the punctuation is on the whole. Now a separate token you'll see there's a few interesting little things. One is this: what's this t up, t up MGM for MGM obviously was originally capitalized right, but the interesting thing is that normally people are the lowercase everything or they leave the case, as is now, if you leave the case, as is then screw you or caps, And screw you lower case are two totally different sets of tokens that have to be learned from scratch or if you love a case them all, then there's no difference at all between screw or you and right. So how do you fix this so that you both get a semantic impact of like I'm shouting now right, but not have every single word have to learn the shouted version versus the normal version, and so the idea I came up with and I'm sure other people Have done this too, is to come up with a unique token to mean. The next thing is all uppercase. So then I lowercase it so now, whatever used to be up case, it's now lowercase, it's just one token, and then we can learn the semantic meaning of all uppercase, and so I've done a similar thing. If you've got like 29 exclamation marks in a row, we don't learn a separate token for 29 exclamation marks.

Instead, I put in a special token for the next thing repeats lots of times, and then I put the number 29 and then I put the explanation. Mark. Okay and so there's a few little tricks like that, and if you're interested in LP have a look at the code tokenizer for these little tricks that I've added in because some of them are kind of fun. Okay, so the nice thing with doing things this way is we can now just NP dot, save that and load it back up later, like we don't have to recalculate all this stuff. Each time like we tend to have to do with with torch text or a lot of other libraries

7. 00:38:00

- **Use batches to overcome RAM “Out of Memory”**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Okay, so we've now got it tokenized. The next thing we need to do is to turn it into numbers,

which we call numerical Eisinger and the way we numeric lies. It is very simple: we make a list of all the words that appear in some order and then we replace every word with its index into that list, the list of all the words that appear or all the tokens that we call the vocabulary. So here's an example of some of the vocabulary, the counter class in in plaque. That is very handy for this. It basically gives us a list of unique items and their counts. Okay, so here are the 25 most common and things in the vocabulary. You can see there are things like apostrophe s and double quote and end of paragraph and also stuff like that. Now, generally speaking, we don't want every unique token in our vocabulary, if it doesn't appear at least two times, then might just be a spelling mistake or a word I mean we can't learn anything about it if it doesn't appear that often also the stuff that we're Going to be learning about at least so far in this part gets a bit clunky once you've got a vocabulary bigger than 60-thousand time permitting. We may look at some work. I've been doing recently on handling larger vocabularies, otherwise that might have to come in a in a future class okay, but actually for classification. I've discovered that doing more than about 60,000. Words doesn't seem to help anyway, so we're going to limit our vocabulary to 60,000 words, things that appear at least twice, and so here's a simple way to do that: use that dot most common pass in the mass vocab size, that'll sort it by the frequency by The way and if it appears less often than a minimum frequency, then don't bother at all okay, so that gives us Ida s. That's the the same name that torch text used.

Remember. It means that into string. So this is just the list of tokens unique tokens in the vocab. I'm going to insert two more tokens a token for unknown, evoke a bite of your unknown and vocab item for padding. Okay, then we can create the dictionary which goes in the opposite direction. So string to end and that won't cover everything, because we intentionally truncated it down to 60,000 words right, and so, if we come across something that's not in the dictionary, we want to replace it with zero for unknown, and so we can use a default dick. For that, with a lambda function that always returns Europe, okay, so you can see all these things were using the kind of kick coming coming back up. So now that we've got our s to a dictionary defined. We can then just call that for every word for every sentence right and so there's an Americanized version and there it is okay, and so, of course, the nice thing is again. We can save that step as well. That's so, each time we get to another step. We can save it, and these are not very big files compared to what you used to with with images text is generally pretty small, very important to also save that vocabulary, but because the this, this list of numbers means nothing. Unless you know what each number refers to and that's what I to restaurants, you okay, so you save those three things and then later on, you can load them back up. So now our vocab size is 60,000 and term and training language model has 90,000 documents.

You know okay, so

8. 00:42:00

- **Final layers to our ResNet model**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

That's the pre-processing you do. We can probably rap a little bit more of that in little utility functions if we want to, but it's all pretty straightforward and basically that exact code will work for any data set. You have once you've got it in that CSV format. So here is a kind of a

new insight, that's not new at all, which is that we'd like to pre train. Something like we know from lesson, four, that if we pre train our classifier by first creating a language model and then fine-tuning that as a classifier. That was helpful. Do you remember it actually got us a new start at the out result. We got the best IMDb classifier result that had ever been published, but quite a bit. Well, we're not going that far enough, though right, because IMDB movie reviews are not that different to any other English document. You know compared to how different they are to a random string or even to a Chinese document. So, just like imagenet allowed us to train things that recognized stuff. That kind of looks like pictures and we could use it on stuff. That was nothing to do with image net like satellite images, why don't we train a language model? That's just like good at English and then fine tune it to be good at major reviews. So this basic insight Leadbeater try building a language model on Wikipedia, so my friend Steven marady has already processed Wikipedia found a subset of nearly you know the most of it, but throwing away the stupid little articles so most of the bigger articles, and he calls that Wiki text 103, so I grabbed wiki text 103 and I trained a language model honor and we and I used exactly the same approach, I'm about to show you for trading an IMDB language model, but instead I trained a wikitext 103 language model and then I saved It and I've made it available for anybody who wants to use it at this URL. So this is not a URL for wikitext 103. The documents this is the wiki text - 103, the language model, and so the idea now is: let's train an IMDB language model which starts with these words there, hopefully to you folks.

This is a extremely obvious, extremely non-controversial idea, because it's basically what we've done in nearly every class so far, but when I first when I first mentioned this to two people in the NLP community. I guess, like June July of last year, there couldn't have been less interest. You know I asked on Twitter, where a lot of the top thread of research is kind of people that I follow and they'll be back. I was like hey what if we like pre-trained like a general language model like no old language, is different. You know you can do that or, like I don't know why you would bother. Anyway. I've talked to people at conferences and they're, like pretty sure, people have tried that and it stupid like there was just this like. I don't know this weird like frank past, and you know I guess, because I am arrogant in obstreperous. I ignored them, even though they know much more about NLP than I do, and just trade it anyway, and let me show you what happened so: here's how we do it right, grab the wiki text models right and if you use double you get minus R it'll. Actually, recursively grab the whole directory. It's got a few things or not. We need to make sure that our language model has exactly the same, embedding size, number of hidden and number of layers. As my wiki text, one did otherwise. You can't load the way it's in here, so here's our pre-trained path, here's our preteen language model path; let's go ahead and torch dot load in those weights from the forward.

Wickety text 103 model; okay, we don't normally use torch dot load. But that's that's! That's the you know the pytorch way of

9. 00:47:00

- **Nearest Neighbors to look at examples**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Grabbing a file - and it basically gives you a dictionary containing the name of the layer and a

tensor of those weights or an array of those words. Now, here's the problem that wiki text language model was built with a certain vocabulary, which was not the same as this one was built on right. So my number 40 was not the same as wiki text. 103 models number 40. So we need to map one to the other. Okay, that's very very simple, because luckily I saved the I to s for the wiki text: okay right! So here's the list of what each word is when I trained the wiki text 103 model, and so we can do the same default dict trick to map it in Reverse right and I'm going to use minus 1 to mean that it's not in the wiki text. Dictionary, and so now I can just say: okay, my new set of weights is just a whole bunch of zeros, with vocab sighs by embedding size, so we're going to create an embedding matrix. I'm then going to go through every one of the words in my IMDB vocab room. Okay, I am going to look it up in s, I so a string to int for the wiki text 103 vocabulary and see if it's words there. Okay - and if that is word there, then I'm not going to get this minus one right, so I will be greater than or equal to zero. So in that case, I will just set that row of the embedding matrix two to the weight that I just looked that which was stored inside this this named element, and so these these names you can look at.

You can just look at this dictionary and it's pretty obvious what each name corresponds to. It looks very similar to the names that you gave it when you set up your module. So here are the encoder weights. Okay, so I grabbed it from the encoder weights. If I don't find it, then I will use the row mean in other words, here is the average imbedding weight across all of the wikitext? What are three things? Okay, so that's pretty simple. So I'm going to end up with an embedding matrix for every word. That's in both my vocab room for IMDB and the week in text 103 vocab. I will use the wiki text 103, embedding matrix weights for anything else. I will just use whatever was the average weight from the wiki text 103, embedding matrix, okay and then I'll go ahead, and I will replace the encoder weights with that turn into a tensor. We haven't talked much about weight tying. We might do so later, but basically the decoder. So the thing that turns the final prediction back into a word uses exactly the same weights. So I pop it there as well and then there's a bit of a weird thing with how we do embedding dropout. That ends up with a whole separate copy of them for a reason that doesn't matter much anyway, so we just popped the weights back where they need to go. So this is now something that a dictionary we can now or a set of torch state which we can load in.

So let's go ahead and create our language model, okay and so the basic approach we're going to use - and I'm going to look at this in more detail in a moment. But the basic approach are going to use. Is I'm going to concatenate all of the documents together into a single, a single list of tokens of length? Twenty four point: nine: nine: eight million okay. So that's going to be what I pass in as my trainings, so the language model right, we basically just take all our documents and just concatenate them back to that: okay and we're going to be continuously trying to predict. What's the next word after these words? What's the next word after these words, have a look at these details in a moment, I'm going to set up a whole bunch of drop out. Look at that in detail moment. Once we've got a model data object. We can then grab the model from it. So that's going to give us a learner, okay and then, as per usual, we can call lo not fit so we first of all, as per usual, just do a single epoch on the last layer just to get that. Okay and the way I've set it up is the last layer is actually the embedding words because that's obviously the thing that's going to be the most wrong, because, like a lot of those, embedding weights didn't even exist in the vocab, so we're just gon na train A single epoch of just the inventing weights and then we'll start doing a few epochs of the full model, and so how is that? Looking? Well, here's lesson, four, which was our academic world's best ever result and after 14 he parks.

We had a four point: two three loss here. After one epoch, we have a four point: one two loss right so by pre-training on wikitext 103. In fact, let's go and have a look. We kept training and training at a different rate. Eventually, we got to four point, one six, so by pre training on wikitext 103, we have a better loss after one epoque than the best loss we got for the language model. Otherwise, yes, Rachel. What is the wikitext 103 model? Is it a double UDL STM again? Yeah and we're about to dig into that. It's the way I trained. It was literally the same lines of code that you see here, but without pre-training it only takes one two, three okay, so let's take a 10-minute break, come back at 7:40 and dig in and have a look at these models. Okay, welcome back before we go back into language models and NLP classifiers a quick discussion about something pretty new at the moment, which is the faster I dock project. So the goal of a fast IO dock project is to create documentation that makes readers say wow. That's the most fantastic document, documentation, I've ever read, and so we have some specific ideas about how to do that. But it's the same kind of idea of like top down. You know thoughtful, take full advantage of the medium approach. You know interactive experimental code first that we're all familiar with if you're interested in getting involved the basic approach you can see in in the docs directory. So this is the this is the readme in the docs directory.

In there there is a, amongst other things, a transforms template dot, a doc. What the hell is a doc, hey Doc is asciidoc. How many people here have come across sq doc? That's awesome. Ascii table is people are laughing because there's one hand up, and it's somebody who was in our study group today, who talked to me about ask you dog, ask you doc is the most amazing project, it's like markdown, but it's like what markdown needs to be to Create actual books and like a lot of actual books are written in ASCII doc, and so it's as easy to use this back down about there's way more cool stuff. You can do with it. In fact, here is an ASCII doc file here, right and as you'll see, it looks very normal there's headings right and this is pre formatted text and there's yeah

10. 00:55:00

- **Fine-Tuning our models and more “Out of Memory” fixes**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

There's lists and whatever else it looks pretty standard and actually I'll show you a more complete sq doc thing more standard after doc thing, but you can do stuff like say, put a table of contents here, please, you can say means put a definition list here. Please plus means this is a continuation of the previous list item and so there's just like little things that you can do, which are super handy or like put this thing, make it slightly smaller than everything else. So it's like turbocharged markdown, and so this asciidoc creates this HTML and I didn't add any CSS or do anything myself like. We literally started this project like four hours ago. So this is like just an example. Basically right, and so you can see, we've got a table of contents. We can jump straight to here. We've got a cross reference. We can click on to jump straight to the cross reference, each method, kind of comes along with its details and so on and so forth right and to make things even easier, rather than having to know that this is meant to be like. The argument list is meant to be smaller than the main part, or how do you create a cross reference or how you meant to format the arguments to the method name and list out each one of its arguments? We've created a special template where you can just write various stuff in curly brackets. Like please put the arguments here and here is an example of one argument,

and here is a cross reference, and here is a method and so forth. So we're in the process of documenting the documentation, template that there's basically like five or six of these little curly braket things you'll need to learn.

But for you to create the documentation of class or a method, you can just copy one. That's already there basically, and so the idea is we're going to have like it'll almost be like a book. You know, there'll be tables and pictures, little videos, segments and hyperloop throughout and all that stuff. You might be wondering what about docstrings, but actually I don't know if you've noticed, but if you look at the Python standard library and look at the doc string, for example, for reg X, compile it's single-line. Nearly every dock string in Python is a single line, and Python then does exactly this. They have done a website containing the documentation that says like hey. This is what regular expressions are, and this is what you need to know about them and if you want them to grow fast to unity, his compile and here's what some information about compile and his examples. It's not in the dog stream and that's why we're doing it as well. Our dog strings will be one line unless you need like two. Sometimes it's going to be very similar to Python, but even better. So everybody is welcome to help contribute to the documentation and hopefully by the time, if you're, watching this on the MOOC it'll be reasonably fleshed out and we'll try to keep a list of things to do all right. So I'm going to do one first. So one question that came up in the break was: how does this compare to word to vet, and this is actually a great thing for you to spend time.

Thinking about during the week is how does this compare to work? L�vesque I'll give you the summary now, but it's a very important conceptual difference. The main conceptual difference is what is word to vet word. Tyvek is a single embedding matrix. Each word has a vector and that's it so, in other words, it's a single. It's a single layer from a pre-trained model and specifically that layer is the input layer and also specifically, that pre-trained model is a linear model. Okay, that is free trained on something called a co-occurrence matrix. So we have no particular reason to believe that this model has learned anything much about the English language or that it has any particular capabilities, because it's just a single linear layer and that's it. So, what's this wikitext 103 model, it's a language model and it has a four hundred dimensional, embedding matrix three hidden layers with 1,150 activations per layer and regularization, and all that stuff tied input. Output matrix equator sees it's basically a state-of-the-art AWD ASTM. So like. What's the difference between a single layer of a single linear model versus a three layer, recurrent neural network everything you know: they're they're, very different levels of capability and so you'll see when you try using a pre trained language model. This is a what Vic layer you'll get very, very different results to the vast majority of tasks.

What if the numpy array does not fit in memory? Is it possible to write a pytorch data loader directly from a large CSV file? It almost certainly won't come up, so I'm not going to spend time on it like these things are tiny. These they're just hints think about how many Institute would need to run out of memory. That's not gon na happen. They don't have a kitten's jpg memory. Just in your memory, so I've actually done a another Wikipedia model which I called Giga wiki, which was on all of Wikipedia and even that easily fits in there and the reason I'm not using. It is because it turned out not to really help very much business. Wiki text 103, but you know I've built a bigger model than anybody else. I've found in the academic literature pretty much and it fits in memory on a single machine. What is the idea behind averaging the weights of embeddings they're going to be set to something? You know like there are words that weren't there, so other options is, we could leave them a zero, but that seems like a very extreme thing to do. Like zero is a very extreme number. Why would it

be zero? We could set it equal to some random numbers, but if so, what would be the mean and standard deviation of those random numbers, or should they be uniform if we just average the rest of the you know the embeddings, then we have something that's reasonable scale. I just declare this is how you're initializing words that didn't appear in the training.

Yeah, that's right and then I think you've pretty much kind of just answered this one. But someone had asked if there's a specific advantage to creating our own pre-trained, embedding over using glob or word back yeah. I think I know we're not creating a preacher and embedding we're putting a pre training model. Okay, so um. Let's talk a little bit more with this is a ton of stuff we've seen before, but it's changed a little bit. It's actually a lot easier than it was in part one, but I want to go a little bit deeper into the language model. Loader, okay. So this is the language model loader, and I really hope that by now you've learned in your editor or IDE how to jump to symbols. Okay, yeah! I don't want it to be a burden for you to find out what the source code of language model loader is alright, and if it's still a burden, please, you know go back and try and learn those keyboard shortcuts in vs code. You know like if

11. 01:03:00

- **Find images similar to a word or phrase &**
- **Find images similar to an image !**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Your editor doesn't make it easy, don't use that editor anymore. Okay, there's lots of good free editors that make this easy; okay, so so here's the source code for language model, loader and yeah. It's it's it's interesting to notice that it it's not doing anything particularly tricky. It's not deriving from anything at all right. What makes it something that's capable being a data loader is that it's something you can iterate over okay, and so specifically, okay, so, specifically, here's the fit function inside foster yo model. This is like whatever, where everything ends up. Eventually, which goes to each epoch and then it creates an iterator from the data loader and then just does a for loop through it. Alright, so anything you can do a for loop through can be a data. Loader specifically, it needs to return tuples of mini-batches, independent and dependent variable for mini batches. So anything with a Dunda in a method is something that can act as an iterator and yield is a neat little Python keywords. You probably should learn about. If you don't already know it, but it basically spits out a thing and wets for you to ask for another thing: normally in like a for loop or something. So in this case we start by initializing the language model, passing it in the numbers.

So this is the numerical eyes, big big long list of all of our documents, concatenated together, and the first thing we do is to batch of fudge, and this is the thing which quite a few of you got confused about last time right if our batch size Is 64 and our we have 24 25 million numbers in our list. We are not creating items of length 64 we're not doing that we're creating 64 items in total, so each of them is of size, T divided by 64, which is three hundred and ninety thousand. Okay, so that's what that's what we do here when we reshape it, so that this axis here is of length 64 and then this minus one is everything else. So that's three hundred and whatever a thousand three hundred ninety thousand blob, okay and then we transpose it. So that means that we now have 64 columns three hundred ninety thousand rows and then what we do, each time we do an iterate is we grab one batch of some sequence length, we'll

look at that in a moment, but basically it's approximately equal to VP TT, Which we set to 70 stands for back prop through time and we just grab that many rows. Okay, so from i to i plus 70 rows, and then we try to predict that plus one remember so we're trying to predict when past, where a wrap, so we've got 64 columns and each of those is one sixty-fourth of our 25 million or whatever it was Tokens, you know hundreds of thousands long and we just grab.

You know 17 at a time, so each of those columns each time and Gravatt is going to kind of hook up to the previous column. Okay, and so that's why we get this consistency. This language model is it's stateful, which is really important. Pretty much. All the cool stuff in the language model is is stolen from Steven Mara, T's, AWD, LS TM, including this little trick here, which is if we always grab 70 at a time. And then we go back at me to a new epoch, we're going to grab exactly the same batches every time, there's no randomness. Now, normally we shuffle out data every time we do an epoch or every time we grab some data, we grab it at random. You can't do that with a language model, because this set has to join up to the previous set, because it's fun to trying to learn the sentence right and if you suddenly jump somewhere else, then that doesn't make any sense as a sentence. So Stevens idea is to say: okay. Well, since we can't shuffle the order, let's step randomly change, the size, the sequence length, okay and so basically, he says all right: 95 % of the time, we'll use vacate 80 70, but 5 % of the time we'll use. Half that right and then he says you know what I'm not even going to make that the sequence length I'm going to create a normally distributed random number with that average and a standard deviation of

12. 01:08:15

- **Homework discussion**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

5 and I'll make that the sequence length right, so the sequence length is seventy ish, and that means every time we go through we're getting slightly different batches. So we've got that little bit of extra randomness. I asked him Steven marady, where he came up with this idea. Did he think of it and he was like, I think I thought of it, but it seemed so obvious, and I bet I didn't think of it, which is like true of like every time. I come up with an idea or deke why I think it always seems so obvious that um somebody else is thought of it, but I think he thought of it so yeah. So this is like a nice thing to look at if you're trying to do something. A bit unusual with the data motor, it's like, okay, here's, a simple kind of role model you can use as to creating a data loader from scratch, something that spits out batches of do so so our language model loader, just took in all of the documents concatenated Together, along with a batch size and the VPT, now, generally speaking, we want to create a learner and the way we normally do, that is by getting a model, data object and they're, calling some kind of method which have various names. But sometimes you don't often we call that method get model, and so the idea is that the model data object has enough information to know what kind of model to give you. So we have to create that model data object, which means we need that that class, and so that's very easy to do right.

So here are all of the pieces we're going to create a custom learner, a custom model data class and a custom class. So a model data class again this one doesn't inherit from anything. So you really see it is, there's almost nothing to do right. You need to tell it most importantly: what's your training set, give it a data loader? What's the validation set, give it a data loader and

optionally, give it a test, set data, loader, plus anything else that needs to know right, so it might need to know the VPT. It needs to know the number of tokens. That's the vocab size. It needs to know what is the padding index and so that it can save temporary files and models model data, as always need to know the path. Ok, and so we just grab all that stuff and we dump it right and that's it. That's the entire initializer. There's no logic there at all. Okay, so then, all of the work happens inside get model right and so get model calls something we'll look at later, which just grabs a normal pie, torch and end module architecture, okay and Chuck's it. On the GPU note, with pytorch. Normally we would save cuda with faster, i it's better to say to GPU, and the reason is that if you don't have a GPU it'll leave it on the cpu and it also provides a global variable. You can set to choose whether it goes on the GPU or not.

So it's a it's a better approach, so we wrapped the model in a language model and the language model is this. Basically, a language model is a subclass of basic model. It basically almost does nothing except it defines layer groups and so remember how, when we do like discriminative learning rates where different layers have different learning rates or like we freeze different amounts, we don't provide a different learning rate for every layer because there can be like A thousand layers we provide a different learning rate for every layer group right. So when you create a custom model, you just have to override this one thing which returns a list of all of your layer groups, and so in this case my last layer group contains the last part of the model and one bit of drop out and the Rest of it, this star here means Paul, is apart, so this is basically going to be one layer per RNN layer, okay, so that's all that is and then finally turn that into a learner and so alone. Oh, you just pass in the model and it turns it into a real owner. In this case, we have overridden learner and the only thing we've done is to say I want the default loss function to be across entropy, okay, so this you know entire set of model, customer model data, custom learner or fits on a single screen, and they always Basically, look like this right, so that's a kind of little dig inside this pretty boring part of the code this.

So the interesting part of this code base is getting out now, because that language model is actually the thing that gives us our AWD lsdm, and it actually contains the big idea that the big, incredibly simple, idea that everybody else here thinks is really obvious - that everybody In the NLP community I spoke to, thought was insane, which is basically every model can be thought of. Pretty much. Every model can be thought of as a backbone plus a head, and if you pre, train the backbone and stick on a random head, you can do fine tuning and that's a good idea, all right, and so here's these two bits of the code literally right. Next to each other, this is kind of all. There is inside this bit of faster I've, dot LM RNN, here's gate, language model, here's gate, classifier, that language model creates an R and n encoder and then creates a sequential model that sticks on top of that. A linear decoder classifier creates an R Ln encoder and then a sequential model that sticks on top of that appalling linear. Classifier well see these. What these differences are in a moment, but you get the basic idea right, they're, basically doing pretty much the same thing. They've got this: it's head linear layer on top, so that's worth digging in a little bit deeper and seeing what's going on here.

Yes, Rachel. There was a question earlier about whether that any of this translates to other languages. Yeah this whole thing works in any language you, like, obviously, would you have to retrain your language model on a corpus from that language, yeah yeah, so the wiki text, 103 pre-trained language model knows English right. You could use it, maybe as a pre train start for, like a French short German model, start by retraining. The embedding layer from scratch might be helpful - Chinese - maybe not so much but like, given that a language model can be trained from any and labeled documents at all. You'll never have to do that right

because every every almost every language in the world has you know plenty of documents. You can grab newspapers web pages. You know parliamentary records, whatever you know, as long as you've got a few thousand documents showing somewhat normal usage of that language, you can create a language model, and so I know some of our students. You know one of our students is a bill collector in a week very embarrassing, tried this approach for Thai and he said like the first model he built easily beat the previous Dennis the anti classifier, like it's yeah like for those of you that are international fellows. This is an easy way for you to kind of to whip out a paper in which you, you know, either create the first ever classifier in your language or beat everybody else's classifier in your language.

And then you can tell them that you've been a student of deep moaning for six months and piss off all the academics in your country, okay, so here's

13. 01:16:45

- **How to: multi-input models on large datasets**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Our edit encoder, it's just a standard inin module most of the text in it is actually just documentation. As you can see it, it looks like there's more going on in it. Then there actually is, but really all there is. Is we create an embedding layer? We create an LST M for each layer. That's been asked for that's it. Everything else in it is dropped out right. Basically, all of the interesting stuff just about in the AWA DL STM paper is all of the places you can put drop out and then the forward is basically the same thing right. It's call the embedding layer. Add some dropout go through each layer. Call that are a ten liya append it to our list of our ports. Add dropout, that's about it! Okay, so it's it's! It's really pretty straightforward and the the paper you want to be reading, as I've mentioned, is the AWD lsdm paper, which is this one here. Regularizing and optimizing lsdm language models and it's it's well-written and pretty accessible right and entirely implemented inside fastai as well right. So you can see all of the code for that paper and like a lot of the code, actually is I'm shamelessly plagiarized, with Stephens permission from his excellent github, repo WBLS TM and the process of which I picked some of his bugs as well. I even told him about them so yeah, so I talked you know, I'm talking and increasingly about. Please read the papers, so here's the paper. Please read this paper and it refers to other papers.

So for things like, why is it that the encoder wait and the decoder wait are the same right? Well, it's because there's this thing called tie weights. This is inside. This is inside that gate, language model. There's a thing called tie weights the defaults to true, and if it's true then the we actually tie, we literally use the same weight matrix for the encoder and the decoder so like they're, literally pointing at the same block of memory. If you like, and so why is that, what's the result of it, that's one of the citations and Stephens paper is also a well-written paper. You can go and look up and learn about wait time. So there's a lot of cool stuff in there. Okay, so we have basically a standard iron in the only reason where it's not standard is it's just got lots more types of dropout in it and then a sequential model. On top of that, we stick a linear decoder, which is literally half the screen of code. It's got a single linear layer, we initialize the weights to some range. We add some dropout and that's it. So it's a linear layer. Okay, so we've got an iron N on top of that mystic linear layer with dropout and we finished okay. So that's the language model, so um. What dropout you choose matters a lot and through a lot of

experimentation, I found a bunch of dropouts. You can see here. We've got like each of these corresponds to a particular accurate, a bunch of dropouts that tend to work.

Pretty well for language models, but if you have less data for your language model, you'll need more dropout. If you have more data, you can benefit from less dropout. You don't want to regularize more than you have to make sense right, rather than having to tune every one of these five things right. My claim is they're already pretty good ratios to each other, so just tune this number. I just multiply it all by something. Okay, so there's really just one number you have to choose so if you're overfitting, then you'll need to increase this number. If you're underfitting you'll need to decrease this number, because other than that these ratio is actually seem pretty good. So one important idea, which may seem pretty minor but again it's incredibly controversial - is that we should measure accuracy when we look at a language model, so normally language models. We look at this. This lost value, which is just cross entropy loss, but specifically, where you nearly always take e to the power of that which the NLP community calls perplexity. Okay, so perplexity is just a ^ cross. Entropy there's a lot of problems with comparing things based on cross entropy loss. I'm not sure I've got time to go into it in detail now, but the basic problem is that it's kind of like that thing. We learned about focal loss, cross entropy loss if you're right yet wants you to be really confident that you're right, you know, so it really penalized as a model that doesn't kind of say like I'm so sure this is wrong.

It's wrong, whereas accuracy doesn't care at all about how confident you are. This cop cares about whether you're right - and this is much more often than the thing which you care about in real life. So this accuracy is what how many? How often do we guess the next word correctly and I just find that a much more stable number to keep track of so so that's a simple little thing that I do okay, so so we train for a while

14. 01:23:15

- **Generative Adversarial Networks (GAN) in Keras**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

And we get down to a 3.9 frost, entropy loss and if you go e ^ that and to kind of give you a sense of like what's happened with language models, if you look at academic papers from about 18 months ago, you'll see them talking about. Perplexities stayed at the our complexities of like over a hundred okay, like the the the rate at which our ability to kind of understand language, and I think, like measuring language model. Accuracy or perplexity, is not a terrible proxy for understanding language. If I can guess what you're going to say next, it's you know, I pretty much need to understand language pretty well and also the kind of things you might talk about pretty well, so this numbers just come down so much. It's been amazing, NLP in the last 12 to 18 months, and it's going to come down a lot more. It really feels like 2011-2012 computer vision. You know we're just starting to understand, transfer, learning and fine tuning, and these basic models are getting so much so much better. So everything you thought about like what NLP can and can't do it's very rapidly going out of that. Like there's lots of stuff NLP is not good at to be clear right. Just like in 2012, there was what's the stuff computer vision wasn't good at, but it's changing incredibly rapidly and now is a very, very good time to be getting very, very good at NLP or starting startups, based on NLP, because there's a whole bunch of stuff which Computers would absolutely at two years ago and now

like not quite as good of people and then next year, they'll be much better than people yeah, two questions, one: what is your ratio of paper reading versus coding in a week gosh? What do you think Rachel? You say me, I mean it's a lot more coding right, it's a lot more coding.

I feel like it also really varies from week to week, like I feel like they're like with that bounding box stuff. You know that, like with that bounding box stuff, there was all these papers and no man through, and so I didn't even know which one to read first and then I'd read the citations and didn't understand any of them, and so there was a few weeks of Just kind of reading papers before I even know what to start coding. That's unusual, though, like most of the time I'm yeah, I don't know anytime, I start reading a paper. I'm always convinced that I'm not smart enough to understand it, always regardless of the paper and somehow eventually I do but yeah I try to spend as much time as I can coding and then the second question is your dropout rate, the same through the training or Do you adjust it and the weights accordingly? I just say one more thing about the last bit, which is very often like the vast majority. Nearly always I after I've read a paper even after I've read the bit that says this is the problem, I'm trying to solve I'll kind of stop there and try to implement something, but I think might solve that problem and then I'll go back and read the Paper - and I read little bits about like all these - are how I solve these problem bits and I'll, be like. Oh, that's, a good idea and then I'll try to implement those, and so like that's why, for example, I didn't actually implement SSD. You know, like my custom here, is not the same: it's because I kind of read the gist of it, and then I tried to create something best as I cord and then go back to the papers and try to see why and so so by the time I got to the focal most paper.

Rachael will tell you. I was like driving myself crazy with like how come. I can't find small objects. How come it's always predicting background? You know I read the focal loss paper and I was like that's why you know so. Like it's so much better when you deeply understand the problem they're trying to solve - and I do find the vast majority of the time by the time I read that bit of the paper, which is like solving a problem - I'm then like yeah, but these three ideas. I came up with, they didn't try, you know and he suddenly realized that you've got new ideas or else. If you just implement the paper, you know mindlessly it's. You know you tend not to have these insights about better ways to do it. Yeah um varying drop out is really interesting and there are some recent papers, actually that suggest gradually changing drop out, and it was either a good idea to gradually make it smaller or gradually make it bigger, I'm not sure which let's try. Maybe one of us can try and find it. During the week I haven't seen it widely used. I tried it a little bit with the most recent paper I wrote and it I had some good results. I think I was graduating, like you get smaller yeah and then the next question is. Am I correct in thinking that this language model is built on word embeddings? Would it be valuable to try this with phrase or sentence embeddings? I ask that I ask this because I saw from Google the other day universal sentence: encoder yeah.

No, this is like this is much better than that. You like to shoot. I mean like it. This is. This is not just an embedding of a sentence. This is an entire model right, so an embedding by definition, is like a fixed thing. Oh, I think they're asking they're saying that this language. Well, i the first question is: is this language model built on word embedded? Yes, but but it's not saying it's a sentence or a phrase: embedding is always a model that creates that right and we've got a model. That's like trying to understand language, it's not just as phrase it's not just a sentence. You know it's a it's a document in the end and it's not just an embedding. That's we're training through the whole thing so like this has been a huge problem with NLP for years. Now is this attachment they have to embeddings, and so even the the paper that the community's been most excited about recently from the from AI to the Allen Institute called

Elmo yo mo, and they found much better results across lots of models. But again it was an embedding. They took a fixed model and created a fixed set of numbers which they then fed into a model, but in in computer vision. We've known for is that that approach of having a fixed, fixed set of features, they're called hyper columns. In in computer vision, people stopped using them like three or four years ago, because fine-tuning the entire model works much better right. So for those of you that have spent quite a lot of time with NLP and not much time with computer vision, you're going to have to start relearning right all that stuff.

You have been told about this idea that there are these things called embeddings and that you learn them ahead of time, and then you apply these things, whether it be word level or phrase level or whatever level. Don't do that all right, you want to actually create a pre trained model and fine-tune it and to it then you'll see some you'll see some specific results, all right. So, as you answer the existing lines for using accuracy instead of perplexity as a metric for the model, could we work that into the loss function rather than just use it as a metric? No, you never want to do that, whether it be computer vision or NLP or whatever it's too bumpy right so first have to be spying, is a loss, function and other thing instead of I use it. In addition to you know, I think it's good to look at the accuracy and to look at the cross entropy, but for your loss function you need something nice and smooth accuracy. Doesn't work very well. You'll, see, there's two different versions of save is save and save. Encoder save saves the whole model as

15. 01:32:00

- **Multi-Layer-Perceptron (MLP)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Per usual, save encoder saves just that bit right, in other words, in the sequential model it saves just that bit and not that bit. In other words, you know this bit, which is the bit that actually makes it into a language model. We don't care about the classifier. We just care about that bit. Okay, so that's why we save two different models here: okay, so let's now create the classifier okay and I'm going to go through this bit pretty quickly because it's the same but like when you go back during the week and look at the code convince yourself. It's the same right. We do get all P, tiresias, P, again, Chuck sighs again get all again save those tokens. Again, we don't create a new I to s vocabulary. We obviously want to use the same vocabulary. We had in the language model, okay too, because we're about to reload the same encoder. Okay same default, dict same way of creating our Americanized list, which, as per before, we can save okay. So that's all the same later on. We can reload those rather than having to rebuild them, so all of our hacker parameters are the same, we're not all of them. Sorry, the construction of the modal hyper parameters are the same. We can change the drop hat optimize a function pick a batch size. That is as big as you can, that doesn't run out of memory, and so this bits a bit interesting, there's some fun stuff going on here. The basic idea here is that for the classifier we do really want to look at one. You know our document right.

We need to say, is this document positive or negative, and so we do want to shuffle the documents right. That's because we we like to shuffle things, but those documents are different lengths, and so, if we stick them all into one batch - and this is a handy thing that last AI does for you - you can stick things of different lengths into a batch and it will Automatically pet them, so you don't have to worry about that. Okay. But if they're wildly

different lengths, then you're going to be wasting a lot of computation times. Then what you one thing there that's 2,000 words long and everything else 250 words long, and that means you end up with a 2005 tensor all right, that's pretty annoying, so James Bradbury who's. Actually, one of Stephen Rarity's colleagues and the guy who came up with torch text came up with a neat idea, which was, let's sort the data set by length, ish right, so kind of make it so. The first things in the list on the whole shorter and the things at the end, but a little bit random as well, okay and so I'll show you how I implemented that. So the first thing we need is a data set right, and so we have a data set passing in the the documents and their labels, and so here's a text data set and it inherits from data data set here - is data set from torch from pytorch And actually data set doesn't do anything at all.

It says you need to get item and if you don't have one you're gon na get an error, you need a length if you don't have one you're gon na get an error okay. So this is an abstract class, so we're going to pass in our X we're going to pass in our Y and get item is going to grab the X and grab the Y and return them. It couldn't be much simpler right, optionally. They could reverse it optionally. It could stick an end or stream at the end optionally. It started beginning we're not doing any of those things. So literally, all we're doing is we're putting in an X putting in a way and that grab an item we're returning the X and the y. As a couple, okay and the length is X, yes, so that that's that's all the data sets right, something with a length that you can in days so to turn it into a data loader. You simply pass the data set to the data loader constructor, and it's now going to go ahead and give you a batch of that at a time. Normally, you can say shuffle equals true or shuffle equals false it'll decide whether to randomize it. For you in this case, though, we're actually going to pass in a sample up perimeter and the sampler is a class we're going to define that tells the data loader shuffle okay, so for the validation set, we're going to define something that actually just sorts right.

It just deterministically sorts it so the that all the shortest documents will be at the start or the longest documents will

16. 01:37:10

- **Deep Convolutional GAN (DCGAN)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Be at the end, and that's going to minimize the amount of padding okay for the training sampler we're going to create this thing. I called a sort H sampler, which also sorts ish right. So this is where, like I really like, pytorch is that they came up with this idea for an API for their data loader, where we can, like hook in new classes, to make it behave in different ways right. So here's a sort sampler but is simply something which again, it has a length which is the length of the data source and it has an iterator which is simply an iterator which goes through the data source sorted by length. Well, the key - and I pass in as the key, a lambda function, which returns the links, okay and so for the sort ish sampler. I won't go through the details, but it basically does the same thing with a little bit of randomness okay. So it's a really beautiful. You know just another of these beautiful little design things in pytorch that I discovered I could take James Bradbury's ideas which he had like written a whole new set of classes around, and I could actually just use the inbuilt hooks inside Python. You will notice data loader, it's not actually PI tortoise data loader, it's actually faster ice data loader, but it's basically almost entirely plagiarized from pytorch but customized in some ways to make it

faster, mainly by using multi-threading instead of multi processing. That's Rachel. Does the pre-trained LS TM depth it'd be bt te to match with the new one we're training? No, no, that the be PTT doesn't need to match.

That's just like how many things do we look at at a time it's about nothing to do with the architecture. Okay, so well now we can call that function. We just wore before get our inane classifier, it's going to create exactly the same encoder, more or less okay and we're going to pass in the same architectural details as before. But this time we can the head that we add on you've got a few more things you can do. One is you: can you can add more than one hidden layer, so this layers here says this is what the input to my classifier section. My head is going to be. This is the output of the first layer. This is the output of the second layer and you can add as many as you like, so you can basically create a little multi-layer, neural net classifier at the end, and so did oh. These are the dropouts to go after each of these layers. Okay, and then here are all of the AWD lsdm dropouts, which we're going to basically plagiarize that idea for our classifier we're going to use

17. 01:40:15

- **Wasserstein GAN in Pytorch**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

The errand in learn - oh just like, before we're going to use discriminative learning rates different layers. Actually this is a separate here. You can try using wait. Okay or not, I've been fiddling around a bit with that to see what happens, and so we start out just training the last layer, and we get ninety two point: nine percent accuracy. Then we freeze one more layer, unfreeze one more layer get. Ninety three point three accuracy, and then we fine-tune the whole thing and after three epochs okay, so this was so here is the famous James pepperi we're talking about. This was kind of the main attempt before our paper came along at using a pre train model and what they did is they used a pre trained translation model, but they didn't fine tune. The whole thing they just took the the activations of the translation model and when they tried IMDB, they got they got 91.8 %, which we beat easily. After only fine-tuning one layer. Okay, they weren't state-of-the-art they're. The state of the art is 94.1, which we beat after fine-tuning the whole thing for three epochs, and so by the end, we're at ninety four point: eight, which is obviously a huge difference because, like in terms of error rate, that's gone down from from five point: Nine and then I'll tell you a simple little trick is: go back to the start of this notebook and reverse the order of all of the documents and then rerun the whole thing right and when you get to the bit that says, WT 103 replace this FWD Before word with PWD for backward, that's a backward english-language model that learns to read English backwards.

So if you redo this whole thing put all the documents in Reverse and change this to backward. You now have a second classifier, which classifiers things by positive or negative sentiment based on the reverse document. If you then take the two predictions and take the average of them, you basically have like a directional model if you've trained each bit separately. That gets you to ninety-five point four percent accuracy. Okay, so we can replace eclis load up from five point. Nine to four point six. So this kind of like twenty percent change in the state-of-the-art, is it's like. It's almost unheard of. You know it's like you have to go back to like Jeffrey Hinton's, imagenet computer vision thing where they drop like thirty

percent off the state-of-the-art like it doesn't happen very often, and so you can see this idea of like just use, transfer learning it's ridiculously powerful that Every new feel thinks their new field is too special and you can't do it right, so it's a big opportunity for all of us, so we turn this into a paper and when I say we, I did it with this guy Sebastian, Trudeau. Now you might remember his name because in lesson five I told you that I actually had shared lesson four with Sebastian, because I think he's a an awesome researcher who I thought might like it. I didn't know him personally at all and much to my surprise. He actually watched the damn video.

I was like what you know what an LP research is gon na watch some beginners video, but he watched the whole video. He was like that's actually quite fantastic. That's like fun! Thank you very much. That's awesome coming from you and he said hey. We should turn this into a paper, and I said I don't write papers. I don't care about papers are not interested in papers. That sounds really boring and he said okay, how about? I write the paper for you and I said you can't really write a paper about this yet because you'd have to do like studies to compare it to other things, they're called oblation studies to see which, if it's actually work like there's no rigor here, I just Put in everything that came in my head and checked it all together and it happened to work, it's like okay. What if I write all the paper and all the oblation studies, then can we read the paper and I said: well, it's like a whole library that, like I haven't, documented and, like you know, I'm not going to yet and like you, don't know how it all Works he said: okay, if I wrote the paper and do the ablation studies and figure out from scratch how the code works. Without bothering you then, can we write the paper. I was like yeah if you did a whole thing. The paper I was like okay, and so then, two days later he comes back. He says: okay, I've done a draft with a paper. So I share this story to say like if you're some, you know student in Ireland - and you want to like - do good work.

Don't did anybody stop you right? I did not encourage him at least right, but in the end it's like look. I want to do this work. I think it's going to be good and I'll figure it out, and you know he wrote a fantastic paper and he did the ablation studies and he figured out how fast I works and now we're planning to write another paper together and so like there's. Some you've got to be a bit

18. 01:46:30

- **Introduction to Pytorch**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Careful right cuz, sometimes I get messages from random people. Saying like I've, got lots of good ideas. Can we have coffee? I don't want to. I don't you know I can have coffee in my office any time. Thank you, but it's very different to say. Like hey, I took your ideas and I wrote a paper and I did a bunch of experiments and I figured out how your code works. They added documentation to it. You know sure we could all submit this to a conference. You sort of mean like there's nothing to stop, you doing amazing work and if you do amazing work that like helps somebody else like in this case. Okay, I'm happy that we have a paper. I don't care about papers, but I think it's cool that you know these ideas now. Have this rigorous study like, let me show you what he did so he took all my code right, so I'd already done all the faster. I got text and stuff like that and as you've seen it lets us work with large corpuses. So you know Sebastian is fantastic. We well-read and he said here's a paper that young

lakorn, some guys just came out with where they tried lots of different classification datasets. So I'm gon na try running your code on all these data sets, and so these are the data sets right and so some of them had. You know many many hundreds of thousands of documents and they were far bigger than, and I think I had tried, but I thought it should work.

Okay, and so you know he had a few good little ideas as we went along, and so you should like totally make sure you, you read, the paper write this paper and he said well this thing that you called in the lessons differential learning rates, differential kind Of means something else like: maybe we should rename it so it's now called discriminative learning rate. So this idea that we had from part one where we use different learning rates for different layers. After doing some literature research, it does seem like that hasn't been done before so it's now officially, I think discriminative learning rates, and so all these ideas like this is something we learnt in Lesson one, but it now has an equation with Greek and everything. Okay. So when you see an equation with Greek and everything that doesn't necessarily mean it's more complex than anything, we did in Lesson one because this one isn't again that idea of, like I'm freezing a layer at a time also it seems to never do number four. So it's now a thing and it's got the very clever name: gradual, I'm freezing. So then one promised what we're going to look at this slanted triangular learning rates. So this actually was not my idea, Leslie Smith, my favorite researchers, who you all now know about emailed me a while ago and said I'm so over so learning rates. I don't do that anymore.

I now do it's like a different version, where I have one cycle which goes up quickly at the start and then slowly down afterwards, and he said I often find it works better. I've tried going back over all of my old data sets and it works better for all of them everyone. So this is what the learning rate looks like right. You can use it in fastai, just by adding use CLR equals to your fit. This first number is the ratio between the highest learning rate and the lowest learning rate, so here this is 1/32 of that. The second number is the ratio between the first peak and the last peak, and so the basic idea is, if you're doing a cycle length. 10, that you want the first cycle, sorry, the first epoch to be the upward bit and the other nine epochs to be the downward bit. Then you would use 10 and I find that works pretty well. That was also Leslie. Suggestion is make about a tenth of it. The upward bit and about my intense the down would be since he told me about it. Actually, it was just maybe two days ago he wrote this amazing paper, a disciplined approach to neural network hyper parameters in which he describes something very slightly different to this again, but the same basic idea. This is a must-read paper. You know it's got all the kinds of ideas that fastai talks about a lot in great depth, and nobody else is talking about this stuff.

It's it's kind of a slog. Unfortunately, Leslie had to go away on a trip before he really had time to edit. It properly so it's a little bit slow reading, but it's don't let that stop you it's amazing. So this triangle this is the equation from my paper - was the best year and Sebastian was like Jeremy. Can you send me the math equation behind that poetry? Rotation? If I just wrote the code, I could not get it into math, so he figured out the math for it. So you might have noticed the first layer of our classifier was equal to embedding size x, 3y times 3 times 3. Because and again, this seems to be something which people haven't done before so new idea, concat pooling, which is that we take the average pool the average pooling over the sequence of the activations, the max pooling of the sequence over the activations and the final set of Activations and just concatenate them all together again. This is something which we talked about in part one, but doesn't seem to be in the literature before so it's now called Combe kept pulling and again it's now got an equation and everything, but this is the entirety of the implementation, pull with average pull with max Concatenate those two along with the final

sequence, so you know you can go through this paper and see how the faster I code implements each piece. So then, to me, one of the kind of interesting pieces is the difference between RNN and coda, which you've already seen and multi batch are in an encoder.

So what's the difference there? Okay, so the key difference is that the the normal RN an encoder for the language model, we could just do V PTT chunk at a time right, no problem and predict the next book, but for the classifier we need to do the whole document. We need to do the whole movie review before we decide if it's positive or negative and the whole movie review can easily be 2,000 words long and I can't fit 2,000 words worth of you know gradients in my GPU memory for every single one of my activations. Well, sorry, for every one of my weights, so what do I do? And so the idea was very simple, which is I go through my whole sequence length one batch of BPT be PTT at a time right and I call super dot forward, so in other words the eridan encoder, all right so just couldn't call the usual iron in Encoder to grab its outputs and then I've got this maximum sequence length parameter where it says: okay, if you've as long as you're doing no more than that sequence length, then start appending it to my list of outputs. Okay, so in other words, the thing that it sends back to this pooling is is only as much there's only as many activations as we've asked it to keep right, and so that way you can basically just figure out how much what's max SEC. Do you can your particular GPU handle that so it's still using the whole document, but let's say max SEC is a thousand thousand words and your longest document length is two thousand words right that it's still going through the I am creating state for those.

First thousand words right, but it's not actually going to store the activations. For the backdrop, the first thousand is only going to keep the last thousand right, so that means that it can't back propagate the loss back to

19. 01:55:20

- **Wasserstein GAN in Pytorch (cont.)**
- **& LSUN dataset**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Any decisions that any state that was created in the first thousand words you know. Basically, that's that's now gone. So it's a really simple piece of code. You know and honestly, when I wrote it, it was like. I didn't spend much time thinking about it. It seems so obviously the only way that this could possibly work, but again it's it seems to be a new thing, so we now have back prop through time for text classification. Yes, I think so you can see there's lots of little pieces in this paper. So what was the result right and so the result was on every single data set. We tried. We got a better result than any previous academic for text classification, so IMDB trekked, 6aj News, dbpedia, Yelp, all different types and honestly IMDB was the only one I spent any time trying to optimize the model so like most of them. We just did it like. Whatever came out first, so if we actually spent time with it, I think this would be a lot better. That and the things that these are comparing to most of them are like you'll, see like they're different on each table because, like they're optimized, you know these. Like customized algorithms, on the whole, so this is saying like one simple fine-tuning: algorithm can beat these really customized algorithms, and so here's the like one of the really cool things that Sebastian did was his ablation studies. All right, which is.

I was really keen that if you're going to publish a paper, we had to say why does it work right? So Sebastian went through and tried. You know removing all of those different contributions. I mentioned right. So what if we don't use gradual freezing? What if we don't use discriminative learning rates? What if, instead of discriminating rates, we use cosign annealing? What, if we don't do any pre training with Wikipedia what, if we don't do any fine tuning and then the really interesting one to me, was: what's the validation error rate on IMDB? If we only use a hundred training examples, this is 200 business 500 and you can see you know very. Interestingly. The the full version of this approach is nearly as accurate on just a hundred training. Examples like it's still very accurate versus for 20,000 training examples. We also, if your training from scratch on 100, it's like almost random, all right so kind of like it's what I expected. You know I've kind of said to Sebastian. I really think that this this is most beneficial when you don't have much data - and this is like - where fast a is most interested in contributing right, there's like small data regimes, small compute regimes and so forth, and so he did these studies to check. So I want to show you a couple of tricks as to how you can run these kinds.

The first trick is is something which I know you're all gon na find really handy. I know you're hoping annoyed when you're running something in a Jeep and a notebook, and you lose your internet connection for long enough, that it decides you've gone away and then your session disappears and you have to start it again from Spanish. Ok. So what do you do? There's a very simple cool thing called VNC, where, basically, you can install on your AWS instance or paper space or whatever X, Windows, a lightweight window manager, a VNC server, firefox a terminal and some fonts chuck these lines. At the end of your VNC x, startup configuration file and then run this command. It's now running a server where, if you now run for it's now running a server where you can then run the type VNC viewer or any VNC viewer on your computer. And you point it at your server right, but specifically what you do as you go. You use SSH port forwarding to port forward port five, nine one three to localhost five, nine one three right, and so then you connect to port five nine one three on on localhost. It will send it off to port five nine one, three on your server, which is the VNC port, because you said thirteen here and it will display an X Windows desktop and then you can. Click on the Linux start like button and click on Firefox and you now have Firefox, and so you can now run and you'll see here in Firefox. It says localhost because this Firefox is running on my AWS server right, and so you now run Firefox. You start your thing running and then you close your VNC viewer, remembering that Firefox is like displaying on this virtual VNC display, not in the real display, and so then later on that day you log back into VNC viewer and it flops up again. So it's like you're persistent desktop and it's shockingly fast.

It works really well. Okay, so there's trick number one and there's lots of different VNC servers and clients and whatever. But this one works fine for me, so there you go so you can see here. I connect to localhost five nine one three trick number two is to create Python scripts, but this is what we ended up doing so I ended up creating like a little Python script for Sebastian to kind of say this is the basic steps you need to do And now you need to create like different versions, certain thing else, and I suggested to him that he tried using this thing, called Google fire. What Google fire does is you create a function with shitloads of parameters right, and so these are all the things that Sebastian wanted to try doing. Different drop out amounts. Different learning rates. Do I use pre training or not? Do I use CLR or not? Do I use discriminative learning rate or not? Do I go backwards or not blah blah blah right? So you create a function and then you add something saying if 9 equals main fire dot fire and the function name, you do nothing else at all. You don't have to add any metadata, any docstrings anything at all, and you then call that script and automatically you now have a

command. That's it right. So that's a super fantastic, easy way to run lots of different variations in a terminal - and this is like this is ends up being easier.

If you want to do lots of variations than using a notebook, because you can just like - have a bash script that tries all of them and spits them all out - you're fine inside the dl2 course directory. There's now something called IMDB scripts and I've put there all of the scripts that Sebastian and I used so you'll - see because we needed to like tokenize every or data set. We had to turn every dataset numerical eyes. Every data set. We had to train a language model on every data set. We had to train a classifier every data set. We have to do all of those things in a variety of different ways to compare them. We have a script for all those things, so you can check out and see all of the scripts that we used when you're doing a lot of scripts and stuff they've got different code all over the place. Eventually, it might get frustrating that you want to. You know you don't want a symlink you're, faster yo library again and again, but you probably don't want to pip install it because that version tends to be a little bit old. We move so fast. You want to use the current version in get if you say, pip install a dot from the fastai repos base. It does something quite neat which is basically creates a symlink, the faster a library to get. You know in your get installation right here inside your site packages directory your site packages directory is like your main. You know Python library, and so, if you do this, you can then access fastai from anywhere.

But every time you do get pull you've got the most recent version. One downside of this is that it installs any updated versions of packages from Kipp which can kind of confuse Condor a little bit. So another alternative here is just to Simla -- nk, the FASTA, a library to your site packages, library like that works just as well, and then you can use faster i'll again from anywhere and it's quite handy when you want to kind of run scripts.

20. 02:05:00

- **Examples of generated images**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

That use past AI from different directories under system. Okay, so one more thing before we, which is something you can try if you like, you, don't have to tokenize words. Instead of tokenizing words, you can tokenize what are called sub word units and so, for example, unsupervised for example, unsupervised quickly. Tokenized his on supervised tokenizer could be tokenized as token either right and then you could do the same thing. The language model that works on sub units, a classifier that works on sub word units, etc. So how well does that work? I started playing with it and with not too much playing. I was getting classification results that were nearly as good as using word level. Tokenization not quite as good, but nearly as good, I suspect, with more careful thinking and playing around. Maybe I could have got as good or better, but even if I couldn't, if you create a a sub word unit, wikitext model, then IMDB model language model and then classifier forwards and backwards, force upward units and then ensemble it with the forwards and backwards word level Ones, you should be able to beat us right, so here's an approach you may be able to beat our state if ya was on. Google has, as Sebastian told me about this particular project. Is great. Google has a project called sentence: peace which actually uses a neural net to figure out the optimal splitting up of words, and so you end up with a vocabulary of sub word units.

In my planning around, I found that creating a vocabulary of about 30,000 sub word units seems to be about optimal. So if you're interested there's something you can try, it's a bit of a pain to install it's C++. It doesn't have great error messages, but it will work like before it and if anybody tries to so I'm happy to drop them, get it working, there's been little if any experiments with ensemble abroad and word level stuff classification, and I do think it should be the Best approach: alright thanks, everybody have a great week and see you next Monday, [Applause,],

21. 02:09:15

- **Lesson 10 conclusion and assignments for Lesson 11**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Lesson 11: Neural Translation

Outline

Today we're going to learn to translate French into English! To do so, we'll learn how to add attention to an LSTM in order to build a sequence to sequence (seq2seq) model. But before we do, we'll do a review of some key RNN foundations, since a solid understanding of those will be critical to understanding the rest of this lesson.

A seq2seq model is one where both the input and the output are sequences, and can be of difference lengths. Translation is a good example of a seq2seq task. Because each translated word can correspond to one or more words that could be anywhere in the source sentence, we learn an attention mechanism to figure out which words to focus on at each time step. We'll also learn about some other tricks to improve seq2seq results, including teacher forcing and bidirectional models.

We finish the lesson by discussing the amazing DeVISE paper, which shows how we can bridge the divide between text and images, using them both in the same model!

Video Timelines and Transcript

1. 00:00:30

- **Tips on using notebooks and reading research papers**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

What a start pointing out a couple of the many cool things that happen this week. One thing that I'm really excited about is we briefly talked about how Leslie Smith has a new paper out and it basically the paper goes text is previous to key papers. Are cyclic or learning rates and super convergence and built on them with a number of experiments to show how how you can achieve super convergence and so super convergence lets you train models five times faster than previous, like kind of stepwise approaches, it's not 5 times faster Than CLR, but it's faster than CLR as well, and the key is that super convergence lets you get up to like massively high learning rates by somewhere between 1 and 3, which is quite amazing, and so the interesting thing about super convergence is that it. You actually train at those very high learning rates for quite a large percentage of your ybox, and during that time the loss doesn't really improve very much. But the trick is like it's doing a lot of searching through the space to find really generalizable areas. It seems so those we kind of had a lot of what we needed in fastai to achieve this, but we're missing a couple of bits, and so silver Google has done an amazing job of fleshing out the pieces that we're missing and then confirming that he Has actually achieved super convergence on training on say, 5 10. So I think this is the first time that this has been done, that I've heard of outside of Leslie Smith himself, and so he's got a great blog post up now on one cycle, which is what Leslie Smith called this approach, and

this is actually it turns Out what one cycle looks like it's: a single cyclic or learning rate, but the key difference here is that the the going up be it is the same length as they're going down right, so you go up like really slowly and then at the end, for like A tenth of the time you, you, then, have this little bit where you go down even further and it's interesting.

Obviously this is a very easy thing to show very easy thing to explain. Sylvia has added at a fastai under the temporarily. It's it's called use. Clr beta by the time you watch this on the video it'll probably be called one cycle - something like that. But you can use this right now. So that's one key piece to getting these massively high learning rates and he shows a number of experiments when you do that. A second key piece is that, as you do this to the learning rate, you do this to the momentum right. So when the loading rates low, it's fine to have a high momentum, but then, when the learning rate gets up really high, your momentum needs to be quite a bit lower. So this is also part of our. What he's added to the library is this cyclic? All momentum, and so with

2. 00:03:15

- **Follow-up on lesson 10 and more word-to-image searches**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

With with these two things you can train for about the fifth of the number of epochs with stepwise learning rate schedule, then you can drop your weight decay down by about two orders of magnitude. You can often remove most or all of your drop out, and so you end up with something that's trained faster and generalizes better, and it actually turns out that silver got quite a bit better accuracy than Leslie Smith's paper. His guess I was pleased to see is because our data augmentation defaults are better than less visas. I hope that's true so check that out another cool thing I just as I say, there's been so many cool things this week, I'm just going to pick two Hamill who seen who's a works at github. I just really like this there's a fairly new project called our cube flow, which is basically 10250 for kubernetes Hamill wrote a very nice article about magical sequence, to sequence, models, building data products on that and using kubernetes to kind of put that in production and so Forth he said that the Google cou flow team created a demo based on what he wrote earlier this year directly based on the skills on moon, fastai, and I will be presenting this technique at KD. Dk t DS one of the top academic conferences, so I wanted to share this as a motivation for folks to blog, which i think is a great point. You know I don't nobody who goes out and writes a blog things that you know. Probably you know.

None of us really think our blog is actually going to be very good, probably nobody's gon na read it you know, and then, when people actually do like it and read it, it's like with great surprise you just got over. That's actually something people were interested to read. So here is the the tool where you can summarize github issues using this tool, which is now hosted by Google on their Cooper org domain. So I think that's a great story of getting you know if Emeril didn't put his work out there, none of this would have happened and yeah you can check out his post. That made it all happen as well. So talking of the magic of sequence to sequence, models, let's build one, so we're going to be specifically working on machine translation. So machine translation is a something that's been around for a long time, but specifically we're going to look at an approach called neural translation, which is using neural networks for translation, and they didn't know that wasn't really a thing in any kind of

meaningful way. Until a couple of years ago, and so thanks to Chris Manning from Stanford for the next three slides 2015 Chris pointed out that neural machine translation kind of first appeared properly and it was pretty crappy compared to the statistical machine. Translation approaches that use kind of classic.

Like feature engineering and standard MLP kind of approaches of lots of stemming and fiddling around this work frequencies and engrams and lots of stuff by a year later, it was better than everything else. This is on a metric called blue, we're not going to discuss the metric, because it's not a very good metric and it's not very interesting, but it's what everybody uses, so that was blue as of when Chris did this slide. As of now, it's about up here, it's about 30, so we're kind of seeing machine translation starting down the path that we saw, starting computer vision, object classification in 2012. I guess which is you know, we kind of just surpassed the state of the art and now we're zipping past it at a great rate. It's very unlikely that anybody watching this is actually gon na build a machine translation model because you can go to

3. 00:07:30

- **Linear algebra cheat sheet for deep learning (student's post on Medium)**
- **& Zero-Shot Learning by Convex Combination of Semantinc Embeddings (arXiv)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Translate, Google calm and use theirs and it works quite well. So why are we learning about machine translation? Well, the reason we're learning about machine translation is that the general idea of taking some kind of input like a sentence in french and transforming it into some other kind of output with arbitrary length such as a sentence in english, is a really useful thing to do. For example, the thing that we just saw that ham all that github did tex github issues and turns them into summaries. Other examples is taking videos and turning them into descriptions or taking a well. I don't know I mean, like you know, basically anything where you're spitting out kind of an arbitrary sized output. Very often that's a sentence, so maybe taking a CT, scan and spitting out a radiology report. You know this is where you can use sequence to sequence, learning. So the important thing about a neural machine translation - these are more slides from Chris and and generally six a sequence of sequence. Models is that you know there's no fussing around with heuristics and Haacke. You know feature engineering, whatever it's end-to-end training we're able to build these distributed representations which are shared by lots of kind of concepts within a single network. We're able to use long term State in the air and ENSO use a lot more context than kind of Engram type approaches and in the end, the text we're generating uses an iron in as well.

So we can build something. That's more fluid we're going to use a bi-directional LS TM, with attention well, actually were going to use a bi-directional giu with attention, but basically the same thing. So you already know about bi-directional: recurrent neural networks and attention we're going to add on top today. These general ideas, you can use or lots of other things as well as Chris points out on this slide. So let's jump into the code which is in the translate notebook, funnily enough and so we're

4. 00:10:00

- **Systematic evaluation of CNN advances on ImageNet (arXiv)**
- **ELU better than RELU, learning rate annealing, different color transformations,**
- **Max pooling vs Average pooling, learning rate & batch size, design patterns.**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Going to try to translate French into English, and so the basic idea is that we're going to try and make this look as much like a standard, neural network approach as possible. So we're going to need three things. You will remember the three things: data, a suitable architecture and a suitable loss function once you've got these three things you run fit and all things going. Well, you end up with something that solves your problem: okay, so data! You know we generally need XY pairs. Okay, because we need something which we can feed it into the loss function and say I took my x-value, which was my French sentence and the loss function says it was meant to generate this English sentence. And then you had your predictions, which you would then compare and see how good it is. Okay. So therefore we need lots of these tuples of french sentences with their equivalent English sentence. That's called a parallel corpus. Obviously, this is harder to find than a corpus for a language model because for a language model we just need text in some language which you can basically all for any living language of which the people that use that language, like use computers. There will be a few gigabytes at least of text floating around the internet, for you to grab okay, so building a language model is only challenging corpus wise or you know, ancient languages. One of our students is trying to do a Sanskrit one.

For example, at the moment, but that's very rarely a problem for translation. There are actually some pretty good. Parallel corpus is available for European languages. The European Parliament basically has every sentence in every European language. Anything that goes through the UN is translated to lots of languages. For French to English, we have a particularly nice thing, which is pretty much any semi official Canadian web site. I will have a French version and an English version, and so this chap kursk, Ellison birch did a cool thing, which is basically to try to transform French URLs into English URLs by like replacing. If I am and hoping that that retrieves the equivalent document and then did that for lots and lots of web sites and into that creating a huge corpus based on millions of web pages, so a French to English, we have this particularly nice resource. So we're going to start out by talking about how to create the data, then we'll look at the architecture and then we'll look at the loss function and so for bounding boxes. All of the interesting stuff was in the loss function, but for New York translation. All of the interesting stuff is going to be in the architecture. Okay, so let's zip through this pretty quickly and one of the things I want you to think about particularly, is what are the relationships, the similarities in terms of the tasks we're doing and how we do it between language, modeling versus euro translation, okay.

So the basic approach here is that we're going to take a sentence, so this case this example is English to German and this slides through Mon Steven emeriti, we steal everything we can from Steven. We start with some sentence in English, and the first step is to do basically the exact same thing: we do in a language model which is to chuck it through an R in it now with our language model. Actually, that's not even think that language model. Let's start even easier: the classification model, okay, so something that turns some. This sentence into positive or negative sentiment. We had a a decoder, you know something which basically took the RNN output and from our paper we grabbed three things. We took a max pull over. All of the time steps we took a mean port over there, all the time steps and we took the value

of the iron in at the last time, step stack all those together and put it through a linear layer. Most people don't do that in most NLP stuff. This is a like. I think it's something we invented people pretty much always use the last time step. So all the stuff we'll be talking about today uses the last time step. So we start out by chucking this sentence through an R and N, and out of it comes some state right, so some state meaning some hidden state, some vector that represents the output of an hour that is encoded. That sentence you'll see the word that Steven used here was encoder.

We've tended to use the word backbone right so like when we've talked about like adding a custom head to an existing model. Like you know, the existing pre-trained imagenet model, for example, we kind of say that's our backbone, and then we stick on top of it. Some-Some head. That does the task. We want, in sequence, to sequence, learning they use the word encoder, but it basically it's the same thing. It's some piece of a neural network architecture that takes the input and turns it into you know some representation which we can then stick a few more layers. On top of to grab something out of it, such as we did for the classifier, where we stuck a linear layer on top over to turn it into a sentiment, positive or negative. So this time, though, we have something that's a little bit harder than just kiding sentiment, which is, I want to turn this state not into a positive or negative sentiment, but into a sequence of tokens where that sequence of tokens is the German. In this case, the German sentence that we want, so this is sounding more like the language model than the classifier, because the language model had multiple tokens for every input word. There was an output word, but the language model was also much easier because the the number of tokens in the language model output was the same length as the number of tokens in the language model input, and not only were there the same length they exactly matched Up it's like after word.

One comes word, two afterward two comes word three and so forth right, but for translating language you don't necessarily know that the word he will be translated as the first word in the output and that love will be. The second word in the output I mean in this particular case. Unfortunately they are the same, but very often you know the subject. Object order will be differential. There will be some extra words inserted or some pronouns we'll need to add some gendered article or whatever. Okay, so this is the key issue we're gon na have to deal with. Is the fact that we have an arbitrary length output where the tokens in the output do not correspond to the same order? You know specific tokens in the input okay, but the general idea is the same: here's an RNN to encode the input turns it into some hidden state, and then this is the new thing. We're going to learn is generating a sequence output. So we already know sequence. Two plus that's IMDB classifier, we already know sequence, two equal length sequence. Where corresponds to the same items: that's the language model, for example, but we don't know yet how to do a general-purpose sequence to sequence. So that's the new thing today. Very little of this will make sense unless you really understand lesson: six, how an RNN works.

Okay. So if some of this lesson doesn't make sense to you - and you find yourself wondering what does he mean by hidden state, exactly how's - that working go back and re-watch lesson six, to give you a very quick review, we learnt that and are an N at its Heart is a standard fully connected Network, so here's one with one two three four layers right takes an input and puts it through four layers, but then at the second layer it can just concatenate in the second important third layer concatenate in a third input, but we Actually wrote this in Python as just literally a four layer, neural network, okay, there's nothing else. We used other than linear layers and values. We used the same weight matrix every time an input came in, we used the same matrix every time. We went from one of these states to the next and that's why there's

errors of the same color? And so we can redraw that previous thing like this yeah, and so we not only did we redraw it, but we took the you know four lines of linear, linear, linear linear code in pytorch and we replaced it with a for loop. Okay, so remember we had something that did exactly the same thing as this, but it just had four lines of code: saying linear, linear, linear layer, and we really literally replaced it with a for loop because that's nice to refactor.

So, like literally, that refactoring, which doesn't change any of the math any of the ideas and if the outputs that refactoring is entirely okay, turning a bunch of separate lines in the code into a Python corner, okay, and so that's how we can draw it, we could Take the output so that it's not outside the loop and put it inside the loop like so right, and if we do that, we're now going to generate a separate output for every input. So in this case this particular one here, the hidden state gets replaced. Each time - and we end up just spitting out the final hidden State, so this one is this example: okay, but if, instead we had something that said, you know h's dot, append, h and returned H's at the end. That would be this picture yeah and so go back and relook at that notebook. If this is unclear, I think the main thing to remember is when we say hidden state we're referring to a vector. Okay, see you here's the vector right H, equals torch, zeroes, okay and hidden. Now, of course, it's a vector for each thing in the mini bash, so it's it's a matrix, but I'm generally, when I speak about these things, I ignore them in each piece and treat it for just a single item. Okay, so it's just a vector of this length. We also learned that you can stack these layers on top of each other, so rather than this first errand ends bidding our output.

There could just spit out inputs into a second area and, if you're thinking at this point, I think I understand this, but I'm not quite sure if you're anything like that me. That means you don't understand this right and the only way you know and that you actually understand it is to go and write this in from scratch in pytorch or an umpire okay. And if you can't do that, then you know: okay, you don't understand it and you can go back and re-watch the lesson 6 and check out the notebook and copy some of the ideas until you can it's really important that you can write that from scratch. It's less than a screen of code. Okay, so you want to make sure you can create a two layer, iron, okay - and this is what it looks like if you unroll it. Okay, so that's the goal is to get to a point that we first of all, have these XY pairs of sentences and we're going to do French to English. So we're going to start by downloading this data set and training a translation model takes a long time. Google's translation model has eight layers of RNN stacked on top of each other, there's no conceptual difference between eight layers and two layers. It's just like if you're google and you have more GPUs GPUs and you know what to do with then you're fine doing that. Where else, in our case, it's pretty likely that the kind of sequence to sequence models we're building are not going to require that level of computation so to keep things simple, let's do a cut-down thing where, rather than learning, how to translate French into English for any Sentence: let's learn to translate French questions into English questions: okay and specifically, questions that start with what, where which, when okay, so you can see here, I've got a regex.

It looks for things at start of WH and in with a question mark, so I just go through the corpus open up each of the two files. Each line is one parallel text written together. I grab the English question the French question and check whether they match the regular submissions. Okay, dump, that out was the pickle, and so they don't have to do it again, and so we now have 52,000 sentences, and here are some examples with us. Well, sentence pairs, and here are some examples of those one nice thing about this - is that what who, where clap questions, tend to be fairly short, which is nice, but I would say the idea that we could learn from scratch with no previous understanding of the idea Of language, let alone of English or a

French that we could create something that can translate one to the other for any arbitrary question, with only 50,000 sentences. It sounds like a ludus cursory, difficult thing to ask us to do right, so I will be impressed if we can make any progress whatsoever. This is very little data to do a very complex exercise correct. So this contains the tuples of French and English. You can use this handy idiom just pick them apart into a list of English questions and list of french questions, and then we tokenize the English question questions and we tokenize the French questions. Okay, so remember that just means splitting them up into separate words or word like things by default.

The tokenizer that we have here and remember this is a wrapper around the Spacey tokenizer, which is a fantastic tokenizer. This wrapper by default assumes English okay. So to ask for French, you just add an extra parameter. The first time you do this you'll get an error, saying that you don't have the space you French model installed and you can Google to get the command something Python. Ms BAE see download French or something like that to grab the French model. Okay, I don't think any of your going to have RAM problems here, because this is not particularly big corpus, but I know that some of you we're trying to train new language models during the week and we're having RAM problems. If you do it's worth knowing what these functions are actually doing so, for example, these ones here is processing every sentence across multiple processes as well. The NP means and remember you, know Farsi. Our code is designed to be pretty easy to read, so you know three or four lines of code, so here's the three lines of code to process or MP find out how many CPUs you have divided by two, because normally with hyper-threading, they don't actually all work In parallel, then, in in parallel run, this process function, so that's going to spit out a whole separate Python process for every CPU. You have. If you have a lot of cause, that's a lot of Python processes, everyone's going to load the whole.

You know all this data in and that can potentially use up all your ramped, so you could replace that with just proc all rather than pro or MP to use less RAM or you could just pros use less cause. So but you know at the moment we were calling this function partition by cause which calls partition on a list and asks to split it into a number of equal length things according to how many CPUs you have so you could replace that you're, not splitting it Into a smaller list and read it on less things:

5. 00:27:15

- **Data Science Bowl 2017 (Cancer Diagnosis) on Kaggle**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Yes, Rachel was an intention layer tried in the language model. Do you think it would be a good idea to try and add one we haven't had learned about attention yet so, let's ask about things that we have cut to not things we haven't. The short answer is no, I haven't tried it properly. Yes, you should try it because it might help okay and you know in general, there's going to be a lot of things that we covered a day which, if you've done some sequence to sequence stuff before you're, want to know about something we haven't covered. Yet I'm going to cover all the sequence of sequence, things. Ok! So at the end of this, if I haven't covered the thing you wanted to know about, please ask me: then: if you ask me before I'll, be answering something based on something that I'm about to teach you, ok, so having tokenized the english and french, you can See how it gets bit out and you can see the

tokenization for french is quite different, looking because french loves their apostrophes and their hi friends and stuff right. So if you try to use in english tokenizer for a french sentence, you're going to get it pretty. Crappy, ok, so, like I don't find you need to know heaps of NLP ideas to use deep learning for NLP, but just some basic stuff, like you know, use the right. Tokenize of your language is important, and so some of the students this week in our study group have been trying to work, build language models for Chinese instance, which of course doesn't really have the concept of a tokenizer.

In the same way, so we've been starting to look at it briefly mentioned last week. This google thing called sentence: peace which basically splits things into arbitrary sub word units, and so, when I say tokenize, if you're, using a language that doesn't have spaces in, you should probably be checking out sentence. Peace or some other similar sub word unit thing instead and hopefully in the next week or two we'll be able to report back with some early results of these experiments with Chinese. Okay, so have you tokenized to it, will save that disk and then remember. The next step after we create tokens is to turn them into numbers and turn them into numbers. We have two steps. The first is to get a list of all of the words that appear and then we turn every word into the index into that list right. If there are more than 40,000 words that appear, then let's cut it off there, so it doesn't go too crazy and we insert a few extra tokens for beginning of stream, padding end of stream and okay. So if we try to look up something that wasn't in the 40,000 most common, then we use a default dicked to return three, which is unknown so now we can go ahead and turn every token into an ID by putting it through the string to integer dictionary. We just created, and then at the end of that, let's add the number two which is industry and you'll, see like this kind. The code you see here is the code.

I write when I'm iterating in experimenting right because, like 99 % of the code I write when I'm either writing experimenting. It turns out to be totally wrong or stupid or embarrass seeing and you don't get to see it right. But like there's no point, you know refactoring that and making it beautiful when I'm riding it's kind of wanting you to see all the little shortcuts I have so like, rather than doing this properly and actually find you know having some constant or something for end of Stream marker and using it when I'm prototyping, I just do the easy stuff. You know I mean not not so much that I end up with broken code. You know, but I I don't you know, try to find. I try to find some middle ground between beautiful code, and you know code that was just heard him mention that we divide number of CPUs by two, because with hyper-threading we don't get a speed-up using all the hyper threaded cores. Is this based on practical experience or is there some underlying reason why we wouldn't get additional speed-up yeah? It's just practical experience and it's like a lot of things kind of seem like this, but I definitely noticed with tokenization hyper threading seem to slow things down a little bit. Also, if I use all the cause you know like often, I want to do something else. At the same time, like generally run some interactive notebook, and I don't have any spare room to do that.

It's a minor issue: yeah, okay, so now for our English and our French, we can grab our list of IDs and when we do that, of course, we need to make sure that we also store the vocabulary. There's no point having IDs. If we don't know like what the number five represents, there's no point having a number five. So that's our vocabulary. It's a string and the reverse mapping string to int that we can use to convert more courses in the future. Okay, so just to confirm it's working. We can go through each ID, convert the into a string and spit that out and there we have our thing back now with an industry marker. At the end, English vocab is 17,000. Our French vocab is 25,000, so you know there's not too big. You know there's not too complex a

vocab that we're dealing with, which is nice to know okay, so we spent a lot of time on the forums during the week discussing how pointless what vectors are and how you should stop getting so excited about them and we're Now going to use them, why is that, basically, all the stuff we've been learning about using language models and pre-trained proper models rather than pre-trained? You know linear single layers, which is what word vectors are, I think, applies equally well to sequence, to sequence, but I haven't tried it yet yet so Sebastian - and I are you know, starting to look at that and slightly distracted by preparing this class at the moment. But after this class is done so there's a whole thing for anybody interested in creating some genuinely new like highly publishable results. The entire area of sequence, to sequence, with pre-trained language models hasn't been touched yet, and I strongly believe is going to be just as good as as classification style and if you you know, work on this and you get to the point where you have something.

That's looking exciting and you want help publishing it. You know I'm very happy to help co-author papers. You know that on stuff, that's looking good, you know so feel free to reach out. If and when you have some interesting results. So at this stage we don't have any of that, so we're going to use you know very little fast. I I actually and very little in terms of kind of fastai ideas. So we you know, all we've got is word vectors anyway, so, let's at least use decent word vectors. So word, Tyvek is very old. Where vectors there are better word vectors now and fast text is a pretty good source of word vectors. There's hundreds of languages available for them your language is likely to be represented so to grab them. You can click on this link download word vectors for a language that you're interested in install install the fast text, Python library, it's not available on pi PI, but here's a handy trick if there is a github repo that has like a setup PI inert and a Requirements text in it you can just chuck get Plus at the start and then stick that in your pip install and it works, like hardly anybody, seems to know this and like it never even like. If you go to the fast text repo, they won't tell you this they'll say you have to download it and CD into it and blah blah blah, but you're done. You can just run that ok, which you can also use for the fastai library by the way, if you want to pip install the latest version of past pay.

I even totally do this, so you grab the library import, it load the model. So here's my English model and here's, my French you'll see there's a text version and a binary version. The binary versions a bit faster, we're going to use that the text version is also a bit buggy and then I'm going to convert it into a standard Python dictionary to make it a bit easier to work with. So this is just going to grow through each word, with a dictionary, comprehension and save it as a pickle dictionary. Ok, so now we've got our pickle dictionary. We can go ahead and look up a word, for example, come up and that will return a vector. The length of that vector is the dimensionality of this set of word vectors. So in this case, we've got three hundred dimensional English and French words: okay,

6. 00:36:30

- **DSB 2017: full preprocessing tutorial, + others.**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

For reasons that you'll see in a moment, I also want to find out what the mean of my vectors are and the standard deviation of my vectors are so the means about 0 and the standard deviation is about point three. So remember that often corpuses have a pretty long, tailed

distribution of sequence length and it's the longest sequences that kind of tend to overwhelm how long things take, and you know how much memory is used and stuff like that, so I'm going to grab you know in This case, the 99th to 97th percentile of the English and French and truncate them to that amount. Originally I was using the 90th percentile, so these are poorly named variable, so apologies for that, okay, so that's just truncating them so we're nearly there. We've got our tokenized numeric alized, English and French data set. We've got some word vectors, so now we need to get it ready for play torch, so app a torch, expects a data set object, and hopefully, by now you all can tell me that a data set object requires two things: a length and an indexer. So I started route writing this now. It's like okay, I need a sector sector dataset and I started out. Writing it and I thought okay, we're going to have to pass it out. X's now wise and store them away, and then my indexer is going to need to return a numpy array of the X's at that point in an umpire row of the Y's at that point, and oh that's it so then after I wrote this, I realized I Haven't really written a sector sector dataset I've just written a totally generic data set, so here's like the simplest possible data set that works for any pair of arrays.

So it's now poorly named it's much more general than a sector sick data set. But that's what I needed it for this a function. Remember: we've got V variables, teeth, tensors a4 arrays. So this basically goes through each of the things you pass it. If it's not already in numpy array, it converts it into a numpy array and returns back at a pool of all of the things that you passed it which are now guaranteed to be mud pirates. So that's a VT 3 very handy little functions. Ok, so that's it! That's our data set. So now we need to grab our English and French IDs and get a training set and a validation set. And so one of the things which is pretty disappointing about a lot of code out there on the Internet is that they don't follow us and simple best practices. For example, if you go to the Play Torche website, they have an example section for sequence, a sequence, translation there example does not have a separate validation set. I tried it training according to their settings and I tested it with their validation set. It turned out that it overfit massively, so this is not just a theoretical problem: the actual pipe torch repo has the actual official sequence, the sequence translation example, which does not check for overfitting and overfits horribly. Also, it fails to use mini batches, so it actually fails to utilize any of the efficiency of pytorch whatsoever.

So there's a lot of like even if you find code in the official pipe torch repo, don't assume it's any good at all right. The other thing you'll notice, is that everybody were when they like pretty much every other sequence. The sequence model I've found in Peyer torch anywhere on the internet, has clearly copied from that shitty pipe or epoch isn't all the same variable names. It has the same problems. It has the same mistakes like another example: nearly every pytorch, convolutional neural network I found, does not use an adaptive pooling layer, so in other words, the final layer is always like average paul, 7. Comma 7 right. So they assume that the previous layer is 7 by 7 and if you use any other size input, you get an exception and therefore nearly everybody. I've spoken to that uses, Piatt torch thinks that there is a fundamental limitation of CN NS, that they are tied to the input size, and that has not been true since vgg right. So every time we grab a new model and stick it in the first day. I repo i have to go in search for paul and add adaptive to the start and replace the 7 with a 1, and now it works on any sized object. All right so just be careful, you know it's still early days and and believe it or not. Even though most of you have only started in the last year, your deep learning journey, you know quite a lot more about a lot of the more important practical aspects and the vast majority of people that are like publishing and writing stuff in official repos and stuff.

So you kind of need to have a little more self-confidence than you might expect when it comes to reading other people's code. If you find yourself thinking that looks odd, it's not necessarily you, it might well be then. Okay, so yeah, I would say, like at least 90 % of deep learning code that I start looking at turns out to have, like you know, like deathly, serious problems that make it completely unusable for anything. And so I kind of been telling people that I've been working with recently. You know if, if the repo you're looking at doesn't have a section on it saying here's the test we did where we got the same results as the paper that suspend to be implementing. That almost certainly means they haven't, got the same results in the paper. They're implementing they probably haven't even checked. Okay, and if you run it, it definitely won't get those results, because it's it's hard to get things right. The first time it takes me 12 goes, you know, probably takes the normal, smarter people than me. Six goes, but if they haven't tested it once, it's almost certainly won't work. Okay, so there's our sequence sequence data set, let's get the training and validation, sets here's an easy way to do that. We have a bunch of random numbers. One feature of your data see if they're bigger than 0.1 or not, that gives you a list of balls index into your array with that list of balls to grab a training set index into that array, with the opposite of that list of balls.

To get your validation set, there's a nice easy way, there's lots of ways of doing it. I just like to do different ways to you: can see a few approaches. Okay, so now we can create our data set with our X's in our Y's, French and English. If you want to translate instead, English to French switch these two around and you're done. Okay, now we need to create data loaders. We can just grab our data loader and pass in our data set and match size. We actually have to transpose the arrays, I'm not going to go into the details about why we can talk about it during the week if you're interested, but have a think about why we might need to transpose their orientation. But there's a few more things. I want to do one is that since we've already done all the pre-processing, there's no point spawning off multiple workers to do like augmentation or whatever, because there know what to do so, making them workers equals. One will save you some time. We have to tell it what our padding index is. That's actually pretty important, because what's going to happen is that we've got different length sentences and faster. I I think it's pretty much. The only baby that does this past day. I will just automatically stick them together and pad the shorter ones to be so that they're all in that equal length because remember a tense, has to be rectangular. Okay in the decoder in particularly, I actually want my padding to be at the end, not at the start like for a classifier.

I want the padding at the start, because I want that final token to represent the last word of the movie review, but in the decoder as you'll see, it actually is going to work out a bit better to have the padding at the end. So I say: prepaired equals false and then, finally, since we've got sentences of different lengths coming in and they all have to be put together in a mini batch to be the same size by padding, we would much prefer that the sentence in a mini batch are Of similar sizes already, because otherwise that it's going to be as long as the longest sentence and that's going to end up wasting time and memory. Okay, so therefore, I'm going to use the sample of tricks that we learnt last time, which is the validation, set. We're going to ask it to sort everything by length, first, okay and then for the training set, we're going to ask it to randomize the order of things, but to roughly make it so that things of similar length are about in the same spot. Okay, that's how we got our sort sample and a sort, assembler, okay, and then at that point we can create a model. Data object. Remember a model data object really does one thing which is it says I have a trading set and a validation set and an optional test set and sticks them into a single object. We also have a path so that it has somewhere to store temporary files models. Stuff like that right, so you know we're doing we're not using fastai for very much at all.

In this example, just kind of a minimal set to show you like you know how, to you know kind of get your model data objects in the end, once you've got a model data object, you can then create a learner, and you can then call fit okay. So that's kind of like minimal amount of faster, faster. I stuff here. This is a standard height watch compatible data set. This is a standard plate or compatible data loader behind the scenes, it's actually using the class AI version, because I do need to do this automatic padding for convenience. So there's a few tweaks in our version that are a bit faster, a bit more convenient the faster ice samplers we're using. But you know, there's not too much going on here so now, we've got our model data object. We can basically tick off number one. Okay, so, as I said, most of the work is in the architecture, and so the architecture is going to take our sequence of tokens. Okay, it's going to spit them in to a encoder or you know, in kind of computer vision, turns what we've been calling a backbone. You know something that's going to try and turn this into some kind of representation. So that's just going to be an iron in okay, that's going to spit out the final hidden state which for each sentence it's just a vector single okay, and so that's all going to take that's none of this is going to be.

Do that's all going to be using very direct, simple techniques that we've already learnt and then we're going to take that and we're going to spread it into a different era, tan, which is a decoder and that's going to have some new stuff. Because we need something that can go through one word at a time: okay and it's got to keep going until it thinks it's finished, the sentence it doesn't know how long the sentence is going to be ahead of time keeps going until it thinks it's finished. The sentence and then it stops and returns a sentence. Okay. So let's start with the with the end coder. So in terms of variable naming here, there's basically identical variable for encoder and decoder buttes for encoder and decoder, the encoder versions.

7. 00:48:30

- **A non-deep-learning approach to find lung nodules (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Have Inc the decoder versions have dead, okay, so for the N coder, here's our embeddings and so like. I always try to mention like what the mnemonics are. You know, rather than writing things out. You know so in turn longhand. So you know just remember: anchors and encoder deckers of decoder and visit embedding. The final thing that comes out is out. The r and n in this case is a gr. U not an LST M they're nearly the same thing. So don't worry about the difference. You could replace it with an LST m and you'll get basically the same results to replace it with an LST M, simply type OST M okay. So we need to create an embedding layer to take because remember what we're being passed is the index of the words into a vocabulary, and we want to grab there fast text embedding and then over time. We might want to also fine tune to train that embedding n to it so to create an embedding we'll call create embedding up here so we'll just say: n n dot. Embedding. So it's important that you know now how to set the rows and columns for your embedding. So the number of rows has to be equal to your vocabulary size, so each vocab riordan has a word vector and the. How big is your embedding well in this case it was determined by fast text and the first text embedding size 300. So we have to use size 300 as well, otherwise we can't start out by using their betting's okay now.

So what we want to do is this is initially going to give us a random set of embeddings, and so we're going to now go through each one of these and if we find it in fast text, we'll replace it with the first text admitting okay. So again, something that you should already know is that a pytorch module that is learn about has a weight attribute and the weight attribute is a variable and that variables have a data attribute, and the data attribute is a tensor. Now your notice very often today, I'm saying here, is something you should know not so that you think. Oh, I don't know that I'm a bad person right, but so that you think okay. This is a concept that you know I haven't learnt yet and Jeremy thinks I ought to know about, and so I've got to write that down and I'm gon na go home and I'm gon na like Google because, like this is a normal pie, Torche attribute in Every single learning platform module this is a normal plate or attribute in every single pipe torch variable. And so, if you don't know how to graph the weights out of a module or you don't know how to grab the tensor out of a variable, it's gon na be hard for you to build new things or to debug things or maintain things or whatever. Yes, so if I say you ought to know this and you're thinking, I don't know this don't run away and hide, go home and learn the thing and, if you're having trouble learning the thing, because you can't find documentation about it or you don't understand that documentation Or you don't know why Jeremy thought it was important.

You know it jump on the forum and say like please explain this thing, here's my best understanding of that thing as I have it at the moment. Here's the resources. I grew up that helped film you, okay and normally. If I respond it's very likely, I will not tell you the answer, but I will instead give you something, a problem that you could solve that if you solver will solve it for you, because I know that that that way, it'll be something you remember. Okay, so again don't be put off. If I'm like. Okay, you like go read this link. Try and summarize that thing tell us what you think, like I'm trying to be helpful, not unhelpful, and if you're still not following just come back and say like I had to look honestly that link you sent. I don't know what I knew. That means I wouldn't know where to start whatever, like I'll, keep trying to help you until you fully understand it. Okay, so so now that we've got our weight tensor, we can just go through our vocabulary and we can look at the word in our pre-trained vectors and if we find it, we will replace the random weights with that pre-trade vector the random weights. Have a standard deviation of 1, our pre-trained vectors, it turned out how to standard deviation of about 0.3. So again, this is the kind

8. 00:53:00

- **Clustering (and why Jeremy wasn't a fan before)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Of hacky thing I do when I'm prototyping stuff, I just multiply two by three. Okay, obviously, by the time you you know see, the video of this mean way, we'll have put all this sequence to sequence, stuff into the faster library. You won't find horrible paths like that and there sure hope, but hack away when you're prototyping some things won't be in fast text, in which case we'll just keep track of it, and I've just added this print statement here, just so that I can kind of see. What's going like, why am I missing stuff? Basically, when you know I'll probably comment it out when I actually commit this to github? That's why that's there? Okay, so we create those embeddings and so when we actually create the sequence to sequence, RNN it'll print out how many or missed - and so remember we had like about 30,000 words, so we're not missing too many and interesting the things that are missing well,

there's our Special toque there's a special token for uppercase, not surprising. That's missing. Also, remember it's not token Tyvek, it's not token text it's. You know it does words. So L, apostrophe and D apostrophe and apostrophe s they're not appearing either. So that's interesting. That does suggest that. Maybe we could have slightly better embeddings if we try to find some which would being tokenized the same way.

We tokenize that's okay, Rachel. Do we just keep embedding vectors from training? Why don't we keep all word embeddings in case you have new words in the test set? Oh, I mean we're gon na be fine-tuning them, you know and so yeah I don't know. I mean it's an interesting idea. Maybe that would work. I haven't tried it. I mean, obviously you wouldn't so. Can you use that? I asked the question so you can also add random embedding to those and the beginning. Just keep them random on, but you're gon na have bigs gon na make it an effect in the sense that you are going to be using those words yeah yep. I think it's an interesting line of inquiry, but I will say this: the vast majority of the time when you're kind of doing this in the real world, your vocabulary will be bigger than 40-thousand and once your vocab areas, bigger than 40-thousand using the standard techniques you The embedding layers gets so big that it takes up all your memory. It takes up all of the time and the backdrop there are tricks to dealing with very large vocab Riis. I don't think we'll have time to handle them in this session, but you definitely would not want to have all three and a half million fast text vectors in a in an embedding layer. So I I wonder so, if you're not touching a word, it's not gon na change. Right like you and we are fine-tuning right.

You are not it's in GPU Ram and you're gon na remember three and a half million times three hundred times the size of a single precision floating point vector plus all of the gradients for them. Even if it's not touched like they're, without being very careful and adding a lot more code and stuff, it is slow and hard and when we wouldn't touch it for now, but as I say, I think, there's an interesting path of inquiry. But it's the kind of path of inquiry that leads to like multiple academic papers. Not you know something that you do on a weekend, but I think would be very interesting for ya. Maybe we can look at at some time. You know, and as I say, I have actually started doing some stuff around incorporating large for cavalry handing handling into fastai. It's not finished, but hopefully by the time we get here. This kind of stuff would be possible. Ok, so we create our encoder. Embedding add a bit of drop out, ok and then we create our an end. This input to the RNN, obviously is the size of the embedding by Bodie by definition, number of hidden is whatever we watch, so we set up to 256 for now. However many layers we want hands and drop out inside the aren't as well. Okay, so this is all standard height watch stuff. You could use ur LS TM here as well and then finally, we need to turn that into some output that we're going to feed to the decoder. So, let's use a linear layer to convert the number of hidden into the decoder embedding sites.

Okay, so in the forward pass, here's how that's used! We first of all initialize our hidden state to a bunch of zeros okay. So we've now had a vector of zeros, which we and then we're going to take our input and put it through our embedding we're going to put that through drop out. We then pass our currently zeros hidden state and our embeddings into our R and N, and it's going to spit out the usual stuff that I ran in spit up, which includes the final hidden state. We're then going to take that final hidden stake and stick it through that linear layer. So we now have something of the right size to feature our decoder. Okay, so that's! That's it and again, this is like ought to be very familiar and very comfortable. It's like the most simple possible iron n. So if it's not go back check out lesson, six make sure you can write it from scratch and that you understand what it does. But the key thing to know is that it takes our inputs and spits out a hidden vector that hopefully will learn to contain all of the information

about what that sentence says and how it says it, because if it can't do that, all right, if we can't Do that, then we can't feed it into a decoder and hope it to spit out our sentence in a different language. So that's what we want it to learn to do and we're not going to do anything special to make it learn to do that. We're just gon na, do you know the three things and cross our fingers, because that's what we do all right so that's H is.

Is that yes, all right, so it's a hidden stick. I guess Stephen used s for state. I used HB hidden, but there you go. You would think the two Australians could agree on something like that, but apparently not so. How do we now do the new bit right, and so the basic idea of the new bit is the same. We're going to do exactly the same thing, but we're going to write our own forward. Okay, and so the full loop is going to do exactly what the for loop inside pytorch does here, but we're going to do it manually right, so we've got to go through the forward and how big is the for loop? It's a output sequence length! Well, what is output sequence length? That's something that passed to the constructor and it is equal to the length of the largest English sentence. Okay, so we're going to do this for loop as long as the largest English sentence, because we're translating into English right. So we can't possibly be longer than that, at least not in this corpus. If we then used it on some different corpus that was longer, this is going to fail. So, but you could make this, you know you could always pass in a different parameter. Of course, all right, so the basic idea is the same: we're going to go through and put it through the in the embedding we're going to stick it through the R and n we're going to stick it through drop out and we're going to stick it through A linear layer all right, so the basic four steps are the same and once we've done that, we're then going to append that output to a list fat and then, when we're going to finish we're going to stack that list up into a single tensor and return. It okay, that's the basic idea: normally a recurrent neural network here is our decoder.

Current neural network, recurrent neural network works on a whole sequence at a time, but we've got a for loop to go through each part of the sequence separately. So we have to add a leading unit access to the start. To basically say this is a sequence of length: 1. Okay, so we're not really taking advantage of the recurrent net. Much at all. We could easily rewrite this with a linear layer. Actually, that would be an interesting experiment if you wanted to try it, so we basically take our input and we feed it into our embedding, and we add something to the front saying treat this as a sequence of length 1 and then we pass that to our M we then get the output of that RNN feed it into our dropout and 15:20 me a layer, so there's two extra things to know that be aware of. Well, I guess it's really one thing. The one thing is what's this: what is the input to that? Embedding, okay - and the answer is it's the previous word that we translated see how the input here is the previous word here. The input here is the previous word here. So the basic idea is, if you're, trying to translate if you're about to translate. You know tell me the fourth word of the new sentence, but you don't know what the third word you just said was that's going to be really hard all right, so we're gon na feed that in at each time, step let's make it as easy as possible.

Okay, and so what was the previous word at the start for the Rawls month? Okay, so specifically, we're gon na start out with a beginning of stream, took it okay, so the beginning of stream token is a zero. So, let's start out our decoder with a beginning of stream token, which is zero, okay and, of course, we're doing a mini batch. So we need batch size number of them. But let's just think about one part, so we've got we start out with a zero. We look at that zero in our in our embedding matrix to find out what the vector for the beginning a stream token. Is we stick a unit axis on the front to say we have a single sequence length of beginning of stream token. We stick that through our RM, which gets not only the fact that

there's a zero is beginning of stream, but also the hidden state which, at this point is whatever came out of our encoder okay. So this now its job is to try and figure out what it's the first word. Okay, what's the first word to translate the sentence pop through some drop out go through one linear layer in order to convert that into the correct size for our decoder embedding matrix? Okay, append that to our list of translated words, and now we need to figure out what word that was because we need to feed it to the next time step. We need to feed it to the next time step. Okay, so remember what we actually output here and and look at use a debugger right, pdb, dot set trace, put it here. What is at P now P is a tensor.

How big is the tensor so before you look it up in the debugger, try and figure it out from first principles and check you're right, so our P is a tensor whose length is equal to the number of words in our English vocabulary, and it contains the Probability for every one of those words that it is that word makes sense right. So then, if we now say at P dot max that looks in its tensor to find out, which word has the highest probability: okay and max imply Torche returns two things. The first thing is: what is that max probability, and the second is: what is the index into the array of that mass probability, and so we want that. Second item index number, one, which is the word index with the largest thing. Okay, so now that contains the word well, the word index into a VOC, a pre of the word. If it's a one right, you might remember one was padding, then that means we're done right. That means we've reached the end because we finished with a bunch of padding okay. If it's not one, let's go back and continue, but now Dec M is whatever the highest probability. Word was, but so we keep looping through either until we get to the largest length of a sentence or until everything in our mini batch is padding and each time we've appended our outputs each time, not not the word with the probabilities.

Okay to this list, which we stack up into a tensor, and we can now go ahead and feed that to a loss function so before we go to a break since we've done one and two: let's do three, which is the last function. The last function is categorical: cross-entropy loss. Okay, we've got a list of probabilities for each of our classes that the classes are all the words in our English vocab and we have a target which is which is the correct class ie, which is the correct word at this location. There's two tweaks, which is why we need to write our own little loss function, but you can see basically it's going to be cross interview. That's right and the tweaks are as follows: tweak number one is, we might have stopped a little bit early right and so the sequence length that we generated may be different to the sequence length of the target, in which case we need to add some padding right Height or padding function is weird if you have a rank: three cancer

9. 01:08:00

- **Using Pytorch with GPU for 'meanshift' (clustering cont.)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Which we do, we have batch size by sorry, we have sequence length by batch size by number of words in the vocab Iraq. Three tensor requires a six tupple each pair, and things in that topple is the padding before and the padding after that. That dimension all right, so in this case the first dimension has no padding. The second dimension has no padding. The third dimension has no padding on the left and as much padding is required on the right okay. So it's good to know how to use that function. Now that we've added any padding that's

necessary, the only other thing we need to do is cross. Entropy loss expects a rank. Two tensor I mentioned matrix, but we've got sequence length by batch size. So, let's just flatten out the sequence length and batch size into a that's. What that minus one in view does okay, so flatten out that for both of them and now we can go ahead and call cross-entropy, that's it! So now we can just use the standard approach: here's our sequence of sequence, R and n. That's this one yeah! So that is a standard! Pipe watch module! Stick it on the GPU. Hopefully, by now you've noticed you, you can call CUDA, but if you call to GPU, then it doesn't put it on the GPU. If you don't have one you can also set faster. I don't use GPU to false to force it to not use the GPU, and that can be super handy for debugging.

We then need something that tells it how to handle learning rates, learning rate groups, so there's a thing called single model that you can pass it to which treats the whole thing as a single learning rate group. So this is like this easiest way to turn a height watch module into a fastai model. Here's the model data object we created before. We could then just call lerner to turn that into a learner, but if we call our NN learner, our NN learner is a learner. It defines cross-entropy as the default criteria. This case we're overriding that anyway. So that's not what we care about, but it does add in these save encoder and load encoder things that can be any sometimes so we could have in this case we're really pretty to set learner, but Aaron Lerner also works. Ok, so here's how we turn our height watch module into a fast AO model into a and once we have a learner, give it our new loss function and then we can call LR find and we can call Fitch and it runs through awhile and we can Save it, you know the normal learn stuff now works. The remember the model attribute of a learner is a standard piped watch model. So we can pass that sum X, which we can grab out of our validation set or you could use lo, not predict array or whatever you like. If you get some predictions and then we can convert those predictions into words by grabbing going dot max 1 to grab the index of the highest probability, words to get some predictions and then we can go through a few examples and print out the French, the correct English and the predicted English for things that are not padding - and here we go alright, so amazingly enough, this kind of like simplest possible, written largely from scratch, pytorch module and only fifty thousand sentences is sometimes capable on a validation set of giving you exactly the Right answer: sometimes the right answer in slightly different wording and sometimes sentences that really are grammatically sensible or even have too many question marks.

So we're we're well on the right track. I think you would agree so you know even the simplest possible sector, SiC trained for a very small number of epochs without any you know, pre-training other than the use of word. Embeddings, surprisingly good. So I think you know the message here and we're going to improve this for the moment after the break, but I think the message here is even sequence to sequence models. You think, as simple of them to possibly work, even with less data than you think you could learn from can be surprisingly and in certain situations this main, even you know, be enough for your needs, so we're going to learn a few tricks after the break, which Will make this much better? So let's come back at 750, so one question that came up during the break is that some of the tokens that are missing in fast text like had a a curly quote rather than a straight quote, for example, and the question was: would it help to normalize Punctuation and the answer for this particular case is probably yes: the difference between curly quotes and straight quotes is rarely semantics. You do have to be very careful, though, because, like it may turn out that people using beautiful curly quotes like using more formal language and they're. Actually, writing in a different way.

So I generally, you know if you're going to do some kind of pre-processing like punctuation normalization, you should definitely check your results with and without because, like nearly

always that kind of pre-processing makes things worse, even when I'm sure it won't hello. What would be some ways over what I seen these sequence of sequence besides through power and weight? Again, let me think about that during the week yeah. It's like you, know a wdl STM, which we've been relying on a lot. Has so many great I mean it's. It's all drop out: well, not all drop out those drop out of many different kinds, there's and then there's the we haven't talked about it much, but there's also a kind of a regularization based on activations and stuff like that as well and on changes and whatever I just haven't seen anybody put in a thing like that amount of work into regularization of sequence, to sequence models and I think, there's a huge opportunity for somebody to do like the AWD LS TM of sectors ik, which might be as simple as dealing all the Ideas from a top UDS, VLS TM and using them directly in Sector SEC. That would be pretty easy to try. I think, and there's been an interesting paper that actually Steven Rarity's added in the last couple of weeks where he used an idea, which I don't know if he stole it from me, but it was certainly something I had also recently done and talked about on Twitter. Either way, I'm thrilled that he's done it, which was to take all of those different AWD, LS, TM, hyper parameters and train a bunch of different models.

And then I use a random forest to find out with feature importance, which ones actually met at the most and then figure out like how to set them yeah. So I think you could totally you know, use this approach to figure out. You know, for sequence, the sequence, regularization approaches which one's the best and optimize them, and that would be amazing yeah. But at the moment I think you know I. I don't know that there are additional ideas to sequence, the sequence regularization that I can think of beyond. What's in that paper for regular language models, stuff and probably all those same approaches would would work. Okay, so tricks trick number one go bi-directional: okay, so for classification, my approach to bi-directional that I suggested you use is take all of your token sequences, spin them around and train a new language model and train a new traffic classifier, and I also mentioned the wiki Text pre train model, if you replace FWD with vwd in the name, you'll get the pre-trained backward model I created for you, okay, so you can use that get a set of predictions and then average the predictions just like a normal ensemble, okay and that's kind of How we do Baidu for that kind of classification there maybe ways to do it end to end, but I haven't quite figured them out yet they're, not in fastai yet, and I don't think anybody's written a paper about them yet so, if you figure it out, That's an interesting line of research, but because we're not doing you know massive documents where we have to kind of chunk it into separate bits and then pall over them and whatever we can do by do very easily.

In this case, which is literally as simple as adding bidirectional equals true to our encoder, people tend not to do bi-directional for the decoder. I think, partly because it's kind of considered cheating, but I don't know like I was just talking to somebody to break about it. There, you know, maybe it can work in some situations, although it might need to be more of a ensemble approach in the decoder, because you kind of it's a bit less obvious anyway, and the ink into the encoder. It's very very simple: bi-directional equals true, and we now have with bi-directional equals true, rather than just having an RNN which is going this direction. We have a second RNN, that's going in this direction and so that second RNN literally is visiting them each token in the opposing order. So when we get the final here in state, it's here rather than here right, but the hidden state is of the same size. So the final result is that we end up with a tensor. That's got an extra to long axis right and you know, depending on what library you use often that will be then combined with the number of layers things. So if you've got two layers and bi-directional that tensor dimension is now length, four with pytorch, it kind of depends which bit of the process you're looking at as to whether you get a

separate result for each layer and offer each bi-directional bit and so forth.

You have to cut the docs and it will tell you inputs/outputs cancer sizes appropriate for the number of layers and whether you have bi-directional equals true. In this particular case, you'll basically see all the changes I've had to make so, for example, you'll see when I added bi-directional, it was true. My linear layer now needs number of hidden times to to reflect the fact that we have that second direction in our hidden state now you'll see in in it hidden. It's now self dot number of layers times two here. Okay, so you'll see there's a few places where there's been an extra two that has to be thrown in yes, your net um. Why making any color by the original is considered cheating? Well, it's it's not just this cheating! It's like we have this loop going on. You know it's not as simple as just kind of having two tenses and then like. How do you turn those two separate loops into a final result? Hey you know, after talking about it, during the break I've kind of gone from like hey everybody knows it doesn't work to like. Oh, maybe it kind of could work, but it requires more thought. It's quite possible. During the week I realize it's a dumb idea and I was being stupid but well think about it. Another question people had: why do you need to have an into that look? Why do I have a watch of the loop? Why do you need to like have a an end to that look, you have like a range if your range yeah, oh, I mean it's because when I start training everything's random, so this will probably never be true so later on it'll, pretty much always break out.

Eventually, but yeah, it's basically like we're going to go for it. It's really important to remember like when you're designing an architecture that, when you start the model, knows nothing about anything right, so you kind of want to make sure it's doing something at least vaguely sensible. Okay, so bi-directional means we had. You know: let's see how we go here, we got out to 358 cross entropy loss, okay, with a single direction with fire direction, gets down to 351 yeah so that improved it a bit. That's good and, as I say the only you know it shouldn't really slow things down too much. You know, bidirectional does mean there's a little bit more sequential processing.

10. 01:22:15

- **Candidate Generation and LUNA 16 (Kaggle)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Have to happen, but you know it's generally, a good win in the Google Translation model of the eight layers. Only the first layer is bi-directional because it allows it to do more in parallel. So if you create really deep models, you may need to think about which ones are bi-directional. Otherwise we have performance issues, okay and so 351. Now, let's talk about teacher for C so teacher forcing is I'm going to come back to this idea that when the model starts learning, it knows nothing about nothing. So when the fertile starts learning it is not going to spit out. Ah, at this point, it's going to spit out some random meaningless work acts. It doesn't know anything about German or about English or about the idea of language or anything. And then it's going to feed it down here as an input and be totally unhelpful yeah, and so that means that early learning is going to be very, very difficult because it's feeding in an input that's stupid into a model. That knows nothing and somehow it says, get better right. So that's it's not asking too much. Eventually it gets there, but it's definitely not as helpful as we can be. So what if, instead of feeding in what? If, instead of feeding in the thing I predicted just now right what? If, instead, we see it in the actual correct word, it was meant to

be right. Now we can't do that at inference time, because, by definition we don't know the correct word it has to translate it.

We can't require the correct translation in order to do translation right. So the way I've set this up is I've got this thing called PR force, which is probability of forcing, and if some random number is less than that probability, then I'm going to replace my decoder input with the actual correct thing right and if we've already gone Too far, and if it's already longer than the target sentence, I'm just going to stop well, obviously I can't give it the correct thing, so you can see how beautiful pytorches for this right because like if you try to do this with some static grafting, like Like classic tensorflow well, I tried right, like one of the key reasons that we switched to pytorch at this exact point in last year's class was because Jeremy tried to implement it. You're forcing in chaos intensive flow and went even more insane and he started right and I was like it was weeks of getting nowhere, and then I literally on Twitter. I think it was Andre, capaci, eyesore announced say it's something about. Oh there's this thing called pytorch just came out and it's really cool and I've tried it that day by the next day I had teacher flossing would and so like. I was like, oh my gosh, you know and like all the stuff of trying to debug things, it was suddenly so much easier, and this kind of you know dynamic stuff is so much easier. So this is a great example of like hey.

I get to use random numbers and, if statements and stuff so yeah, so here's the basic idea is at the start of training. Let's set P our force really high right so that nearly always that gets the actual correct. You know previous word, and so it has a useful input right and then, as I train a bit more, let's decrease PR force so that by the end, PF force is zero and it has to learn properly, which is fine, because it's now actually feeding in sensible Inputs most of the time anyway, so let's now write something such that in the training loop, it gradually decreases pyaare force. So how do you do that? Well, one approach would be to write our own training loop, okay, but let's not do that because we already have a training loop that has progress, bars and uses exponential weighted averages to smooth out the losses and keeps track of metrics. And you know it does a bunch of things which they're not rocket science.

11. 01:26:30

- **Accelerating K-Means on GPU via CUDA (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

But they're kind of convenient and they also kind of keep track of you - know calling the reset hour and ends at the start of an epoch, to make sure that the hidden states set to zeros - and you know little things like that. We'd, rather not have to write that in scratch. So what we've tended to find is that, as I start to kind of write some new thing and I'm like oh, I need to kind of replace some part of the code. I then kind of add some little hook so that we can all use that book to make things easier in this particular case. There's a book that I've ended up using all the damn time now, which is the hook called the stepper. And so if you look at our code model, dot, pi

12. 01:27:15

- **ChatBots ! (long section)**
- **Staring with “memory networks” at Facebook (research)**

Is where our fit function lives right and so the fit function and model dot. Pi is kind of we've seen it before. I think it's like the lowest level thing that doesn't require and learner. It doesn't really require anything much at all. This requires a standard, pytorch model and a model data object. You just need to know how many epochs are standard, pipe torch, optimizer and the standard page watch boss function all right, so you can call. I don't. We've hardly ever used it in the class. We normally call learn but fit, but learn dot fit calls this. So this is our lowest level thing, but we filtered the source code here. Sometimes, we've seen how it loops through each epoch and that lives through each thing in our batch and calls step or dot, step right and so step or dot step is the thing that's responsible for calling the model getting the last finding the last function and calling The optimizer, and so by default, step or dot step uses a particular class called stepper, which there's a few things you don't know where. Basically, it calls the model all right. So the model winds up inside M Zero's the gradients cause the loss function, calls backwards, does gradient flipping if necessary, and then calls the optimizer alright. So you know they're the basic steps that back when we looked at kind of pytorch from scratch. We had to do so. The nice thing is, we can replace that with something else, rather than replacing the training loop right.

So if you inherit from step up and then write your own version of step right, you can just copy and paste the contents of step and add whatever you like right or if it's something that you're going to do before or afterwards, your could even call super Dot step in this case, I rather suspect I've been unnecessarily complicated. Here I probably could have replaced, commented about all of that and just said super darts, DEP X's, comma Y comma epoch, because I think this is an exact copy of everything right. But you know, as I say when I'm prototyping, I don't think carefully about how to minimize my code. I copied and pasted the contents of the code from step and I added a single line to the top, which was to replace PR force in my module with something that gradually decreased linearly for the first 10 a box and after 10 B box it was zero. Okay, so total hack, but good enough to try it out, and so the nice thing. What is that I can now you know everything else is the same: I've replaced I've added these three lines of code to my module and the only thing I need to do other that's differently is when I call fit. Is I pass in my customized step, a class okay and so that's going to do teacher forcing, and so we don't have bi-directional, so we're just changing one thing at a time. So we should compare this to our unidirectional results, which was three point.

Five, eight - and this is three point - four: nine okay, so that was an improvement, so that's great needed to make sure at least two ten epochs, because before that it was cheating by using the teacher forcing so yeah okay. So that's good! That's an improvement! So we've got another trick, and this next trick is a it's a bigger trick. It's a it's a pretty cool trick and it's it's called attention and the basic idea of attention is this, which is expecting the entirety of the sentence to be summarized into this single hidden vector is asking a lot. You know it has to know what was said and how it was said and everything necessary to create the sentence in general, and so the idea of attention is basically like. Maybe we're asking too much all right, particularly because we could use this form of model where we output every step of the loop to not just have a hidden state at the end, but to hit a hidden state after every single word and like. Why not try and use that information? It's it's like it's already there, but so far, we've just been throwing it away, and not only that, but by directional we've got every step. We've got to. You know vectors of state that we can use. So how could we use this piece of state this piece of state, this piece of state, this piece of state and this piece of state, rather than just the

final state, and so the basic idea is well. Let's say I'm doing this word translating this word right now, which of these five pieces of state.

Do I want and of course the answer is, if I'm doing well, actually, that's bigger more interesting would pick this one. So if I'm trying to do loved, then clearly the hidden state I want is this one right, because this is the word okay and then for for this preposition of little preposition, whatever this little word here? No, it's not a preposition. I guess it's part of the boat so for this part of the verb, I probably would need like this and this and this to kind of make sure that I've got kind of the tense right and know that I actually need this part of the verb. And so forth, so depending on which bit I'm translating I'm going to need one or more bits of this, of these various hidden States, and in fact you know like I probably want some weighting of them so like what I'm doing here, I probably mainly want this State right, but I maybe run a little bit of that one and a little bit of that one right. So, in other words, for these five pieces of hidden state, we want to wait an average right and we wanted waited by something that can figure out which bits of the sentence the most important right now. So how do we figure out? Something like which bits are the symptoms are important right now we created a neural net and we train the neural net to figure it out. When do we train that you're on it and to end so, let's now train to neural nets? Well, we've actually already kinda got a bunch right, we've got an hour and an encoder.

We got an RNN decoder. We've got a couple of linear layers. What the hell, let's add, another neural net into the mix. Okay and this neural net is going to spit out a wait for every one of these things and we got to take the weighted average at every step and it's just another set of parameters that we learn all at the same time. Okay, and so that's called attention, so the idea is that once that attentions been learned, we can see this terrific demo from Chris Olerud roncada. Each different word is going to take a weighted average, see how they're weighted the weights are different, depending on which word is being later all right, and you can see how it's kind of figuring out the color. The deepness of the blue is how much weight it's using. You can see that each word is basically or here at which word are we transferring from so when we say European, we need to know that both of these two parts, you only influenced ordering economic. Both these three parts are very influenced, including the gender of the the definite article and so forth, right so check out this distill. The pub article are, these things are all like little Hospital interactive diagrams, basically shows you how weights how attention works and what the actual attention looks like, and a trained translation model. Okay. So, let's try and implement attention so with attention it's basically, this is all identical right and the encoder is identical and all of this bit of the decoder is identical.

There's one difference which is that we see where this happens here we go. We basically are going to take a weighted average and the way that we're going to do the weighted average is we create a little neural net which we're going to see here and here and then we use softmax because of course, the last thing about softmax is That we want to ensure that all of the weights that we're using add up to 1 - and we also kind of expect that one of those weights should probably be quite a bit higher than the other ones right and so soft mattes gives us the guarantee that They add up to 1 and because it's that aid of that unit it tends to encourage one of the weights to be higher than the other ones all right. So, let's see how this works. So, what's going to happen, is we're going to take the last layers? Hidden state and we're going to stick it into a linearlayout and then we're going to stick it into a nonlinear activation and then we're going to do matrix multiply, and so, if you think about it, linear layer, nonlinear activation matrix multiply, that's a neural net. It's a neural net with one hidden layer: okay, stick it into a softmax okay and then we can use that to weight our air encoder outputs, okay.

So now, rather than just taking the last encoder output, we've got, this is going to be the whole tensor of all of the encoder outputs, which I just weight by this little neural net that I created and that's basically it right so , okay.

So what I'll do is I'll put on the wiki thread couple of papers to check out there was. There was basically one amazing paper that really originally introduced this idea of attention, and I say amazing because it actually introduced a couple of key things which have really changed how people work in this field. They say, area of attention has been used, not just for text but for you know things like reading text out of pictures or kind of doing various stuff with computer vision and stuff like that, and then there's a second paper which actually Geoffrey Hinton was involved in Called grammar as a foreign language which used this idea of Iranians with attention to basically try to replace rules based grammar with an R and n which automatically basically tagged the grammatical. You know each word based on this grammar and turned out to do it better than any rules based system which today, like actually kind of, seems obvious. I think we're now used to the idea that neural nets do lots of this stuff better than rules-based systems, but at the time I was considered really surprising anyway. One nice thing is that they're kind of summary of how attention works is really nice and concise. You know, can you please explain a thing again sure so here's the idea, I like that nice crisp request. That's very easy to understand.

Okay: let's go back and look at our original encoder, so an eridan spits out two things: it spits out a list of the the state after every time step, and it also tells you the state at the last time step and we used the state at the Last time, step to create the input state for our decoder, okay, which is what we see you won, but we know that it's actually creating a vector at every time step. So, wouldn't it be nice to use them all right, but wouldn't it be likes to use the one or ones that's most relevant to translating the word I'm translating now. So, wouldn't it be nice to be able to take a weighted average of the hidden state at each time step weight it by whatever is the appropriate weight right now, which, for example, in this case litter, would definitely be time step number two: here's what it's all About because that's the word I'm translating so how do we get? How do we get a list of weights that is suitable for the word? We're training right now? Well, the answer is by training a neural net to figure out the list of weights, and so anytime. We want to figure out how to train a little neuron that that does and tasks the easiest way normally always to do. That is to include it in your module and train it in line with everything else. The minimal possible neural net is something that contains two layers and one nonlinear activation function. So here is one linear layer, okay, and in fact you know.

Instead of a linear layer, we can even just grab a random matrix if we don't care about bias right and so here's a random matrix, it's just a random tensor wrapped up in a parameter. A parameter. Remember is a is just a pytorch variable. It's like identical to a variable that it just tells pytorch. I want you to learn the weights for this place. Alright. So here we've got a linear layer. Here, we've got a random matrix, and so here at this point where we start out our decoder. Let's take that final, let's take the current hidden state of the tked of the decoder right put that into a linear layer. Right because, like how what's the information we use to decide what words we should focus on next? Well, we, the only information we have to go on, is what the decoders hidden state is now all right. So, let's grab that put it into the linear layer, put it through a non-linearity, put it through one more nonlinear layer. This one actually doesn't have a bias in it, so it's actually just a matrix multiply, put that into a soft Max and that's it right, that's a little neural net. It doesn't do anything. We could it's just a neuron that no neuron it's do anything they're. Just linear layers with nonlinear activations with random weights right, but it starts to do something if we give it a job to do right, and in this case the

job we give it to do is to say don't just take the final state.

But now, let's use all of the encoder states and let's take all of them and multiply them by the output of that little neuron it right and so, given that the things in this little neural net are learnable weights. Hopefully it's going to learn to wait. Those encoder outputs, those encoder hidden States by something useful, all right, that's all in neural net ever does is. Is we give it some random weights to start with and a job to, do and hope that it learns to do the job and the ants, and it turns out that it does all right. So everything else in here is identical to what it was before. We've got teacher forcing it's not bi-directional, so we can see how this goes right. You can see. Actually, I am oh yes here, yeah using bi-directional, that's are using teacher forcing so teacher foreseeing had three point: four, nine and so now we've got nearly exactly the same thing, but we've got this little minimal, neural net figuring out what weightings to give our inputs? Oh wow, now it's now down to three point: three. Seven, all right remember these things are logs right, so e ^. This is quite a significant change. So three point three: seven: let's try it out, not bad right. Where are they located? What are their skills? What'd? You do it's still not perfect. Why or why not, but it's quite a few of them are correct and again considering that we're asking it to learn about the very idea of language for two different languages and how to translate them between the two and grammar and vocabulary, and we only have 50,000 Sentences and a lot of the words only appear once I would say this is actually pretty amazing.

Yes Annette, why do we use tongue H? You still free Lu for attention minute. I don't quite remember: it's been a while, since I looked at it, you should totally try using value and see how it goes. Obviously, then, the key difference is that it kind of go in each direction and it's as limited both at the top and the bottom. I know very often like in it like, for the gates inside our own ends and Ellice Tian Xin Jie, I use fan often works out better, but it's been about a year since I actually doctored that specific question. So I look at it during the week, but the short answer is, you know you should try a different activation function and see if you can get a better result, be interested to hear what you find out. So what we can do also is. We can actually grab the attentions out of the model right, so I actually added this return. Attention equals true here, look see here in my forward like forward. You can put anything you like in forward, so I added a return. Attention parameter false by default, because obviously the the training loop, it doesn't know anything about it. But then I just had something here saying: if returned attention, then stick the attentions on as well right and the attentions is simply that value a just check it in a list. Okay, so we can now call the model with returned attention equals true and get back. The probabilities and the attentions betweens, as well as printing out these here we can draw pictures at each time step of the attention, and so you can see at the start.

The attentions on the first word. Second, word third word a couple of different words, and this is just for one particular sentence right, but so you can kind of see this is equivalent. This is like you know, when you're Chris, solar and and and and Sean kata, you make things that look like this when you Jeremy, Howard, the exact same information, looks like this, but it's the same thing. Okay, just pretend that it's beautiful bit, so you can see basically at each different time step. We've got a different, a different attention and it's really important when you try to build something like this. Like you, don't really know if it's not working right, because if it's not working - and you know as per usual, my first twelve attempts that this were broken and and they were broken in the sense that it wasn't really learning anything useful and so therefore was basically Giving equal attention to everything and therefore it wasn't worse, it just wasn't better whenever it wasn't much better, and so I'm sure you actually

find ways to visualize the thing in a way that you know what it ought to look like ahead of time. You don't really know if it's working, so it's really important that you try to find ways to kind of check your intermediate steps in your outputs. Yes, you know, so we won't ask you: what is the last function for the attention on your network? No, no! No! Loss function to determine your network right, it's trained in to end, so it's just like it's just sitting here inside out decoder, look right, so the loss function for the decoder loop is that this result contains it's exactly the same as before.

Just the outputs, the probabilities of the words right so, like the loss function, it's it's the same. Loss function right, so so how come the? How come the little little mini? Neural nets? Learning something well because, in order to make the outputs better and better it, it would be great if it made the weights of these little weighted-average better and better right. So part of creating our output is to please do a good job of finding a good set of weights, and if it doesn't do a good job of finding good set of weights, then the loss function won't improve from that bit. So, like end-to-end learning means like you throw in, you know everything that you can into one loss function and the gradients of these of all the different parameters. Point in a direction that says basically hey. You know if you had put more weight over there, it would have been better and thanks to the magic of the train rule it then no, it's like oh well. It would have put more weight over there if you would like change the parameter in this matrix. Multiply a little bit over there all right, and so that's that's the magic of end-to-end learning. So it's a very understandable question of like how did this little mini in your network, but you've got to realize. There's nothing particularly about this code that says: hey this particular bits.

A separate little mini neural network anymore than the grooc is a separate little neural network, or this linear layers is a little function like it's all ends up pushed into one output, which is a bunch of probabilities which ends up in one lost function. That returns a single number that says this either was or wasn't a good translation right, and so thanks to the magic of the chain rule, we then back propagate little updates to all the parameters to make them a little bit better okay. So this is a a big, weird counterintuitive idea and it's totally ok if it's a bit mind-bending right and it's the bit where even back to lesson 1. You know it's like how did this? How did we make it find dogs versus cats? So we didn't. You know all we did was we said this is our data. This is our architecture. This is our loss function, please back propagate into the weights to make them better and after you've made them better, a while it'll start finding cats from dogs right. That's just in this case we haven't used somebody else's like convolutional network architecture. We have said here's like a custom architecture, which we hope is going to be particularly good at this problem right and even without this custom architecture, it was ok all right, but then, when we kind of made it in a way that made more sense short, we Think it ought to do it worked even better, but at no point did we kind of do anything different other than say: here's, a data, here's an architecture, here's a loss, function, go and find the parameters, please pay ins and it did it because that's what your Net to do okay, so that is sequence, the sequence planning and you know if you want to encode an image into you, know using CNN backbone of some kind and then pass that into a decoder which is like a our an N with attention, and you make Your y-values, the actual, correct caption speech of those images.

You will end up with an image caption generator if you do the same thing with videos and captions you'll end up with a video caption generator. If you do the same thing with 3d CT scans and radiology reports, you'll end up with a radiology report generator if you do the same thing with github issues, and you know people's chosen summaries of them you'll get a github issue, summary generator, which you know it Sector, sick, I agree, you know they're

magical, but they they work. You know, and I don't feel like people have begun to scratch the surface of how to use sector SEC models in their own domains. So that's like not being a github person, it would never have occurred to me that, like oh, it would be kind of cool to start with some. You know issue and automatically create a summary, but now I'm like huh, of course. Next time I go to github, I want to see a summary written there for me. I don't want to write my own damn commit message through that. You know why should I write my own, like summary, of the code review when I finished adding comments, just lots of lines, it should do that for me as well. Now, I'm thinking like Oh github so behind it could be doing this stuff. So what are the things in your industry? You know that you could like start with a sequence in generate something from it. I can't begin to imagine right so again, it's kind of like it's a fairly new area. The tools for it are not easy to use.

They're not even built into fastai, yet, as you can see, hopefully there will be you know soon and, like you know, I don't think anybody knows what the opportunities are. Okay, so I've got good news, bad news, the bad news is, we have 20 minutes to cover a topic which in last year's course took a whole lesson. The good news is that when I went to rewrite this using fastai Empire torch, I ended up with almost no code, so all of the stuff that made it hard last year is basically gone now, so we're going to do something bringing together for the first Time our two little worlds, we focused on text and images and we're going to try and bring them together, and so this idea came up really in a paper by this extraordinary deep learning, practitioner and researcher named Andrea, Framm and Andrea was Google at the time and Her basic crazy idea was to say, like you know, words can have a distributed representation, a space which, particularly at that time was, you know, really was just word vectors right and images can be represented in a space I mean like in the end, if we have Like a fully connected layer, they kind of ended up as like a vector representation. Could we merge the two? Could we either? It could be somehow encourage the vector space that the images end up with be the same vector space that the words are in and if we could do that, what would that mean? What could we do with that right? So, though, what could we do with that? Covers things like? Well? What, if I'm wrong, you know what, if I'm predicting, that this image is a beagle, and I predict jumbo jacked, you know, and you nets model predicts Corgi. The normal loss function says that your net and Jeremy's models are equally good, ie they're both wrong right. But what if we could somehow actually say, though, you know what like Corky's closer

13. 01:57:30

- **Recurrent Entity Networks: an exciting area of research in Memory Networks**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

To beagle than it is to jumbo jets, so units models better than Jeremy's, and we should be able to do that right because in in in in word, vector space, vehicle and Corgi are pretty close together, but jumbo jet, not so much okay. So it would give us a nice situation where, hopefully, our inferences would be like wrong insane aways if they're wrong. It would also allow us to search for things that aren't in our you know an image net since set ID. You know like a category in image net like like dog and cat. Why did I have to train a whole new model to find dog vs. cats when we already have something that found Corky's and Paddy's right? Why can't, I just say, find me dogs? Well, if I had trained it in in word vector space, I totally could right because, like that there there's our word vector, I can find things with the right image, vector and so forth. So we'll look at some cool things we can do with it in a moment,

but first of all, let's train a model where this model is not learning a category, a one hotted coded ID where

14. 01:58:45

- **Concept of "Attention" and "Attentional Models"**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Every category is equally far from every other category, let's instead trainer model where we're finding the dependent variable, which is a word vector, so what word vector? Well, obviously, the word vector for the word you want right. So if it's Corgi, let's train it to create a word vector, that's that's the Corgi work that don't, if it's a jumbo jet, let's train it with a dependent variable that says this is the word vector for a jumbo jet. Okay. So, as I said, it's now shockingly easy, okay, so let's grab the fart. The fasttech word vectors again load the men. We only need English this time right and so here's an example of the word vector for King, but it's just a 300 numbers. So, for example, you know little J, Jeremy and big J Jeremy have a correlation to point six. I don't like bananas at all. This is good banana and Jeremy point. One four right so like words that you would expect to be correlated are correlated in words. That should be as far away from each other as possible. Unfortunately, they're still slightly correlated, but not so much right. So let's now grab all of the imagenet classes because we actually want to know you know which one's Corgi and which ones jumbo jet. So we've got: we've got a list of all of those up on files, doc, bastard AI. We can grab them and that's also grab a list of all of the nouns in English, which I've made available here as well.

Okay, so here are the names of each of the thousand imagenet classes, and here are all of the nouns in English, according to work net, which is a popular thing for kind of representing what words are or not. So we can now go ahead and load that list of nouns and sorry load. The list of imagenet classes turn that into a dictionary. So these are the class IDs for the 1000 images that are in the competition data set they're at a thousand okay. So here's an example and 0-100 Allah is attentively is a kind of fish. Let's do the same thing for all those word net nouns and you can see. Actually, it turns out that image net is using word net class names, so that makes it nice and easy to map between the two and word net. You know most basic thing is an entity and then that if it's an abstraction and a physical entity could be an object and so forth right. So these are our two worlds: we've got the image net thousand and we got the 80 mm which are in wordnet. So we want to map the two together and which is as simple as creating a couple of dictionaries to map them based on the on the scene set ID or the web net ID, and it turns out that 49 thousand four hundred and sixty nine, let's see Since set oh okay, so what I need to do now is grab the 80 mm nouns in word net and try and look them up in fast text.

Okay, and so I've managed to look up 49 thousand of them in fast text all right. So I've now got a dictionary that goes from sin set ID, which is what wordnet calls them to word vectors. Okay. So that's what this dictionary! Yes, sin sin set to work vector and I've also got the same thing specifically for the 1000 word net classes. So save them away. That's fine! Now I grab all of the image net which you can actually download from cable. Now, if you look at the cackle image net, localization competition that contains the entirety of the image net classifications as well, it's got a validation set of 28,650 items in it, and so i can basically just grab for every image in image net. I can grab using that sin set to work vector grab it's its

word net. Sorry, it's a fast text word vector and I can now stick that into this image. Vectors array stack that all up into a single matrix and save that away, and so now what I've got is something for every image. Net image they've also got. The fast text word vector that it's associated with like just by looking up. You know the the sin set ID going to word net then going to first text and grabbing, though the the word vector, okay and so here's a cool trick. I can now create a model data object, which is specifically it's an image.

Classifier data object and I've got this in court from names, an array I'm not sure if we've used it before, but we can pass it a list of file names, and so these are all of the file names in an image net and we can just pass It an array of our dependent variables - and so this is all of the fast text, word vectors right and then I can pass in the validation indexes, which in this case is just all of the last IDs. I need to make sure that they're the same as image: net users, otherwise I'll be cheating, okay and then I pass in continuous equals, true, which means this puts a lie again to this image. Classifier data is now really an image. Regressive data so continuous equals true means, don't one-hot encode my outputs, but treat them just as continuous values. So now I've got a model data object that contains all of my file names and for every file name, a continuous array representing the word vector for that. So I have an X, I have a Y, so I have data now. I need an architecture and the last function that once I've got that I should be done. So let's create an architecture, and so we can roll revise this next week, but basically we can use the tricks we've learnt so far, but it's actually incredibly simple fast. Ai has a ComNet builder, which is what, when you say, you know, come flow, no dot pre-trained. It calls this and you basically say: okay, what architecture do you want, so we're going to use ResNet 50? How many classes do you want in this case it's not really classes, it's. How many outputs do you want, which is the length of the fast text? Word vector? That's 300. Obviously, it's not model class classification, it's not classification at all. Is it regression? Yes, it is regression okay and then you can just say all right.

What fully connected layers do you want? So I'm just going to add one fully connected layer, hidden layer of length, a thousand and twenty four, why a thousand and twenty four well I've got the the last layer of resin 850. Is I think, it's a thousand twenty four long? The final output I need is three hundred long. I obviously need my penultimate layer to be longer than three hundred. Otherwise it's not enough information, so I kind of just picked something a bit bigger. Maybe different numbers would be better, but this worked for me. How much dropout do you want? I found that the default dropout. I was consistently under fitting, so I just decreased the dropout from 0.5 to 0.2, and so this is now a convolutional neural network. That does does not have any softmax or anything like that, because it's regression, it's just a linear layer at the end and yeah. That's basically it that that's that's my model, so I can create a common floater from that model. Give it an optimization function. So now all I need I've got data. I've got an architecture right, so the architecture, because I said I've got this many three hundred outputs. It knows that there are three hundred outputs, because that's the size of this array right so now. All I need is a loss function. Now the default loss function for regression is l1 loss, so the absolute differences - that's not bad right, but unfortunately, in really high dimensional spaces.

Anybody whose could have studied the bit of machine learning probably knows this in really high dimensional spaces. In this case it's three hundred dimensional and basically, everything is on the outside okay and when everything's on the outside distance. Is it's not meaningless, but it's it's a little bit awkward like things you know things. Can things tend to be close together or fire doesn't really mean much in these really high dimensional spaces, but everything's on the edge right? What does mean something, though, is that if one things on

the edge over here and one things on the edge over here, you can form an angle between those vectors and the angle is meaningful right, and so that's why we use a cosine similarity when we're. Basically, looking for like how close or far apart are things in high dimensional spaces right and if you haven't seen cosine similarity before it's basically the same as you're fitting in distance, but it's normalized to be basically a unit norm. That's so it basically divided by the length. So we don't care about the length of the vector we only care about its angle. Okay, so there's a there's, a bunch of like stuff that you could easily learn in a couple of hours. But but if you haven't seen it before it's a bit mysterious for now just know that loss functions in high dimensional spaces, where you're trying to find similarity, you you care about angle and you don't care about distance.

Okay, if you didn't use this custom loss function, it would still work. I tried it. It's just a little bit less good. Okay, so we've got an architecture. We've got data. We've got a loss function. Therefore, we're done okay, so we can go ahead and fit now. I'm training on all of imagenet right, that's going to take a long time, so pre-compute equals true. As your friend okay, you remember, precompute equals true. That's that thing we learnt ages ago that caches the output with the final convolutional layer that and just trains the fully connected bit yeah and like even with pre compute, equals true. It takes like three minutes to train an epoch on all of imagenet, so I trained it for a while. I trained it for a while longer. So it's like an hour's worth of training right, but it's pretty cool that you know with fastai. We can train, you know a new custom head, basically on all of imagenet for 48 bucks. You know in an arrow, so okay, and so at the end of all that we can now say all right grab the 1000 images all right and, let's predict, on our whole validation, set right and let's just take a look at a few pictures. Okay, so here's a look at you know a few pictures and because the validation set is ordered, you know they're, all all the stuff is the same type as in the same place. I don't know what this thing is and what we can now do is we can now use nearest neighbors search all right, so nearest neighbors search means. You know.

Here's one 300 dimensional vector here's a whole lot of other three-dimensional vectors which things is it closest to Wow, and normally that takes a very long time, because you have to look through every 300 dimensional vector caplets at distance and find out how far away it is. Okay, but there's an amazing, almost unknown library quote NMS Lib that that does that incredibly fast, like almost nobody's heard of it, some of you may have tried other nearest neighbors libraries. I guarantee this is faster than what you're using. I can tell you that, because it's been benched much like by people who do this stuff for a living, this is by far the fastest on every possible dimension right. So this is basically a super fast way. We basically look here. This is angular distance right, so we want to create an index on angular distance and we're going to do it on all of our imagenet word vectors right, adding a whole batch create the index, and now I can query a bunch of vectors all at once. Get there ten nearest neighbors users, Bodi threading, it's it's absolutely fantastic, this library, okay, you can install it from pip, it just works and it tells you how far away they are and their indexes right, and so we can now go through and print out. The top three, so it turns out that bird actually is a limpkin okay, so here are there. This is the top three for each one. Interestingly, this one doesn't say it's a limpkin and I looked it up. It's the fourth one.

I don't know much about birds but like everything else here is brown with white spots. That's not so I don't know if that's actually a limb tune or if there's the mislabel, but I sure, as hell doesn't look like the other birds. So you know I thought that was pretty interesting, that yeah it's kind of saying like. I don't think it's that now. This is not a particularly hard thing to do, because it's only a thousand imagenet classes. It is not doing anything new right, but what? If

we now bring in the entirety of wordnet - and we now say which of those forty-five thousand things as a closest to exactly the same right, so it's now searching all of wordnet right. So now like. Let's do something a bit different, which is take all of our predictions right so basically take our whole validation, set of images and create a KNN index of the image representations, because remember it's predicting things that are meant to be word vectors and now, let's grab the Fast text, vector for boat and boat, is not an image net concept right and yet I can now find all of the images in my predicted word vectors in my validation set that are closest to the word boat and it works. Even though, though, it's not something that was ever trained on what if we now take engine's vector and boats vector and take their average and what, if we now look in our nearest neighbors for that these are boats with engines, I mean yes, this is this is Actually, a boat with an engine it just happens to have wings on as well right by the way sail is not an image net thing boat is not an image net thing. Here's the average of two things that are not image net things, and yet, with one exception, it's bound me to sailboats.

Well, okay, let's do something else: crazy, let's open up an image in the validation set. Here it is hey. I don't know what it is. Let's call predict array on that image to get you know, it's kind of like word victor, like thing and let's do a nearest neighbors search on all the other images and here's all the other images of whatever. That is that, so you can see this is like crazy. We've trained a thing on all of imagenet in an hour using like a custom head that required basically like two lines of code and these things like run in like 300 milliseconds to do these searches like I. I actually taught this basic idea last year as well, but it was in chaos and it was just like pages and pages and pages of code and everything took a long time and it's complicated and back then I kind of said yeah. I can't begin to think all the stuff you could do with this, as I don't think anybody's really thought deeply about this yet, but I think it's fascinating and so go back and read the devise paper because, like Andrea, had a whole bunch of other thoughts and Now that it's so easy to do, hopefully people will will dig into this now, because I think it's crazy and amazing all right thanks. Everybody see you next week, [Applause,]

Lesson 12: Generative Adversarial Networks (GANs)

Outline

We start today with a deep dive into the DarkNet architecture used in YOLOv3, and use it to better understand all the details and choices that you can make when implementing a resnet-ish architecture. The basic approach discussed here is what we used to win the DAWNBench competition!

Then we'll learn about Generative Adversarial Networks (GANs). This is, at its heart, a different kind of loss function. GANs have a generator and a discriminator that battle it out, and in the process combine to create a generative model that can create highly realistic outputs. We'll be looking at the Wasserstein GAN variant, since it's easier to train and more resilient to a range of hyperparameters.

Video Timelines and Transcript

1. 00:00:05

- **K-means clustering in TensorFlow**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

So we're going to be talking about Ganz today, who has heard of Ganz yeah most of you very hot technology, but definitely deserving to be in the cutting edge, deep learning part of the course because they're not quite proven to be necessarily useful for anything but they're. Nearly there they're definitely going to get there and we're going to focus on the things where they're definitely definitely going to be useful in practice and there's a number of areas where they may turn out to be useful in practice. But we don't know yet. So I think the area that we're going to be that they're, definitely going to be useful in practice, is the kind of thing you see on the left here, which is, for example, turning drawings into rendered pictures. This comes from a paper. That's just came out two days ago, so there's a very active research going on right now before we get there, though, let's talk about some interesting stuff from the last class. This was an interesting thing that one of our diversity fellows Christine Payne did Christine. Has a master's in medicine from Stanford, and so she OC had an interest in thinking. What would it look like if we built a language model of medicine and one of the things that we briefly touched on back in less than four, but didn't really talk? Much about last time is this idea: you can actually seed a generative language model which basically means you've trained a language model on some corpus and then you're going to generate some text from that language model, and so you can start off by feeding it.

A few words you know to basically say here's the first few words to create the hidden state in

the language model and then generate from there. Please, and so, when Christine so Christine did. Something clever which was to kind of pick up was to seed it with a question and then repeat the question. So it's 3 times Christine three times and then where'd it generate from there, and so she fed a language model, lots of different medical texts and then fed at this question. What is the prevalence of malaria and the model said in the US about 10 percent of the population? Has the virus, but only about one percent is infected with a virus about 50 to 80 million infected. She said: what's the treatment for ectopic pregnancy and it said it's: a safe and safe treatment for women with the history or symptoms may have a significant impact and clinical response. Most important factor is development and management of ectopic privacy, etc, and so what I find interesting about this is, you know it's it's pretty close to it being a to me, as somebody who doesn't have a masters in medicine from Stanford are pretty kind of close to Being a believable answer to the question, but it really has no bearing on reality whatsoever and I kind of think it's an interesting kind of ethical and user experience quandary.

So actually, I'm involved also in a company called dr., a that's trying to basically doing a number of things, but in the end provide an app for doctors and patients which can help kind of create a conversation or user interface around helping them with their medical issues. And I've been kind of continually saying to the software engineers on that team. Please don't try to create a generative model using like an LST em or something because they're going to be really good at creating bad advice. That sounds impressive. You know kind of, like you know, political pundits or tenured professors. You know people who can say with great Authority, so I think yeah. So I thought it was really. I thought it was a really interesting experiment and great to see. You know what what about diversity fellows doing? I mean this is why we have this program. I suppose I should just say masters in medicine, actually, a Juilliard trained, classical musician or on actually also a Princeton valedictorian in physics, so also a woman's a computing expert, yeah. Okay, so she does a bit of everything, so yeah, really impressive group of people and great to see such exciting kind of ideas coming out and if you're wondering you know, I've done some interesting experiments should I let people know about it. Well, I Kristin mentioned this. In the forum I went on to mention it on Twitter, too, which I got this response. You are you looking for a job.

You may be wondering whose AVM aerotrain is well. He is the founder of a hot new medical, AI startup. He was previously the head of engineering at Quora before that he was the guy net flips around the data science team and built their recommender systems, and so this is what happens if you do something cool. Let people know about it and get get noticed by awesome. People like stevia, so let's talk about sci-fi 10 and the reason I'm going to talk about so far 10. Is that we're going to be looking at some more? You know: bare-bones pytorch stuff today to build these generative adversarial models. There's there's no really fast. Ai support to speak up at all Dan's at the moment. I'm sure there will be soon enough. Apparently there isn't so we're going to be building a lot of models from scratch. Now it's been a while, since we've done much, you know,

2. 00:06:00

- **'find_initial_centroids', a simple heuristic**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Serious model building a little bit of model building, I guess for our bounding box staff, but really all the interesting stuff there was the loss function. So we looked at sci-fi 10 in the plat 1 of the course, and we built something which was getting about 85 % accuracy, and I remember a couple of hours to Train. Interestingly, there's a competition going on now to see who can actually train sci-fi turn the fastest going through this earth, Stepford Dawn bench and currently, so the goal is to get it to train to 94 % accuracy. So everything to see if we can build an architecture then can get four percent accuracy, because that's a lot better than our previous attempt, and so hopefully in doing so, we'll learn something about creating good architectures. That will be then useful for looking at these gams today, but I think also it's useful, because I've been kind of looking much more deeply into the last few years papers about different kinds of CN n architectures and realize that a lot of the insights in those Papers are not being widely leveraged and clearly not widely understood, so I want to show you what happens if we can leverage some of that understanding. So I've got this note book called sci-fi 10 darknet. That's because the the particular architecture we're going to look at is it's quite, is really very close to the darknet architecture, but you'll see in the process that the darknet architecture has in not the whole euro version, three end-to-end thing, but just the part of it that They pre trained on imagenet to do classification.

It's almost like the most generic simple architecture. Almost you could come up with, and so it's a really great starting point for experiments. So we're going to call it dark net, but it's not quite that net and you can fiddle around with it to create things that definitely aren't dark. Now it's really just the basis of nearly any modern ResNet based architecture. So so far 10 remember is a fairly small data set. The images are only 32 by 32 in size, and I think it's a really great data set to work with, because it's you can, you can train it. You know relatively quickly, unlike image net, it's a relatively small amount of data, unlike image net. Now it's actually quite hard to recognize the images, because 32 by 32 is it's kind of too small to easily see what's going on, so it's it's somewhat challenging. So I think it's a really underappreciated data set because it's old, you know - and you know who a deep mind or even an AI - wants to work with a small old data set when they could use their entire server room to process something much bigger. But you know to me: I think this is a really great data set to focus on so so go ahead and and kind of import, our usual stuff and we're going to try and build a network from scratch to train this with one thing that I think Is a really good exercise for anybody who's, not a hundred percent, confident with their kind of broadcasting and pytorch and so forth. Basic skills is figure out how I came up with these numbers.

Okay, so these numbers are the averages for each channel and the standard deviations for each channel insofar cap. So try and that's a bit of a homework just make sure you can recreate those numbers and see if you can do it and you know no more than a couple of lines of code. You know no loops all right. Ideally, I want to kind of do it in one: go Ken alright, because these are fairly small. We can use a larger batch size, unusual, 256 and the size of these images is 32 transformations. Normally we kind of have this standard set of sidon transformations. We used for photos of normal objects, we're not going to use that here, though, because these images are so small that trying to rotate a 32 by 32 image, a bit is going to introduce a lot of you know: blocky kind of distortions, so the kind of Standard transforms that people tend to use is a random horizontal flip and then we add size divided by 8. So 4 pixels of padding on each side and one thing which I find works really well, is by default fast. Ai doesn't add black padding, which basically every other library does. We actually take the last 4 pixels of the existing photo and flip it and reflect it, and we find that we get much better results by using this reflection padding by default. So now that we've got a 36 by 36 image, this sort of transforms in training will randomly pick a 32 by 32 crop. So we got a little bit of

variation, but not he's all right, so we can use a normal from past grab our data.

So we now need an architecture, and what we're going to do is we've got to create an architecture which fits in one screen. Okay, so this is from scratch. As you can see, the only you know, I'm using the predefined come 2d veteran onto a daily key value modules, but I'm not using any blocks or anything they're all being defined. So the entire thing is here on one screen, so if you're ever wondering can I understand a modern good quality architecture? Absolutely, let's study. Let's study this one okay, so my basic starting point with an architecture is to say: okay, it's it's! It's a stacked bunch of layers and, generally speaking, there's going to be some kind of hierarchy of layers. So at the very bottom level, there's things like a convolutional layer and a batch nom layer. But, generally speaking any time you have a convolution you're, probably going to have some standard sequence and normally it's going to be con fetch norm. Then a nonlinear activation like a value right. So you know I try to start kind of right from the top by saying. Okay, what are my basic units going to be, and so by defining it here that way, I don't have to worry about. I don't have to worry about kind of trying to try to keep everything consistent, it's going to make everything a lot simpler. So here's my con flavor and so anytime, I say, come flare. I mean cons batch norm rally.

Now, I'm not quite saying value, I'm saying leaky value and that's I think we've briefly mentioned it before, but the basic idea is that normally Lu looks like that right. Hopefully, you all know that now a leaky rail you looks like that right. So this part as before, has a gradient of 1, and this part has a gradient of. It can vary, but something around point one or point zero. One is common now and the idea behind it is that, when you're in this negative zone here, you don't end up with a zero gradient, which makes it very hard to update it. In practice, people have found leaky value more useful on smaller datasets and less useful and big datasets. But it's interesting that for the Yola version 3 paper they did use early key value and got great performance from it. So it really makes things worse and it often makes things better. So it's probably not bad. If you need to create your own architecture to make that your default go to is to use leaky value, ok, you'll notice, I don't define a pipe launch module here. I just go ahead and go sequential. This is something that if you read other people's height watch code, it's really underutilized. People tend to write. Everything is applied, watch module with an inert and a forward, but if you're, if the thing you want is just a sequence of things, one after the other, it's much more concise and easy to understand to just make it a sequential right.

So I just got a simple plain function: it just returns a sequential model. Alright, so I mentioned that there's generally kind of a number of hierarchies of kind of units in most modern networks, and I think we know now that the the kind of next level in this unit hierarchy for rez nets and kind of this. This is a type of resin, it is the is the the res block or the residual block. I quoted here as Leia and back when we lasted sci-fi 10. I over simplify this. I cheated a little bit we had X coming in and we put that through a conf and then we added it back up to X to go out okay, so we ended up so but in general you know, we've got your output is equal to your input, Plus some function of your input right and the thing we did last year was: we meant we made F was a 2d conf, okay, but actually the in the real res block is actually two of them: okay, so it's actually Khan of Khan's of X, okay, and When I say conf, I'm using this as a shortcut for outcome flare - in other words, in other words, calm Vectren on real you, okay, so you can see here. I've created two coms, and here it is. I take my ex put it through the first com. Put it through the second con and add it back up to my input again to get my basic Reds block, okay, so one kind of interesting approach or one interesting insight here - is kind of what are the number of channels in these convolutions right. So we've got coming in

some ni.

Some number of input channels number of inputs well number of input filters, okay, the way that the darknet folks set things up is they said: okay, we're going to make every one of these rez layers spit out the same number of channels that came in, and I Kind of liked that that's why I used it here because it makes life simpler right and so what they did is they said. Okay, let's have the first cons, have the number of channels and then the second con double it again. So ni goes to ni, / and then ni / goes to ni right. So you've kind of got this like funneling thing where, if you've got like 64 channels coming in you kind of get squished down with a first come down to 32 channels and then taken back up again to 64 channels coming out. Yes, retro, why is in place equals true in the leaky rally? Oh thanks for asking a lot of people forget this. I don't know about it, but this is a really important memory technique. If you think about it, this cone flower, it's like the lowest level thing. So pretty much everything in our resin it once it's all put together is going to be complex, complex, complex if you don't have in place, equals true. It's going to create a whole separate piece of memory for the output of the value, so like it's gon na allocate a whole bunch of memory. That's that's totally unnecessary and actually, since I wrote this, I come up came up with an another idea. The other day which I'll now implement, which is you could do the same thing for the res layer rather than going.

Let's just reorder this to say X, plus that you can actually do the same thing here yeah. Hopefully, some of you might remember that in pytorch, pretty much every function has an underscore suffix version which says do that in place. So plus there's also a add, and so that's add in place, and so that's now suddenly reduced my memory there as well. So these are, these are really handy, little tricks and I actually forgot the in place equals true at first for this. In my literally, he was having to decrease my batch size to much lower amounts, and I mean you should be possible and it was driving me crazy, and then I realized that that was missing. You can also do that with drop out by the way. If you have dropped out so drop out and all the activation functions you can do in place and then generally any arithmetic operation you can do in place as well. Why is bias usually like in ResNet set to false in the conflate yeah? So if you're watching the video pause now and see if you can figure this out right, because this is a really interesting question is like: why don't we need bias? Okay, so I wait for you to pause. Okay, welcome back! So if you figured it out, here's the thing right immediately after the Kahn is a batch not and remember. Batch norm has to learn herbal parameters for each activation.

The the kind of the thing you multiply by and the thing you add so since we're if we, if we had bias here to add - and then we add another thing here, we're adding to you things, which is totally pointless like that's two weights, where one would Do right, so if you have a batch norm after a cons, then you can, you can either say in the batch norm. Don't don't include the add bit there. Please or easier is just to say: don't include the bias in it. There's no particular harm. But again it's it's. It's gon na take more memory because that's more gradients that it has to keep track of so best. To avoid also another thing: little trick is most people's comm players have padding as a parameter, but, generally speaking, you should be able to calculate the padding basically enough right, and I see people like trying to like implement. You know special same padding modules and all kinds of stuff like that, but like if you've got a stride, one and you've got or pretty much any strategy and you've got padding of oh sorry and kernel size of three right. Then. Obviously, that's going to overlap by kind of one unit on each side, so we want padding of one or else if its stride one, then we don't need any padding. So in general, padding of kernel size, integer, divided by two: that's what you need. There's some tweaks.

Sometimes, but in this case this works perfectly well so again trying to simplify my code by having the computer calculates stuff for me rather than me having to do it myself. Another thing here with the two common players: so we we kind of have this idea of a bottleneck. This idea of reducing the channels and then increasing them again is also what kernel size we use. So here's a one by one conf right, and so this is again something you might want to pause the video now and think about. What's the one by one con really what actually happens in a one by one con. So if we've got, you know a little 4x4 grid here right and, of course, there's a filters or channels access as well - maybe that's like 32 okay and we're going to do a one by one cons. So, what's the kernel for a one by one, Kong's gon na look like it's gon na be 1 by 32 right, so remember when we talked about the the kernel size, we never mention that last piece, but let's say it's 1 by 1 by 32, because that's Part of the filters in and filters out so in other words, then what happens is this this one thing gets placed first of all here on the first cell and we basically get a dot product of that 32 deep bit with this 32 bit deep bit, and That's going to give us our first oops and that's going to give us our first output, alright and then we're going to take that 32 bit bit and put it with the second one to get the second output right.

So it's basically going to be a bunch of little dot, pallets, ok for each point in the grid. So it's what it basically is then, is a it's basically something which is allowing us to to to kind of change the dimensionality in whatever way we want in the in the channel dimension, and so that would be. That would be one of our filters right and so, in this case we're creating an AI divided by two of these

- 00:12:30 A trick to make TensorFlow feel more like Pytorch
& other tips around Broacasting, GPU tensors and co.

3. 00:24:30

- **Student's question about "figuring out the number of clusters"**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Right so we're going to have ni divided by two of these dot products, always different, basically different weighted averages of the input channels. Okay, so it basically lets us. You know with very little computation at this additional step of calculations and nonlinearities. So that's what that's a cool trick! You know this idea of taking advantage of these one by one cons, creating this bottleneck and then pulling it out again with three by three comms. So that's actually going to take advantage of the you know. The 2d nature of the input properly real so one by one comm, doesn't take advantage of that at all. So these two lines of code there's not much in it, but it's a really great test of your understanding and kind of your intuition about what's going on. Why is it that a one by one cond, going from ni to NI over two channels, followed by a three by three conferring from ni over to de and i and i channels like? Why does it work? Why do the tensor ranks line up? Why do the dimensions or line up nicely? Why is it a good idea? What's it really doing like it's a really good thing to fiddle a fiddle around with maybe create some small ones in jupiter notebook, you know run them yourself, see what inputs now, let's come in and out. You know really get a feel for that once you've done so

4. 00:26:00

- **“Step 1 was to copy our initial_centroids and copy them into our GPU”,**
- **"Step 2 is to assign every point and assign them to a cluster "**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You can then play around with different things right and there's actually one of the really unappreciated papers. Is this one wide residual networks? Okay and it's really quite a simple paper, but what they do is they basically fiddle around with with these two lines of code right and what they do is they say well what if this wasn't 2/2, but what if it was times, two like that'd, be totally Allowable all, right, that's going to line up nicely or what, if we had another comfrey after this, and so this was actually ni over to ni /, and then this is an AO. To again that's going to work right, kernel say is one three one going to half the number of kernels leave it at half and then double it again at the end, and so they come up with this kind of simple notation for basically defining what this can Look like, and then they show lots of experiments and basically what they show is that this approach of a bottlenecking of decreasing the number of channels which is like almost universal and resinates, is probably not a good idea. In fact, from the experiments definitely not a good idea, because what happens is it lets you create really deep networks right and the guys who created resinates got particularly famous recruiting a 1001 bio network? Ok, but the thing about a thousand and one layers is you can't calculate layer, two until you're finished layer 1 from jakarta calculate layer 3 until you finish calculating where so, it's sequential GPUs don't like sequential.

So what they showed is that if you have less layers fat with more active with more calculations per layer - and so one easy way to do, that would be to remove the /. No other changes all right like try. This at home try running, sci-fi and see what happens right or make even more apply it back to you or fiddle around and that basically lets your GPU do more work. And it's very interesting because the vast majority of papers that talk about performance of different architectures never actually time how long it takes to run a batch throat. Like they literally say this one requires X number of floating-point operations per batch, but then they never actually bother to run the damn thing like a proper experimentalist and find out whether it's faster or slower, and so a lot of the architectures that are really famous. Now turn out to be slow as molasses and take crap loads of memory and just totally useless, because the that the research has never actually bothered to see whether they're fast and to actually see whether they fit in RAM with normal size batch batch sizes. So the wide ResNet paper is unusual in that it actually times how long it takes, as does the yellow version 3 paper, which which made the same insight. I'm not sure they might have missed the wide resonance paper, because that the yellow version 3 paper came to a lot of the same conclusions, but I'm not even sure

5. 00:29:30

- **‘Dynamic_partition’, one of the crazy GPU functions in TensorFlow**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

They cited the wide resinous paper, so they might not be aware that all that works being done, but they're both both great to see people actually timing, things and noticing what actually makes sense. Yes, which cell you looked really hot in the paper which came out. But I noticed

that you don't use it. What's your opinion on cell you so sell. You is something largely for fully connected layers which allows you to get rid of batch norm, and the basic idea is that if you use this different activation function, it's it's kind of self normalizing, that's what yes, and so it stands for so self. Normalizing means that all always remain at a unit. Standard deviation and zero mean, and therefore you don't need that pattern on it hasn't really gone anywhere and the reason it hasn't really gone anywhere is because it's incredibly finicky, you have to use a very specific initialization. Otherwise, it doesn't start with exactly the right standard deviation of mean very hard to use it with things like embeddings. If you do, then you have to use a particular kind of embedding initialization, which doesn't necessarily actually make sense for embeddings. So you know - and you do all this work very hard to get it right and if you do finally get it right, what's the point? Well, you've managed to get rid of some batch norm layers which weren't really hurting you anyway, and it's interesting because that paper that sell me a paper.

I think one of the reasons people noticed it or, in my experience the main reason people noticed it was because it was created by the inventor of LST, m/s and also it had a huge mathematical appendix and people were like lots of maths from a famous guy. This must be great, you know, but in practice I don't see anybody using it to get any state-of-the-art results or win any competitions or anything like that. Okay. So this is like some of the tiniest bits of code we've seen, but there's so much here and it's fascinating to play with so now, we've brought this block, which is built on this block and then we're going to create another block on top of that block. Okay, so we're going to call this a group layer and it's going to create a bucket. It's going to contain a bunch of rez layers and so a group layer. It's going to have some number of channels or filters coming in okay, and what we're going to do is we're going to double the number of channels coming in by just using a standard, comm flare, optionally, we'll have the grid size by using a stride of two Okay and then we're going to do a whole bunch of rez blocks a whole bunch of rez layers. We can pick how many that could be two or three or eight back, because remember these rez layers, don't change the grid size and they don't change.

The number of channels - so you can add as many as you like anywhere you like, without causing any problems, and this is going to use more computation and more RAM, but there's no reason other than that. You can't add as many as you like. So a group layer, therefore, it's going to end up doubling the number of channels, because if this initial convolution, which doubles the number of channels, its initial convolution, doubles the number of channels and depending on what we pass in a stride, it may also have the grid Size if we put straight equals to and then we can do a whole bunch of res block computations as many as we like all right so then to define our darknet or whatever. We want to call this thing. We're just going to pass in something that looks like this and what this says is create five group layers. The first one will contain one of these extra rez layers. The second will contain two then 4, then 6, then 3, and I want you to start with 32 filters. Alright, so the first one of these res res layers will contain 32 filters and they'll just be one extra rez layer, the second one. It's going to double the number of filters, because that's what we do each time we have a new group layer, we double the number, so the second one will have 64 and then 128, then 256 and then 512 and then that'll. Be it all right. So that's going to be like nearly all of the network is going to be those bunches of layers and remember.

Every one of those group layers also has one convolution of this data. Okay, and so then all we have is before that all happens. We're going to have one convolutional layer at the very start and at the very end, we're going to do our standard adaptive average pooling flatten and

a linear layer to create the number of classes out at the edge. Alright. So one convolution at the end that we're pulling and one linear layer at the other end and then in the middle, these group layers each one consisting of a convolution or layer, followed by n number of resumes, and that's that's it again. I think we've mentioned this. A few times, but I'm yet to see any code out there, any any exam Falls anything anywhere that uses adaptive average cooling, everyone, I've seen rats it like this and then spits a particular number here right, which means that it's now tied to a particular image size Which definitely isn't what you want, so most people, even the top researchers I speak to. Most of them are still under the impression that a specific architecture is tied to a specific size and that's a huge problem when people think that, because it really limits their ability to like use smaller sizes to kind of kick-start their modeling or to use smaller sizes. For doing experiments and stuff like that again, you'll notice, I'm using sequential here rather than a nice way to create architectures, is to start out by creating a list. In this case.

This is a list with just one comp layer in and then my function here make group layer. It just returns another list right. So then, I can just go plus equals patter pending that list to the previous list, and then I could go plus equals to a pen this bunch of things to that list and then, finally, sequential of all those layers right. So there's a very nice thing. So now my forward is just self dot layers. Okay, so here's a kind of you know. This is a nice kind of picture of how to make your architectures as simple as possible. Okay, so you can now go ahead and create this and, as I say you can fiddle around, you know you could even parameterize this too to make it a number that you kind of pass in here's to pass in different numbers. So it's not too. Maybe it's times two: instead, you could pass in things that change the kernel size or change the number of convolutional layers you know fiddle around with it, and maybe you can create something. I've actually got a version of this which I'm about to run for you, which kind of implements all of the different parameters. That's in that wide ResNet paper, so I could fiddle around. Let's see what worked well so, once we've got that we can use confluent from bottle data to take our pipe watch model module and that a model data object and turn them into a learner. Give it a criterion, that's a metric.

So if we like, and then we can call fit in a way go, could you please explain adaptive average pooling how does setting to one work sure before I do I just want to like, since we've only got a certain amount of time in this class. I wanted to see, I do want to see how we go. You know,

6. 00:37:45

- **Digress: “Jeremy, if you were to start a company today, what would it be ?”**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

With this simple network against these state-of-the-art results, so to make life a little easier, so we can reconstruct later so I've got the command ready to go. So we've basically take taken all that stuff and put it into a simple little Python script and I've modified some of those parameters. I mentioned to create something I've caught of wrn 22 Network, which doesn't officially exist, but it's got a bunch of changes to the parameters. We talked about based on my experiments, we're going to use the new leslie smith, one cycle thing, so there's quite a bunch of cool stuff here, so the one cycle implementation was done by our students yoga. I think i don't know how to pronounce his name exactly so where this the trains life, our

experiments were largely done by brett currents and stuff like getting the half position. Floating-Point implementation integrated into fastai, was done by Andrew Shaw, so it's been a cool kind of bunch of different student projects coming together to allow us to run this, so this is going to run actually on a AWS Amazon, AWS p3, which has eight GPUs the P3 has these newer Volta architecture GPUs, which actually have special support for half precision floating point. First, AI is the first library I know of to actually integrate the volatile, optimized half position floating point into the library. So we can just go, learn half now and get that support automatically and is also the first one to integrate one cycle.

So these are the parameters for the one cycle, so we can go ahead and get this running. So what this actually does is it's using pytorches multi-gpu support, since there are eight GPUs, it's actually going to fire off eight separate Python processors and each one's going to train on a little bit and then at the end, it's going to pass the gradient updates Back to kind of the master process, that's going to integrate them all together, so you'll see here they are right. Lots of

7. 00:40:00

- **Intro to next step: NLP and translation deep-dive, with CMU pronouncing dictionary**
- **via spelling_bee_RNN.ipynb**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Progress bars will pop up together and you can see it's training. You know three or four seconds when you do it this way. Where else when I had where else when I was training earlier, I was getting. Let's see 30 epochs in, I was getting about 30 seconds per clock so doing it. This way we can kind of train things like 10 times faster or so, which is pretty cool. Okay, so we'll leave that running. So you are asking about adaptive average pooling and I think specifically is what's the number one doing so. Normally, when we're doing average pooling - let's say: we've got four by four: let's say we did average pooling, comma okay, then that creates a 2x2 area and takes the average of those four right and then we can pass in the stride all right. So if we said stride one then the next one is we look at this block of 2x2 and take that average and so forth right, so that's like what a normal to go to average pooling would be, and so that would in that case, if we didn't Have any parting that would spit out a 3x3 okay, because it's true here to here to here, okay, and if we added padding, we can make it three by three as well. That's right! So if we wanted to spit out something, we didn't want 3x3. What? If we wanted one by one right, then we could say average pull four comma four right, and so that's gon na do 4, comma, 4 and average the whole lot right and that would spit out one by one. But that's just one way to do it rather than saying the size of the the kind of appalling filter.

Why don't we instead say? Well, I don't care what the size of the input grid is. I always want one by one right. So that's where, then you say: adaptive average Hall and now you don't say: what's the size of the palling filter, you instead say what's the size of the app. What I want - and so I want something that serve one by one and if you already put a single int, it assumes you mean one by one. So, in this case, adaptive average pooling one with a four by four grid coming in is the same as average pooling for Hummer four. If it was a 7x7 grid coming in, it would be the same as 7, comma 7 right. So it's the same operation. It's just expressing it in a

way that says regardless of the input. I want something of that sized output. Please, okay, how's our little thing going along: oh okay! Well, we got 294 and it took three minutes and 11 seconds and the previous state of the art was one hour and seven minutes. So was it worth fiddling around with those parameters and learning a little bit about how these architectures actually work and not just using what came out of the box well holy. We just used a publicly available instance. We use the spot instance, so it's cost you that so that cost us, like $ 8 per hour for three minutes - cost us a few cents to train this from scratch 20 times faster than anybody's ever done it before. So that's like the most crazy state at the at result.

I think we see we've seen many, but this one just blew it out of the water right, and so you know this is kind of partly thanks to just fiddling around with those parameters of the architecture. Mainly frankly, about using Leslie Smith's one cycle thing and swampers implementation of that, and remember not only so does to remind you of what that's doing. It's basically saying this is batches right, and this is mining right. Okay, it creates an upward path. That's equally long as the downward path right. So it's a true clr triangular cycle called learning rate. As per usual, you can pick. The ratio between those two numbers right so X, divided by Y in this case is, is a lumber that you get to pick in this case. We picked 50 okay, so we started out with a much smaller one here and then it's got this cool idea, which is you get to say what percentage of your epochs then is spent going from the bottom of this down all the way down pretty much to Zero and that's what this second number here is, so 15 percent of the batches are spent going from the bottom of our triangle even further. So importantly, though, with that's not the only thing one cycle does we also have momentum right and momentum goes from point. Nine. Five to 0.85 like this, in other words, when the loading rates really low, we use a lot of momentum and when the learning rates really high, we use very little momentum, which makes a lot of sense, but until Leslie Smith showed this in that paper, I've never Seen anybody do it before, so it's a really really cool trick, so you can now use that by using the UCLA beta parameter in in fastai, and you should be able to basically replicate this state-of-the-art result.

You can use it on your own computer or your paper space. Obviously, the only thing you won't get is the multi-gpu piece, but that makes it a bit easier to Train anyway, so in a single GPU, you should be able to beat this this. This on a single GPU yeah make group layer contains stride equals two. So this means stride is one for layer, one and two for everything else. What's the logic behind it, usually the strides I have seen our odd no strides are either one or two, I think you're thinking of kernel sizes, so strata calls to means that I jump to across and so stride of two means that you have your grid size. So I think you might've cos got confused between stride and kernel size there, and so, if we have a stride of one, the grid size doesn't change. If you have a stride of two, then it does - and so in this case, because this is for say, fire 10 32 by 32 is small and we don't get to have the grid size very often right, because pretty quickly we're going to run out of cells And so that's why the first layer has a stride of one. So we don't decrease the grid size straightaway, as you play, and it's kind of a nice way of doing it, because that's why we kind of have a low number here. So we can. We can start out with you know not too much computation on the big grid, and then we can gradually doing more and more computation as the grids get smaller and smaller, but because the smaller grid, the computation, will take less time.

Okay, so I think so that we can do all of our gaining in one go: let's take a slightly early break and come back at 7:30, okay, so we're going to talk about generative adversarial networks, also known as ganz, and specifically we're going to focus on the Vasa Stein gang caper, which included some guy called seumas chintala, who went on to create some piece of software

called hide watch. The Vasa Stein Gann, was heavily influenced by the Sun is going to call this w again, that's the time the DC gain or deep convolution deep convolutional generative adversarial networks paper, which also seemeth was involved with. So it's a really interesting paper to read a lot of it looks like this. The good news is, you can skip those bits because there's also a bit that looks like this, which says: do these things all right now. I will say, though, but like a lot of papers have a theoretical section, which seems to be they're entirely to get past the reviewers need for theory, that's not true with a W again paper. The theory bit is actually really interesting. Like you, don't need to know what to use it, but if you want to learn about like some some cool ideas and see the thinking behind why this particular algorithm, it's absolutely fascinating, and almost nobody before this paper came out.

I didn't know literally I'd Lou knew nobody who had studied the math that it's based on so like everybody had the method is based on ins, and so the paper does a pretty good job of laying out all the pieces. You have to do a bunch of reading yourself, so if you're interested like in digging into the deeper math behind some paper to see what it's like to study it. I would pick this one because at the end of that theory, section you'll come away saying like okay, I can see now why they made this algorithm the way it is okay and then having come up with that idea. Like the other thing is often, these theoretical sections are very clearly added after they come up with the algorithm. They'll come up the algorithm based on intuition and experiments and then later on, post hoc, justify it, or else this one. You can clearly see it's like. Okay. Let's actually think about what's going on in Ganz and think about what they need to do and then come up with the algorithm. So the basic idea of a began is it's a generative model? Okay, so it's something that is going to create sentences or create images. Kind of generate stuff right and it's going to try and create stuff, which is very hard to tell the difference between generated stuff and real stuff. That's what generative model could be used to face. What a video you know, a very well-known, controversial thing of deep fakes and fake pornography and stuff happening at the moment could be used to fake somebody's voice.

It could be used to fake the answer to a medical question, but in that case it's not really a fake right. It could be a generative answer to a medical question. That's actually a good answer right, so you like generating language. You could generate a caption to a to an image, for example, so generative models have lots of apple, interesting adaptations, but generally speaking they're, they need to be good enough that, for example, if you're using it too, you know automatically create a new scene for Carrie Fisher. In the next Star, Wars, movies and she's not around to play that part anymore, you want to, you, know, try and generate an image of her. That looks the same. Then it has to fool the Star Wars, audience into thinking like okay, that that doesn't look like some weird Carrie Fisher. That looks like the real Carrie Fisher or, if you're, trying to generate an answer to a medical question, you want to generate English. That reads, you know nicely and clearly - and it sounds authority of meaningful, so the idea of a generative adversarial Network is we're going to create not just a generative model to create, say the the the the generated image that a second model. That's going to try to pick which ones are real and which ones are generated and we're going to call them fake, okay, so which ones are real and which ones are fake. So we've got a generator, that's going to create our fake content and a discriminator.

That's going to try to get good at recognizing, which ones are real and which ones are fake, so they're going to be two models and then they're going to be adversarial, meaning the generator is going to treat a tree trying to keep getting better at falling. The discriminator into thinking that fake is real and the discriminator is going to try to keep getting better at discriminating

between the real and the fake and they're going to go head to head like that right and it's basically as easy as I just described. It really is, but it's going to build two models in play: torch we're going to create a training loop that first of all says the loss function for the discriminator is. Can you tell the density, real and fake and then update the weights of that and then we're going to create a loss function for the generator which is just going to say? Can you generate something which pulls the discriminator and update the weights? True from that loss and we're going to look through that a few times and see what happens, and so let's come back to the pseudo code here, if the algorithm and let's read the real code first, so there's lots of different things you can do with Gans And we're going to do something, that's kind of boring but easy to understand and and it's kind of cool that it's even possible, which is we're just going to generate some pictures from now. We're just going to get it to draw some pictures.

Okay and specifically, we're going to get it to draw pictures of bedrooms right, you'll,

8. 00:55:15

- **Create spelling_bee_RNN model with Keras**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Find if you hopefully get a chance to play around of this during the week with your own data sets, if you pick a dataset, that's very varied, like imagenet and then get again to try and create image net pictures. It tends not to do so well because it's it's not really clear enough. What do you want a picture of alright? So it's better to give it. For example, the there's a data set called celeb, a which has pictures of celebrities faces that works great with dance. You create really clear celebrity faces that don't actually exist. The bedroom data set also a good one right: lots of things, pictures of the same kind of thing. Okay, so that's just a suggestion. So there's something called the L son same classification data set. You can download it using these steps. I've also it's pretty huge, and so I've actually created a kegel data set of a 20 % sample right. So, unless you're really excited about generating bedroom images, you might prefer to grab the 20 % sample. So then we do the normal steps of creating some different paths, and in this case you know I, as we do before. I find it much easier to kind of grow the CSV route when it comes to handling our our data. So I just generate a CSV with the list of files that we want and a fake label of 0, because we don't really have labels for these.

At all and I actually create two part, two CSV files, one that contains everything in that bedroom data set and one that just contains a random 10 %. That's just nice to do that because then I can most of the time use the sample when I'm experimenting because like because there's well over a million files, just reading in the list takes a while okay. So this will look pretty familiar. So here's a con block. This is before I realized that sequential models are much better. So if you compare this to my previous con, lock with a sequential model, there's just a lot more lines of code here, but you know it does the same thing of doing con rally better on okay and we calculate our padding and here's a bias pulse. So this is the same as before, basically, but with a little bit more code, alright, so the first thing we're going to do is we're going to build a discriminator, so a discriminator is going to receive as input an image. Okay and it's going to spit out a number and the number is meant to be lower if it thinks this image is real. Okay, now, of course the what does it do for a lower number thing doesn't appear in the architecture, that'll be in the loss function. So what we have to do is create

something that takes an image and spits out a number okay. So a lot of this code is borrowed from the original authors of this paper of the paper. So some of the naming scheme and stuff is different to what we're used to so sorry about that.

But hopefully I've tried to make it look at least somewhat familiar. I probably should have renamed things a little bit, but it looks very similar to actually what we had before we start out with a convolution, so in become block is conquering a veteran on okay, and then we have a bunch of extra comm flares. This is not gon na use a residual right, so it looks very similar to before a bunch of extra layers, but these are going to become players rather than res layers and then at the end, we need to append enough stride to enough strive to calm players That we decrease the size, the grid size down to be no bigger than 4x4 right. So it's going to keep using straight to divide the size by twos, tried to divide by those by two and till our grid. Size is no bigger than four, and so this is quite a nice way of like creating as many layers as you need in a network to handle arbitrary sized images and turn them into a fixed loan grid size. Yes, Rachel does again need a lot more data than say dogs versus cats or NLP, or is it comparable? You know honestly, I'm kind of embarrassed to say I am NOT an expert practitioner in games, so you know the stuff i teach in part. One is stuff. I'm happy to say I know the best way to you know pretty close the best way to do these things, and so I can show you stayed at the out results like I just did with safe at ten, with the help of some of my students.

Of course, I'm not there at all with Ganz, so I'm not quite sure how much you need like in general Pitt seems unique quite a lot, but remember the only reason we didn't need too much and dogs and cats is because we had a pre trained model And could we leverage pre-trained and models and fine-tune them? Probably I don't think anybody's done it. As far as I know, that could be really interesting thing for people that are kind of think about an experiment with. Maybe people have done it and there's some literature there? I haven't come across, so I'm somewhat familiar with the main pieces of literature and Gans, but I don't know all of it. So maybe I've missed something about transfer learning in games, but that would be the trick to not needing too much data, and so it's the huge speed-up, a combination of one cycle, learning rate and momentum, annealing, plus the eight GPU parallel training in the half precision is That only possible to do the half precision calculation with consumer GPU. Another question: why is the calculation eight times faster from single to half precision while from double the single is only two times fast? Okay, so the SyFy? Ten result: it's not eight times faster from single to half it's about two or three times as fast from single to half the Nvidia claims about the about the flops performance of the tensor cause are academically correct, but in practice meaningless, because it really depends on what Calls you need for what pieces so about two or three X improvement for half so yeah.

The half precision helps a bit the the extra GPUs helps a bit. The one cycle helps an enormous amount. Then another key piece was the playing around with the parameters that I told you about so kind of ten reading the wide ResNet paper carefully identifying the kinds of things that they found there and then writing a version of the architecture you just saw that made. It really easy for me to fiddle around with prep one action for me for Brett pans to fiddle around with parameters staying up all night, trying every possible combination of different kernel, sizes and numbers of kernels and numbers of layer groups and size of layer groups and The amount of remember we did a bottleneck, but actually we tended to focus not on bottlenecks bonds instead on widening, so we actually like things that increase the size and then decrease it, because it takes better advantage of the GPU. So, though, all those things combined together, I'd say the one cycle was perhaps the most critical, but but every one of

those resulted in a big speed-up. That's why we were able to get this 30x improvement over the state-of-the-art so factor and we got some ideas for other things like after this Dorn bench finishes, and you know maybe we'll try and go even further see if we can beat one minute one day. That'll be fun okay, so so here's, okay, so here's our discriminator right.

I mean that the important thing to remember about an architecture is, it doesn't do anything rather than have some input, tensor size and rank and some output tensor size and range. So this is going to spit out. You see. The last calm here has one channel right. This is a bit different to what we're used to right, because normally our last thing is a linear block right. But our last thing here is a common block right and it's only got one channel, but it's got a grid size of something around 4 by 4, it's no more than 4 by 4, so we're gon na spit out a let's say: it's provoke or a 4 By 4 by 1 cancer, so what we then do is we then take the mean of that fat. So it goes from 4 by 4 by 1 to the scalar right. So this is kind of like the ultimate adaptive average pulling right, because we've got something with just one channel. We take the mean, so this is a bit yeah a bit different. Normally, we first do average pooling, and then we put it through a fully connected layer to get our one thing out. In this case, though, but getting one channel out and then taking the mean of with that, I haven't fiddled around with like. Why did we do it that way? What would instead happen? If we did the usual average Pauling, followed by a fully connected layer? Would it work better? Would it not I I don't know, I rather suspect it would work better if we did it like the normal way, but I haven't tried it and I don't really have a good enough intuition to know whether I'm missing something finished, the experimenter truck.

If somebody wants to stick an adaptive average Pauling layer here and a fully connected layer afterwards with a single output, it should keep working. You know we should do something. The loss will go down to see whether it works okay. So that's the discriminator right. So it's gon na be a training loop. Let's put, let's assume, we've already got a generator. Somebody says: okay, Jeremy, here's a generator, it generates bedrooms. Ok, I want you to build a model that can figure out which ones are real and which ones aren't. So I'm going to take the data set and I'm going to basically label a bunch of images which are fake bedrooms from the generator and a bunch of images of real bedrooms from my else. I'm data set to stick a 1 or a 0 and h1 and then I'll try to get there discriminator to tell the difference. Ok, so that's gon na be simple enough, but I haven't been given a generator. I need to build one, so a generator, and we haven't talked about the last function. Yet, okay, we're just going to assume that there's some loss function. That does this thing. Okay, so a generator is also an architecture which doesn't do anything by itself until we have a loss, function and data. But what are the ranks and sizes of the tenses? Well, the input to the generator is going to be a vector of random numbers. Okay and in the paper, they call that the prior, but it's going to be a vector of random numbers.

How big? I don't know some big 64 128 right and the idea is that a different bunch of random numbers will generate a different bedroom. Okay. So that's the idea, so our our gann generator sorry has to take as input a vector and it's going to take that vector. So here's our import all right and it's going to stick it through in this case so sequential model and the sequential models going to take that vector it's going to turn it into a 2 by 2. Sorry, a turn it into a well. I rank for tensor or if we take off the batch bit around three tensor height by width by three okay, so you can see at the end here our final step here, NC number of channels, so I think that's going to have to end up being 3 Equals going to create a 3 channel image of some size, yes, Rachel in coms block forward. Is there a reason? My batch dorm comes after Lu, ie self dot batch norm, dot self dot right? No, that's not it's just what they had

in the code. I borrowed from, I think that the order is reversed yeah. So again, unless my intuition about Dan's is all wrong and say some some reason need to be different to what I'm used to. I would normally expect to yeah. I would actually no sorry. I would normally expect to go rally your then batch norm that this is actually the order. That makes more sense to me, but I think the order I had in the darknet was what they used in the darknet paper.

So I don't know everybody seems to have a different order of these things and in fact most people for sci-fi 10 have a different order again, which is they actually go bien then rally then conf, which is kind of a quirky way of thinking about it. But it turns out that for often for residual blocks that works better, that's called a pre activation ResNet. So if you google for pre activation ResNet, you can see that so yeah there's there's a few, not so much papers, but more blog posts out there. Where people will have experimented with different orders of those things and yeah, it seems to depend a lot on what specific data said it is and what you're doing with, although in general, the difference in performance is small enough. You won't care unless it's through a competition. Okay, so the generator needs to start with a vector and end up with a rank. Three tensor. We don't really know how to do that yet. So, how do we do that? How do we start with a vector and turn it into a rank? Three tensor. We need to use something called a deconvolution and a deconvolution is, or as they call it, an fie torch, transposed, convolution same name, that's right same same thing, different name, and so a deconvolution is something which, rather than decreasing the grid size, it increases the grid size. So, as with all things, it's easiest to see in an Excel spreadsheet, so here's a convolution right.

We start, let's say with a four by four grid: cell: okay, with a single channel, a single photo and let's put it through a three by three kernel again with a single output filter. Okay, so we've got a single channel in a single filter kernel, and so, if we don't add any padding we're going to end up with two by two right, because that three by three you can go in one two. Three four places right can go in. One of two places across some one of two places down: if there's no padding. Okay, so there's our there's our convolution right, remember: the convolution is just the sum of the product of the kernel and the appropriate grid cell. So, there's a there's our standard 3x3 upon one channel one filter, so the idea now is. I want to go the opposite direction. I want to start with my 2x2 and I want to create a 4x4, and specifically, I want to create the same 4x4 that I started with, and I want to do that by using a convolution. So how would I do that? Well, if I have a 3x3 convolution, then if I want to create a 4x4 output, I'm going to need to create this much padding. That goes with this much padding, I'm going to end up with one two, three four by one, two, three four. You see why that is so. This filter can go at any one of four places across from four places up and down. So let's say my convolutional filter was just a bunch of zeros. Then I can calculate my my error for each cell just by taking this attraction and then I can get the sum of absolute values.

They are one loss later summing up the absolute values of those errors, all right so now I could use optimization so in Excel. That's called solver to do a gradient descent. Okay, so I'm going to set that cell equal to a minimum, try and reduce my loss by changing my filter. Okay and I'll go solve okay, and you can see it's come up with a filter such that you know fifteen point seven compared to sixteen seventeen. That's right, seventeen point it. So it's not perfect right and in general you can't assume that a deconvolution can exactly create the same. You know the exact thing that you want because there's just not enough, you know, there's only nine things here and there's sixteen things. You're trying to create right, but it's it's made a pretty good attempt. Okay. So this is what a deconvolution looks like asteroid, one 3x3 deep deconvolution on a 2x2 grid cell input. Did you have a question? How difficult is it to create a

discriminator to identify fake news versus real news? Well, you don't need anything special, that's just a classifier right, so you would just use the NLP classifier from previous previous to previous class and listen for right. It's there's! Nothing like it and in that case, there's no generative piece right, so you just need the data set. That says these are the things that we believe are fake news, and these are the things we consider to be real news and it should actually work very well.

You know like as to the best of our knowledge, if you, if you try it, you should get a you know as good a result, as anybody else has got whether it's good enough to be useful to practice. I don't know, as I say, that it's very hard very hard there's not a good solution that does that well, but I don't think anybody in our course has tried and nobody else outside our cause knows of this technique right so like there's, beaners we've as we've Learned we've just had a very significant jump in NLP classification capabilities and yeah. I mean I think it's obviously the best you could do. I think at this stage would be to generate a kind of a triage that says these things look pretty sketchy based on how they're written and then you know some human could go and in fact check them. You know I mean an NLP classifier and I run in kind of fact-check things, but it could recognize like all these. These are written in that kind of you know that kind of highly popularized style, which often fake news is written in, and so maybe these ones are worth paying attention to. I think that would probably the best you could hope for without drawing on some kind of external data sources yeah. But it's important to remember you know the discriminator is basically just a classifier and you don't need any special techniques beyond what we've already learned to do.

An LP classification, ok so to to do that kind of deconvolution in in pytorch, just say: chrome, transporters, 2d and in the normal way you say the number of input channels, the number of output channels, the kernel size, the stride, the padding the bias. So these parameters are all the same right and the reason it's called a comm transpose is because actually it turns out that this is. This is the same as the calculation of the gradient of convolution. That's that's! That's, basically, why they call it that. So this is a really nice example back on the older Theano website. That comes from a really nice paper, which actually shows you some visualizations. So this is actually the one we just saw of doing a 2x2 deconvolution. If there's a stride to, then you don't just have padding around the outside, but you actually have to put padding in the middle as well. Okay, they're not actually quite implemented this way, because this is slow to do in practice you in a different way, but it all happens behind the scenes. We don't have to worry better okay yeah this we've we've talked about this convolution arithmetic tutorial before and if you're still not comfortable with convolutions and in order to get comfortable with D convolutions. This is a great site to go to. If you want to see the paper, just Google for convolution arithmetic it'll be the first thing that comes up.

Let's do it now, so you know you've found it.

9. 01:17:30

- **Question: "Why not treat text problems the same way we do with images' ? "**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Here it is, and so that Theano tutorial actually comes from this paper, but the paper doesn't

have the animated gifs okay, so it's interesting there, no decomp block looks identical to a comp lock except it's got the word transpose written here. Okay, we just go comb for Elliot better on this before it's about input, filters, output, fit filters. The only difference is that stride 2 means that the grid size will double. Rather than half both nn comp transpose to D and n n dot up sample seemed to do the same thing: ie expand, grid, size, height and width from the previous layer. Can we say that Kampf transpose to D is always better than up sample since up sample? Is merely resizing and filling unknown unknowns by zeros or interpolation? No, you can't so there's a fantastic interactive paper on distilled pub called deconvolution and checkerboard artifacts, which points out that what we're doing right now is extremely suboptimal, but the good news is everybody else. Does it if you have a look here, could you see these checkerboard architect artifacts like it's all like that blue light, blue dark, blue light blue, you know you kind of, and so these are all from from actual papers right and basically they noticed every one of These papers, with generative models, has these checkerboard artifacts and what they realized is it's because, when you have a stride to convolution of size, three kernel, they overlap right, and so you basically get like some pixels get twice as much kind of active some grid cells.

I guess wise as much activation, and so even if you start with random weights, you end up with a check of what artifact. So you can kind of see it here right and so the deeper you get kind of the worse it gets. Their advice is actually less director than it ought to be. I've found that for most generative models up sampling is better right. So all if you do end up sample, then all it does is it's. It's basically doing pooling right. Basically, but it's kind of its kind of the opposite of Pauline right. It says: let's replace this one pixel or this one grid cell with four two by two and there's a number of ways to up sample. One is just to kind of copy it across to those four and other is to use kind of lit by linear or bicubic interpolation. There are various techniques to kind of try and create a smooth off sample version, and you can pretty much choose any of them in pytorch. So if you do a two by two up sample and then a regular stride, one three by three khans, that's like another way of doing the same kind of thing as a commons plus. Well, it's it's doubling the grid size and doing some convolutional arithmetic on it, and i found for generative models and it pretty much always works better and in that distorter pub publication. They kind of indicate that maybe that's a good approach, but they don't just come out and say just do this, whereas i would just say just do this.

Having said that for gains, i haven't had that much success with it yet, and I think it probably requires some tweaking to get it to work, as I'm sure some people have got it to work. The the issue, I think, is that in the earliest ages it doesn't create enough noise ahead. I don't have it here. I had a version actually where I tried to do it within an up sample and you could kind of see that the noise didn't look very noisy so anyway, it's an interesting question, but um next week, when we look at style, transfer and super-resolution and stuff, I Think you'll see and ended up sample really comes into its own okay, so the generator we can now basically start with the vector we can decide and say, like okay, does not think of it as a vector, but actually it's a one by one grid cell and Then we can turn it into a four by four and they by eight and so forth, and so that's why we have to make sure it's a it's a suitable multiple, so that we can actually create something of the right size. And so you can see it's doing the exact opposite as before, right it's making the cell size smaller and smaller by two at a time as long as it can sorry, bigger and bigger, so this cell size, bigger and bigger, as long as it can until it Gets to half the size that we want and then finally we add one more on at the end.

Sorry, we add n more on at the end of just with no stride and then we add one more calm transpose to finally get to the size that we wanted and we're done. Finally, we put that through

a fan and that's got a force us to be in the zero to one range, because of course we don't want to spit out arbitrary size, pixel values. Okay, so that's so, we've got to generate our architecture, which spits out an image of some given size, with the correct number, with the correct, whatever a readout of all correct number of channels and with values between zero and one. So at this point we can now create our model data object. These things take a while to train, so I just made it 128 by 128. So this is just a convenient way to make it a faster and that's going to be the size of the input, but then we're going to use transformations to turn it into 64. By 64, okay, there's been more recent advances which have attempted to really increase this up to kind of like high resolution sizes, but they still tend to require either like a batch size of 1 or like lots and lots of GPUs or whatever. So we're kind of trying to do things that we can do 1 conceal all consumer, GPUs yeah, so here's an example of one of the 64 by 64 bedrooms. Okay, so we're gon na do if pretty much everything manually. So let's go ahead and create our two models: our generator and our discriminator and, as you can see, they're DC can so, in other words, they're. The same modules that came up were appeared in this paper.

So, if you're interested in reading the papers, you it's well worth going back and looking at the DC gain paper to see what these architectures are, because it's assumed that when you read the vasa stein gain paper that you already know that, yes, you don't. We use a sigmoid if we want values between 0 and 1, I always forget which ones which okay so sick - yes, so sigmoid is 0 to 1 fan is 1 to minus 1. I think what will happen is I'm gon na have to check that? I beg you remember thinking about this when I was writing this notebook and realizing that 1 2 minus 1 made sense for some reason, but I can't remember what that reason was now. So let me get back here about that during the week and remind me if I forget it's good question. Thank you. Ok, so we've got our generator in a discriminator, so we need a function that returns a prior vector, so a bunch of noise. So we do that by creating a bunch of

10. 01:26:00

- **Graph for Attentional Model on Neural Translation**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Zeros NZ is the size of Zed, so like very often in our code. If you see a mysterious letter, it's because that's the letter they used in the paper that says ed is the size about noise, vector okay, so there's the size of our noise vector and then we use a normal distribution to generate random numbers inside that and that Needs to be a variable because it's going to be participating in the in the gradient updates, so here's an example of creating some noise, and so here are four different pieces of noise. Okay, so we need an optimizer in order to update our gradients in the vasa stein game paper. They told us to use rmsprop. So that's fine! That's! So! When you see this thing saying, do an rmsprop update in a paper. That's nice! We can just do an rmsprop update with pipe watch, okay and they suggested a learning rate of five Enid five. I think I found one a nigga four seem to work, so I was made a little bit bigger. So now we need a training loop, and so this is the thing that's going to implement this algorithm, so a training loop is going to go through some number of epochs that we get to pick. So that's going to be a parameter, and so remember when you do everything manually, you've got to remember all the manual steps to do so. One is that you have to set your modules into training mode when you're training them and into evaluation mode when you're evaluating them, because in training mode batch norm updates happen and dropout, happens in evaluation

mode.

Those two things get turned off: okay, so it's basically difference. So put it into training mode, we're going to grab a iterator from our training data loader we're going to see how many steps we have to go through and then we'll use two qdm to give us a progress bar and then we're going to go through that. Many steps, okay, so the first step of this algorithm is to update the is to update the discriminator. So in this one I'm just trying to remember yes, they don't call it a discriminator, they call it a critic right. So W are the weights of the of the critic, so the first step is to train our critic a little bit and then we're going to train our generator a little bit, and then we go go back to the top of the loop right. So this inner. So we got a while loop on the outside okay, so here's our while loop on the outside and then inside that there's another loop for the critic and so here's a little loop inside that for the critic. Okay, we call it a discriminator. So what we're going to do now is we're going to try. We've got, we've got a generator and at the moment it's random right, so our generator is going to generate stuff. That looks something like this right, and so we need to first of all teach our discriminator to tell the difference between that and a bedroom right, which shouldn't be too hard. You would hope so we just do it in basically the usual way, but there's a few little tweaks.

So, first of all, we're going to grab a mini batch of real bedroom photos, so we can just grab the next batch from our iterator turn it into a variable. Okay, then we're going to calculate the loss for that right. So this is going to be. How much does the discriminator think this looks? This looks fake right through the real ones book fake and then we're going to create some fake images and to do that will create some random noise and we'll stick it through our generator, which at this stage is just a bunch of random weights. Okay and that's going to create a mini batch of fake images, okay and so then we'll put that, through the same discriminator module as before, okay to get the loss for that. So how fake to the fake ones look remember when you do everything manually, you have to zero the gradients in in your loop and, if you've forgotten about that, go back to the part. One lesson where we do everything from scratch. So now, finally, the total discriminator loss is equal to the real loss, the fake loss, okay, and so you can see that here they don't talk about the loss. They actually just talk about one of the gradient updates, so this here is a symbol for get the gradients alright, so inside here is the loss right and like trying to like learn to throw away in your head all of the boring stuff. So when you see sum over m / M, that means take the average so just throw that away and replace it with NP, don't mean in your head. There's another NP domain right.

So you want to get quick at like being able to see these common idioms. So anytime, you see one over m sum over m. You go okay and peed on me right so we're taking the mean of and we're taking the mean of. So that's all fine X, I what's X, I it looks like it's next to the power of I, but it's not right. The math notation is very overloaded. They showed us here what X I is, and it's a set of M samples from a batch of the

11. 01:32:00

- **Attention Models (cont.)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Real data, so, in other words this is a mini batch right. So when that, when you see something

same sample, it means just grab a row grab a row grab a row and you can see here, grab it m times and we'll call the first row X, parentheses, 1, the second row X, parentheses, 2. One of the annoying things about math notation is the the way that we index into arrays is everybody uses different approaches, subscript superscript things in brackets, combinations, commas square brackets, whatever right so you've just got to look in the paper and be like ok at some point. They're going to say, take the I throw from this matrix or the ithe image in this batch. How are they going to do it in this case? Let's say superscript in parentheses? Okay, so that's all sample means and curly brackets means it's just a set of them. This little squiggle, followed by something here, means according to some probability, distribution, and so in this case like and very very often in papers, it simply means hey. You've got a bunch of data right grab a bit from it at random. Okay, so that's like that's the the probability distribution of the data you have is the data you have right, so this says: grab em things at random from your real data. This says: grab em things at random from your prior samples, and so that means, in other words, call create noise to create m random vectors. So now we've got em real images. Each one gets put through our discriminator. We've got em bits of noise.

Each one gets put through our generator to create em generated images. Each one of those gets put through. Look FW, that's the same thing, so these ones of those gets put through our discriminator to try and figure out whether they're, fake or not. And so then it's this minus this and the mean of that and then finally get the gradient of that in order to figure out how to use our own s prop to update our weights using some learning rate. Okay, so in height or CH, we don't have to worry about getting the gradients. We can just specify their last bit. Okay and then just say: lost, stop backward discriminator, optimizer, diet, step. Okay, now there's one key step right, which is that we have to keep all of our activations, so all of our weights, which are the parameters in a page horch module in this small range between 0.01 negative 0.01 and point out one. Why? Because the mathematical assumptions that make this algorithm work only only apply in like a small ball all right, so I'm not going to tell I don't. I think it's kind of interesting to understand the math of why that's the case, but it's very specific to this one paper and understanding it won't help. You understand any other paper, so only study it. You know if you're interested in you know, I think it's nicely explained. I think it's fun, but it won't be information that you're reuse elsewhere.

Unless you get super into dance I'll also mention after the paper came out and improved froster stein, Gann came out. That said, hey there are better ways to ensure that your that your weight space is in this tight ball, which was basically that kind of penalize gradients that are too high, so actually nowadays, they're at there is slightly different ways to do this anyway. That's why this line of code there it's kind of the key contribution this you know this one line of code actually is the one line of code. You add to make it a versus time again, basically that the work was all in knowing like that. That's the thing that you can do that makes everything work better. Ok, so, at the end of this, we've got a discriminator that can recognize an industry in real bedrooms and our totally random crappy generated images. So, let's now try and create some better images so now set trainable discriminator to false set trainable the generator to true zero out the gradients of the generator, and now our loss again is fw. That's the did that remember. That's the discriminator of the generator applied to some more random noise. Okay, so here's our random noise, here's our generator, here's our discriminator. I think I can remove that now because I think I've put it inside the discriminator, but I won't change it now, because it's going to confuse me so it's

12. 01:37:20

- **Neural Machine Translation (research paper)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Exactly the same as before, where we did generator on the noise and then pass a discriminator, but this time the thing that's trainable is the generator, not the discriminator, so in other words, in this pseudocode, the thing they update is theta, which is the generators parameters rather Than W, which is the discriminators parameters - and so hopefully you'll see now that this this W down here is telling you these are the parameters of the discriminator. This theta down here is telling you. These are the actually better these. This theta here is telling you. These are the parameters of the generator okay, again, it's not as universal mathematical, notation, it's a thing they're doing in this particular paper, but it's kind of nice. When you see some some suffix like that, there's like try to think about what it's telling you. Okay, so it takes noise, generate some images, try and figure out if they're, fake or real, and use that to get gradients with respect to okay, the generator as opposed to earlier. We got them with respect to the discriminator and use that to update our weights with our s prop with an alpha learning rate. Okay, you'll see that it's kind of unfair that the discriminator is getting trained and critic times which they set to five for every time that we train the generator once and the paper talks a bit about this.

But the basic idea is like there's no point making the generator better if the discriminator doesn't know how to discriminate yet okay. So that's why we've got this while loop and here's that v right and see something which was added. I think in the later paper, or maybe a supplementary material - is the idea that from time to time and a bunch of times at the start, you should do more steps at the discriminator so kind of make sure that the discriminator is pretty capable from time to Time: okay, so do a bunch of epochs of training, the discriminator a bunch of times to get better at telling the dentistry in real and fake and then do one step of making the generator being better at generating. And that is an epoch. And so let's train that for one epoch and then let's create some noise, so we can generate some examples actually going to do that later. Let's first of all decrease the learning rate by 10 and do one more pass. So we've now done two epochs and now let's use our noise to pass it to our generator, okay and then put it through our denormalization to turn it back into something we can see and then plot it, and we have some bedrooms: okay, there's not real bedrooms And some of them don't particularly like bedrooms, but some of them look a lot like bedrooms. So that's that's the idea.

Okay, that's again, and I think, like the best way to think about again, is it's like an underlying technology that you'll probably never use like this, but you'll use in lots of interesting ways. For example, they're going to use it to create now a cycle game and we're going to use the cycle Gann to turn horses into zebras. You could also use it to turn mono prints into photos or to turn photos of Yosemite in summer into winter. So it's gon na be pretty yes, Rachel, two questions, one. Is there any reason for using rmsprop, specifically as the optimized optimizer, as opposed to Adam? I don't remember it being explicitly discussed in the paper. I don't know if it's just experimental or the theoretical reason yeah have a look in the paper and see what it says I don't recall and which could be a reasonable way of detecting overfitting, while training or of evaluating the performance of one of these Gann models. Once we are done training, in other words, how does the notion of train validation test sets translate to Ganz? That's an awesome question and there's a lot of people who make jokes about how Cannes is the one field where you don't need a test set and people take advantage of that by making stuff up and

saying it looks great. There are some pretty famous problems with with ganz.

One of the famous problems with ganz is called mode collapse and mode collapse happens where you look at your bedrooms and it turns out that there's basically only three kinds of bedrooms - that every possible noise vector, mapped or you look at your gallery and it turns at All that turns out they're all just the same thing or there's just three different things. Mode collapse is easy to see. If you collapse down to a small number of modes, like you know three or four, but what if you have a mode collapse down to 10,000 modes, so there's only 10,000 possible bedrooms that all of your noise vectors collapse to that's not like you, wouldn't be able To see it here right because it's pretty unlikely, you would have two identical bedrooms out of 10,000 or what? If every one of these bedrooms is basically a direct copy of what? If the basically had never memorized some input, you know, could that be happening and the truth is most papers, don't do a good job or sometimes any job of checking those things. So the question of like how do we evaluate ganz, and even like the point of like hey, maybe we should actually evaluate games properly, is something that is not widely enough understood even now, and some people are trying to

13. 01:44:00

- **Grammar as a Foreign Language (research paper)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You know really push so Ian Goodfellow who a lot of you will know, because he came and spoke here at a lot of the book club meetings last year and of course was the first author on the most famous deep learning book. He is the inventor of games and he's been sending a continuous stream of tweets, reminding people about the importance of testing dance properly. So yeah. If you see a paper that claims exceptional, gaen results, then this is definitely something to look at. You know is: have they talked about mode collapse? Have they talked about memorization, okay, so um? This is going to be really straight forwards because it's just a neural net right. So all we're going to do is we're going to create an input containing lots of zebra photos and with each one we'll pair it with an equivalent horse photo and we'll just train a neural net. That goes from one to the other, or you could do the same thing for every Monet painting create a dataset containing the photo of the place. Oh wait: that's not possible, because the places that Monet painted aren't there anymore and there aren't exact zebra versions of horses and oh wait. How the hell is this kind of work. This seems to break everything we know about what neural nets can do and how they do them. Alright, make sure you know ask me a question. Just spoil a whole train of thought. Come on very good.

Can Ganz be used for data augmentation yeah? Absolutely you can use a gain for data augmentation should you I don't know like there are some papers that try to do semi-supervised learning with ganz I haven't found any that are like particularly compelling showing state-of-the-art results on really interesting data sets that have been widely studied. I'm a little skeptical and the reason I'm a little skeptical is because, in my experience, if you train a model with synthetic data, the neural net will become fantastically good at recognizing. The specific problems of your synthetic data and that'll be ended up what it's learning from, and there are lots of other ways of doing semi-supervised models which do work. Well. There are some places that can work. For example, you might remember Octavio gawd

created that fantastic visualization in part, one of like the zooming ComNet, where it kind of showed or let her go through em list. He at least at that time had the was the number one autonomous remote-controlled car. Okay, I mean in in autonomous remote control car competitions and he trained his model using synthetically, augmented data where he basically took real videos of a car driving around the circuit and added like fake people and fake other cars and stuff. Like that - and I think that worked well because well a because he's kind of a genius and B, because I think he had a kind of a world to find kind of little subset that he had to work in but yeah in general. It's really hard.

It's really really hard to use synthetic data. I've tried using synthetic data and models for decades now, obviously not ganz cuz they're pretty new, but in general it's very hard to do very interesting research question all right, so somehow these folks at Berkeley created a model that can turn a horse into a zebra. Despite not having any photos unless they went out there and painted horses and took before-and-after shots, but I believe they didn't right, so how the hell did they do this? It's it's kind of it's kind of genius. I will say the person I know who's doing the most interesting practice of cycle. Gann right now is one of our students, Elena sarin she's, the only artist I know of who was a psychic, an artist. Here's an example: I love she created this little doodle in the top of left and then trained a psycho Gann to turn it into this beautiful painting in the bottom row. Here's some more of her amazing works, and I think it's really interesting, like I I mentioned at the start of this class that, like Dan's, are in the category of like stuff: that's not there yet, but it's nearly there and in this case, like there's at least One person in the world now who's creating beautiful and extraordinary artworks using gas and there's lots of pairs, specifically cycle games and there's actually like at least maybe a dozen people I know of who are just doing interesting.

Creative work with neural nets more generally and the field of creative area is going to expand dramatically, and I think it's interesting with Elena right. I mean I don't know her personally, but from what I understand of her background. She's a you know: she's a software developer. You know it's her full-time job and an artist as her hobby and she's kind of started. Combining these two by saying gosh. I wonder what this particular tool could bring to my art, and so, if you follow her Twitter account we'll make sure we add it on the wiki. Somebody can find it as for Lena sarin sarin. She basically posts a new almost every day and they're, always pretty amazing. So here's the basic trick. Okay - and this is from the cycle Gann paper - we're going to have to kind of two images assuming we're doing this with images right, but the key thing is they're, not paired images. So we're not. We don't have a data set of horses and the equivalent zebras we've got bunch of horses, bunch of zebras grab. One horse grab one zebra: okay, we've now got an X, so X, let's say X, is horse and Y is zebra. We're going to train a generator and what they call here. A mapping function that turns horse into zebra, we'll call that mapping function, G and we'll create one mapping function, generator that turns a zebra into a horse and we'll call that F well create a discriminator. Just like we did before, which is going to get as good as possible at recognizing real from fake horses, so that'll be DX and then another discriminator which is going to be as good as possible and recognizing real from fake zebras I'll call that dy okay.

So that's kind of our starting point, but then the key thing to making this worse work. Okay, so we're kind of generating a loss function here right. Here's one bit of the loss function here. The second bit of the loss function we're going to create something called cycle, consistency loss which says after you turn your horse into a zebra with your G generator and check whether or not I can recognize that it's real. So I keep forgetting which one's false and which one zebra I apologize. I forget my X's and Y's backwards, though my horse into a zebra and

then going to try and turn that zebra back into the same horse that I started with okay and so then I'm going to have another function. That's going to check whether my this horse, which are generated knowing nothing about generated entirely from this zebra, is similar to the original horse or not right. So the idea would be if your generated zebra doesn't look anything like your original horse. You've got no chance of turning it back into the original horse, so a loss which compares X hat to X is going to be really bad unless you can go into Y and back out again and you're, probably only going to be able to do that. If you're able to create a zebra that looks like the original horse so that you know what the original horse looked like and vice versa, take the original.

Take your zebra turn it into a fake horse and check that you can recognize that and then try and turn it back into the original zebra and check that it looks like the original so notice here. This F right is our zebra two horse. This G is our horse, two zebra right, so so the G and the F are kind of doing two things they're both turning the original horse into the zebra and then turning the zebra back into the original horse. Okay, so notice that there's only two generators right: there isn't a separate generator for the reverse mapping. You have to use the same generator that was used for the original mapping. Okay, so this is the cycle consistence you lost, and I just think this is like this is genius. You know like the idea that this is a thing that could be even be possible honestly when this came out. It just never occurred to me as a thing that I could even try and solve. It seems so obviously impossible and then the idea that you can solve it like this. I just think it's it's so damn smart. So it's good to look at the equations in this paper because they're just good example like they're written pretty simply I you know, there's it's not like some of the stuff in the fastest time game paper, which is just like lots of theoretical proofs and whatever else In this case they're, you know they're just equations that just lay out what's going on and you really want to get to a point where you, where you can read them and understand so like let's kind of start talking through them.

So we've got a horse and a zebra okay. So for some mapping function, G, okay, which is our horse zebra mapping function. Then there's AG and loss right, which is the bit we're already familiar with. It says I've got a horse, a zebra, a fake zebra recognizer and a horse zebra generator. Okay and the loss is except it's what we saw before it's our ability to draw one zebra out of our zebras okay and recognize whether it's real or fake, okay and then generate a take a horse and turn it into a zebra and recognize whether that's real Or fake, okay and then you're, you then do one minus the other, and in this case they've got a log in there. The logs not terribly important. So this is this is the thing we just saw. That's that's why we did for sustained gain. First is this is just a standard gann loss in math form? Did you have a question right wrong? All of this sounds awfully like translating in one language to another, then back to the original, have gans or any equivalent been tried in translation. Not that I'm not that I know of , yeah yeah, because there's the Unversed is this: this unsupervised machine translation, which does kind of do something like this, but I don't. I haven't looked at it closely enough to know if it's nearly identical or if it's just vaguely similar yeah did so to kind of back up to what I do know normally with translation, you require this kind of paired input.

You require parallel text, so you know this is the French translation of this English? I dunno there's been a couple of recent papers that show the ability to create good quality translation models without paired data. I haven't implemented them and I don't understand anything I haven't implemented but yeah. They may well be doing the same basic idea, we'll look at it during the week and get back to you. Okay, all right, so we're going to again loss. The next piece is the cycle, consistency, loss right, and so the basic idea here is that we start with our horse, use our zebra generator on that to create a zebra use our horse generator on that to

create a horse and then compare that to the original Horse and this double lines were the one we've seen this before this is the l1 loss? Okay, so this is the sum of the absolute value of differences. Where else. If this was a turn, it would be the l2 loss or the two norm, which would be the sum of squared differences, the square root of it actually and again. We now know this squiggle idea, okay, which is from our horses, grab a horse. Okay, that's so this is what we mean by sample from a distribution. There's all kinds of distributions, but most commonly in these papers were using an empirical distribution. In other words, we've got some rows of data grab a row okay.

So when you see this thing, squiggle other thing this thing here when it says P data that means grab something from the data and we're going to call that thing X. So from our horses pictures grab a horse turn it into a zebra turn it back into a horse, compare it to the original and some of the absolute values. Okay, do that for horse to zebra. Do it for zebra to horse as well add the two together, and that is our cycle consistency. You lost okay, so now we get our loss function and the whole loss function depends on our horse generator a zebra generator our horse, recognizer, our zebra recognizer discriminator and we're going to add up the gain loss for recognizing horses, the gain loss for recognizing zebras and The cycle cycle consistency, loss for our two generators, okay and then we've got a lambda here, which hopefully we're kind of used to this idea. Now that is when you've got two different kinds of loss. You chuck in a parameter there. You can multiply them by so they're of account about the same scale, okay, and we did a similar thing with our bounding box loss compared to our classifier loss, when we did that localization stuff. Okay, so then we're going to try to for this map for this loss function maximize the capability of the discriminators are to discriminate discriminating whilst minimizing that for the generators, so the generators and the discriminators are going to be facing off against each other.

So when you see this min max thing in papers, you'll see it a lot. That means it basically means this idea that in your training loop, one thing is trying to make something better. The other is trying to make something worse and you generally there's lots of ways to do it, but most commonly you'll alternate between the two and you'll often see this just referred to in math papers as minnows. Okay, so we see min max, you think you should immediately think okay adversarial training. So let's look at the code and we're only going to probably might be able to finish this today, but we're going to do something almost unheard of which is. I started looking at somebody else's code and I was not so disgusted that I threw the whole thing away and did it myself. I actually said I quite like this. I like it enough, I'm going to show it to my students, so this is where the code comes from. So this is one of the people that created the original code for cycle games and they've created a high-torque version, and I had to clean it up a little bit, but it's actually pretty damn good. It's I think, the first time I found code that I didn't feel the need to rewrite from scratch before I showed it to you, and so the cool thing about this is one of the reasons I liked doing it this way of like finally, finding something: that's Not awful is that you're now going to get to see almost all the bits of fastai or, like all the relevant bits of fastai written in a different way about somebody else right and so you're going to get to see like oh how they do.

Data sets and data loaders and models and training, loops and so forth. Okay, so you'll find there's a C game directory, which is basically nearly this with some cleanups, which I hope to submit as a PR sometime. It was written in a way that unfortunately made it a bit over connected to how they were using it as a script, but I so cleaned it up a little bit, so I could use it as a module, but other than that. It's pretty similar. So C can is basically their code copied from their from their github repo, with some minor changes. So the way the second mini

library has been set up is that the configuration options they're assuming are being passed into like a script. So they've got this train options. Parser method, and so you can see I'm basically past passing in an array of like script options: okay, where's, my data: how many threads do I want to drop out? How many iterations? What am I going to call this model rich GPU? Okay? So that gives me a opt object which you can then see. Well, you know what it contains. You'll see that it contains some things I didn't mention. That's because it's got defaults for everything else that I didn't mention. Okay, so we're going to, rather than using fastai stuff, we're going to largely use CG and stuff. So the first thing we're going to need is a downloader, and so this is also a great opportunity for you again to practice your ability to navigate through code with your editor or IDE of choice, so we're going to start with create data loader. So you should be able to go, find symbol or in vim tag, to jump straight to create data loader, and we can see that's creating a custom data set loader and then we can see custom data set.

Loader is a base data loader, okay, so that doesn't really do anything. It creates okay. So basically we can see that it's going to use a standard, pytorch data loader. So that's good, and so we know if you're, to use a standard plate or data loader. You have to pass it a data set and we know that a data set is something that contains a length and a an indexer. So, presumably, when we look at create data set, it's going to do that here is create data set okay, so this library actually does more than just cycle gain it handles both aligned and unaligned image pairs. We know that our image pairs are unaligned, so we're going to an online data set okay here it is and as expected, it has a get item and a length good, and so the main, obviously the main the length is just whatever of our. So a and B is our horses and zebras got two sets. So, whichever one is longer is the length of the data loader and so getitem is just going to go ahead and randomly grab something from each of our two horses and zebras. Open them up with pillow or PIL, run them through some transformations, and then we could either be turning horses into zebras or zebras into horses. So there's some direction and then I'll just go ahead and then return our horse and our zebra and our path to the boss and the path of zebra so yeah. Hopefully, you can kind of see that this is looking pretty similar to the kind of stuff that fastai does first day.

I obviously does quite a lot more when it comes to transforms and performance and stuff like this, but you know remember: this is like research code for this one thing like it's pretty cool that they did all this work, so we've got a data loader, so we Can go and load our data into it, and so that'll tell us how many mini batches are in it. That's the length of the data loader in Piper watch next step. We've got a data. Loader is to create a model, so you can go to go tag for create model there. It is okay, same idea: we've got different kinds of models, so we're gon na be doing a cycle gain. So here's our cycle gain model. Okay, so there's quite a lot of stuff in a cycle gain model, so let's go through and find out what's going to be used, but basically at this stage we've just called initializer, and so when we initialize it, you can see it's going to go through and It's going to define two generators, we're just not surprising a generator for our horses and a generator for a zebras here. Okay, there's some way for it to generate a pool of fake data and then here we're going to grab our our Dan loss and, as we talked about our cycle, consistency loss is an l1 loss. That's interesting, they're going to use Adam so sleepy, recycle cans can Adam works pretty well, and so then we're going to have an optimizer for our horse, discriminator and optimizer for our zebra discriminator and an optimizer for our generator.

Okay, the optimizer for the generator is going to contain the parameters both for the horse generator and the zebra generator all in one place. So, okay, so the initializer is going to set up all of the different networks and wasp functions. We need and they're all going to be stored

inside this model, and so then it prints out and shows us exactly the piped watch bottles we have, and so it's interesting to see that they're using resonance, and so you can see the resinates look pretty familiar. We've got con vet norm rally, you can fetch norm so instance. Norm is just the same as batch norm. Basically, but it applies it to one image at a time and the difference isn't particularly important. Okay and you can see they're doing on reflection, padding just like we are, so you can kind of see like when you when you try to build everything from scratch like this. It is a lot of work and you know you can kind of forget the little. You know the nice little things that fastai does automatically for you. You kind of have to do all of them by hand, and only you end up with a subset of them. So you know over time, hopefully soon we'll get all of this Gans stuff into fast. Ai and it'll be nice and easy okay, so we've got our model and remember the model contains the loss functions. It contains the generators it contains, the discriminators all in one convenient place, so I've gone ahead and kind of copied and pasted and slightly refactored the training loop from their from their code so that we can run it inside the notebook.

So this one a lot pretty familiar right: loop to go through each epoch and a loop to go through the dealer. Before we did this, we set up a tower. This is actually not a play torch data set. I think this is what they used slightly confusingly to talk about their you know their combined. What we would call a model data object. I guess all the data that they need whip through that with TQ DM to get a progress bar and so now we can go through and see what happens in the model right so set input so set input. So so it's kind of a different approach to what we do in fastai. This is kind of neat. You know it's quite specific to cycle games, but basically, internally inside this model. Is this idea that we're going to go into our data and grab? You know we're knowing when we're either going horse to zebra or zebra to horse, depending on which way we go. We are the you know, a is over the horse or the zebra and vice versa, and, if necessary, put it on the appropriate GPU and then grab the appropriate paths. Okay. So the model now has a mini batch of horses and a mini batch of zebras, and so now we optimize the parameters. Okay, so it's kind of nice to see it like this. You can see each step right. So, first of all try to optimize the generators. Then try to optimize the horse discriminator then try to optimize the zebra discriminator. Zero grad is part of pipe torch.

Step is part of pytorch, so the interesting bit is the actual thing that captain twitch does the backpropagation on the generator. So here it is and let's jump to the key pieces, there's all the bits or the formula that we basically just saw yeah the paper. So let's take a horse and generate a zebra. So we know what a fake zebra and let's now use the discriminator to see if we can tell whether it's fake or not okay, so it spread fake and then, let's pop, that into our loss function, which we set up earlier to see if we can basically To get a loss function based on that prediction, then let's do the same thing to do. The gain loss so take a go in the opposite direction and then we need to use the opposite discriminator and then put that through the loss function again and then let's do the cycle. Consistence, you loss. Okay, so again we take our fake, which we created up here, okay and try and turn it back again into the original and then let's use that last function of cycle consistency, loss function. We created earlier to compare it to the real original and here's that lambda right, so there's some weight that we used and that would set up. Actually, we just use the default that I suggested in there options and then do the same for the opposite direction and then add them all together do the backward step and that's it.

So we can then do the same thing for the first discriminator and since basically, all the works being done now, there's much less to do here, okay, so there. That is so. I won't step all through it, but it's basically the same same basic stuff that we've already seen so optimized

parameters basically is calculating the losses and doing the optimizer step from time to time, save and print out some results and then, from time to time, update the Learning rate so they've got some learning rate annealing built in here as well, isn't very exciting, but okay, so they've basically got some kind of like fastai they've got this idea of schedulers, which you can then use to update your learning rates. So I think kind of, like you know, for those of you, are interested in better understanding, deep learning, api's or interested in contributing more to fastai or interested in like creating. You know your own version of some of this stuff in some different back-end. It's cool to like look at a second kind of API that covers some subset, that some of the similar things to get a sense for how are they solving some of these problems and what are the similarities and what are the differences? So we train that for a little while and then we can just grab a few examples, and here we have them. So here are horses here they are a zebras, and here they are back as horses.

Again, here's a hot zebra into a horse back of a zebra, it's kind of thrown away its head for some reason, but not so much a quick get it back again. This is a really interesting one like this is obviously not what zebras look like, but if it's going to be a zebra version of that horse, it's also interesting to see it's foliar situations. I guess it doesn't very often see. Basically, just an eyeball have no idea how to do that one. So some of them don't work very well. This one's done a pretty good job, this one's interesting. It's done a good job with that one in that one, but for some reason the one in the middle didn't get go yeah this one's a really weird shape, but it's done a reasonable job of it. This one looks good this one's pretty sloppy again for just a head up there, so you know I didn't it took me quite a bit for me, like 24 hours, to train it even that far so it's kind of slow - and I know Hellena - is constantly complaining On Twitter about how long these things take, I don't know how she's so productive with them so yeah. I will mention one more thing that just came out yesterday, which is there's now a multi-modal image to image translation of unpaired, and so you can basically now create different cats, for instance from this dock. So this is basically not just creating one example of the output that you want, but creating multiple ones. So here's a house cat, a big cat and here's a big cap house cat. This is the paper yeah.

So this came out like yesterday or the day before. I think I think it's pretty amazing captain dog, so you can kind of see how this technology is developing and I think you know if you, oh there's so many opportunities to you know, maybe do this with music or speech or writing or to create kind of Tools for artists or alright thanks everybody and see you next week, [Applause,]

Lesson 13: Image Enhancement

Outline

For the start of today's lesson we'll cover the CycleGAN, which is a breakthrough idea in GANs that allows us to generate images even where we don't have direct (paired) training data. We'll use it to turn horses into zebras, and visa versa; this may not be an application you need right now… but the basic idea is likely to be transferable to a wide range of very valuable applications. One of our students is already using it to create a new form of visual art.

But generative models (and many other techniques we've discussed) can cause harm just as easily as they can benefit society. So we spend some time today talking about data ethics. It's a topic that really deserves its own whole course; whilst we can't go into the detail we'd like in the time available, hopefully you'll get a taste of some of the key issues, and ideas for where to learn more.

We finish today's lesson by looking at style transfer, an interesting approach that allows us to change the style of images in whatever way we like. The approach requires us to optimize pixels, instead of weights, which is an interesting different way of looking at optimization.

Video Timelines and Transcript

1. 00:00:10

- **Fast.ai student accepted into Google Brain Residency program**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Welcome to lesson 13, where we're going to be talking about image, enhancement and image enhancement, we'll cover things like this painting that you might be familiar with. However, you might not have noticed before that this painting actually has a picture of an eagle in it. The reason you may not have noticed that before is this painting actually didn't used to have an eagle in it. By the same token, actually on that first page, this painting did not used to have Captain America's shield on it either, and this painting did not used to have a clock in it either. This is a cool new paper. Actually that just came out a couple of days ago called deep, painterly harmonization and it uses almost exactly the technique. We're going to learn in this lesson with some minor tweaks. But you can see the basic idea is: take one picture pasted on top of another picture and then use some kind of approach to combine the two and the basic approach is something called a style transfer before we talk about that, though, I wanted to mention this Really cool contribution by William Horton, who added this stochastic weight averaging technique to the first library that is now all merged and ready to go and he's written a whole post about that, which I strongly recommend you check out, not just because stochastic weight averaging actually lets.

You get higher performance from your existing neural networks, with basically no extra work, it's as simple as adding these two parameters to your fit function, but also he's described his process of building this and how he tested it and how it contributed to the library. So I think it's interesting. You know if you're interested in doing something like this. I think William had not built this kind of library before so he describes how he did it. Another very cool contribution to the faster, a library is a new train phase API and I'm going to do something. I've never done before, which are actually going to present somebody else's notebook, and the reason I haven't done it before is because I haven't liked any notebooks enough to think they're worth presenting, but so has done a fantastic job here of not just creating this new API. But also creating a beautiful notebook describing what it is and how it works and so forth, and the background here is, as as you guys know, we've been trying to train networks faster, partly as part of this dawn bench competition and also for a reason that you'll Learn about next week - and I mentioned on the forums last week - it would be really handy for our experiments if we had an easier way to try out different learning rate schedules and stuff, and I basically laid out an API that I had in mind as it'd. Be really cool if somebody could write this because I'm going to bed now and I kind of need it by tomorrow, and so when I replied on the forum. Well, that sounds like a good challenge and by 24 hours later it was done and it's been super cool I wan na.

I want to take you through it because it's it's going to allow you to do research into things that nobody's tried before. So it's called the Train phase API and the easiest way to show it is to show an example of what it does, which is here here is a iteration against learning rate chart as you're familiar with seeing and it. This is one where we train for a while at the learning rate of 0.01, and then we train for a while a learning rate of 0.001. I actually wanted to create something very much like that learning rate chart because most people that trained imagenet use this stepwise approach and it's actually not something that's built into fastai, because it's not generally something we recommend, but in order to replicate existing papers, I wanted To do it the same way, and so rather than writing a number of fit -- fit -- fit -- calls with different learning rates, so it'd be nice to be able to basically say train for n at this learning rate and then M epochs. At that learning rate, and so here's how you do that you can say phases. So a phase is a period of training with you know particular optimizer parameters, and it consists of a number of training phase objects. A training phase objects is how many epochs to train for what optimization function, to use and what learning rate, amongst other things that we'll see, and so here you'll see the two training phases that you just saw on that graph.

So now, rather than calling were not fit, you say low n dot fit with an optimizer scheduler with these phases get op and then from there most of the things you pass in can just get sent across to the fit function as per usual. So most of the usual parameters will work fine, but in this case, generally speaking, actually we can just use these training phases and you'll see it fits in the usual way and then, when you say plot LR there it is alright and not only does it plot The learning rate - it also plots momentum and for each phase it tells you what optimizer it used. You can turn off the printing of the optimizers. You can turn off the printing of momentum x' and you can do other little. Things like a training phase could have an LR decay parameter, so here's a fixed learning rate and then a linear decay, learning rate and then a fixed learning rate, which gives us that picture - and this is like might be quite a good way to Train. Actually, because we know at high learning rates you get to kind of explore better and they're, not low learning rates, you get to fine-tune better and it's probably better to gradually slide between the two. So you know this actually isn't a bad approach. I suspect you can use other decay types, not as linear, so cosine. This probably

2. 00:06:30

- **Cyclical Learning Rates for Training Neural Networks (another student's paper)**
- **& updates on Style Transfer, GAN, and Mean Shift Clustering research papers**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Makes even more sense as a genuinely potentially useful and learning rate, an ailing shape exponential, which is super popular approach. Polynomial which isn't terribly popular but actually in the literature, works better than just about anything else, but seems to have been largely ignored. So polynomials could to be aware of and what surveillance done is he's given us the formula for each of these curves and so with a polynomial. You get to pick what polynomial to use. So here it is with a different size and I believe, a P of 0.9 is the one that I've seen really good results for fYI. If you don't give a couple of learning rates, when there's an LR decay, then it will decay all the way down to zero right and, as you can see, you can happily start the next cycle at a different point. So the cool thing is now. We can replicate all of our existing schedules using nothing, but these training phases. So here's a function called phases. S JDR, which does SGD, are using the new training phase api, and so you can see if he runs this schedule and here's what it looks like. But he's even done the little trick. I have where you're training a little really low learning rate just for a little bit and then pop up and do a few cycles and the cycles are increasing in length and that's all done in a single in a single function. So the new one cycle we can now implement with again a single little function, alright, and so, if we run if we fit with that, we get this triangle all followed by a little flatter bit and the momentum is a cool thing.

The momentum has a momentum, decay right and then here we've got a fixed momentum at the end. So it's doing the momentum and the learning rate at the same time. So something that I haven't tried yet, but I think would be really interesting - is to use he's calling a differential learning rates. We've changed the name now to discriminative learning rates to use oops. Yes, what fix it? Just scream inactive learning race. So a combination of discriminative learning rates and one cycle - no one's tried yet so that would be really interesting. There's actually a the only paper I've come across, which has discriminative learning rates is called uses, something called Lars lar s, and it was used to Train imagenet with very, very large batch sizes by basically looking at the ratio between the gradient and the mean at H. Layer and using that to change the learning rate of each layer automatically and they found that they could use much larger batch sizes. That's the only other place. I've seen this kind of approach used, but there's lots of interesting things. You could try with combining discriminative learning rates and different interesting schedules, so you can now write your own LR finder of different types specifically because there's now this stop div parameter, which basically means that it'll, you know, use whatever schedule you asked for, but when the loss Gets too bad at all stop training, so here's one with no learning rate versus loss and you can see it stops itself automatically.

One useful thing that's been added. Is the linear parameter to the plot function? If you use linear schedule, rather than an exponential schedule and your learning rate finder, which is a good idea, if you've kind of fine-tuned into roughly the right area, then you can use linear to find exactly the right area. And then you probably want to plot it. With a linear scale, so that's why you can also pass linear to plot now as well. You can change the optimizer HBase and that's more important than you might imagine, because actually, the current state-of-the-art for

training on really large batch sizes really quickly for imagenet actually starts with rmsprop for the first bit, and then they switch to SGD for the second vid, and So that could be something interesting to experiment more with is like because at least one paper has now shown that that can work well and again, it's something that isn't well appreciated as yet, and then the bit I find most interesting is you can change your data And why would we want to change our data? Because you remember from lessons 1 and 2? You could use small images at the start and later bigger images later, and the theory is. The theory is that you could use that to kind of train the first bit more quickly with smaller images and remember, if you like, have half the height and half the width and you've got a quarter of the activations, basically every layer. So it can be a lot faster and it might even generalize better.

So you can now create a couple of different, for example, it's because he's got 28 and then 32 sized images. This is just so far 10, so there's only so much you can do and then, if you pass in an array of data in this data list parameter when you call fit pop shared it'll use a different data set for each face. So that's really cool because we can use that now like we could use that in our dorm bench entries and see what happens when we actually increase the size with very little code. So what happens when we do that? Well, the answer is here in Dorn bench: training on imagenet, and you can see here that Google is one this with half an hour on a cluster of TP use. The best non cluster of TPU result is fast: AI plus students under three hours, beating out Intel on 128 computers. Where else we ran on a single computer, we also beat Google running on a TPU, so using

3. 00:13:45

- **Tiramisu: combining Mean Shitft Clustering and Approximate Nearest Neighbors**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

This approach we've shown the fastest GPU result. The fastest single machine result the fastest, publicly available infrastructure result. These TPU pods, you can't use unless your Google and the cost is tiny like this Intel, one cost them $ 1,200 worth of compute. They haven't even written it here, but that's what you get a few user. Sorry, 128 computers in parallel, each one with 36 cause each one with 140 gig compared to our single AWS instance. So this is, you know, a kind of a breakthrough in in what we can do like the idea that we can train imagenet on a single, publicly available machine and this $ 72. By the way it was actually $ 25 because we used a spot instance. So one of our students, Andrew Shaw, built this whole system to allow us to throw a whole bunch of spod instance. Experiments up and run them simultaneously and pretty much automatically, but dawn binge doesn't quote the actual number we used. So it's actually 25 bucks, not 72 bucks. So this data list idea is super important and helpful, and so our sci-fi 10 results also now up there officially - and you might remember, the previous best was a bit over an hour and the trick here was using one cycle. Basically, so all this stuff, that's in silver, has training phase. Api is really all the stuff that we used to get these top results and really cool another fastai student, who goes by the name here. Bkj, has has taken that and done his own version.

He took a resonant 18 and added the concat polling that you might remember that we learnt about on top and used leslie cycles once leslie smith's one cycle and so he's got on the leaderboard. So all the top three first day i students, which is wonderful and same for cost, the top three and you can see paper space, so brett ran this on paper space and got the the

cheapest result. Just ahead of dkj been his name is, i believe, okay, so so i think you can see like a lot of the kind of interesting opportunities at the moment for the training stuff more quickly and cheaply, you're all about kind of the learning rate annealing and size. Annealing and like training with different parameters at different times, and I still think we buddies scratching the surface - I think we can go a lot faster and a lot cheaper and that's really helpful for people. You know in resource constrained environments, which is basically everybody except Google. Maybe Facebook architectures interesting as well, though, and one of the things we looked at last week was just like creating a simpler architecture which is basically state of the art. You know like the really basic kind of dark net architecture, but there's a piece of architecture. We we haven't talked about which is necessary to understand the inception network. The inception network is actually pretty interesting because they use some tricks to to actually make things more efficient and we're not currently using these tricks, and I kind of feel that maybe we should try. It, and so this is the the most interesting most successful inception network is their inception resident to network and most of the blocks in that looks something like this and it looks a lot like a standard ResNet block in that there's an identity connection here and then There's a conv confirmation is so a one by one.

Conf is simply saying for each grid cell. In your input, you've got a basically it's a vector, write a 1x1 by number of filters. Tensor is basically a vector right so for each grid cell. In your input, you're just doing a dot product with that tensor right and then, of course, it's going to be one of those vectors for each of the hundred and ninety two activations we're creating soon, basically do 192 dot products with grid cell 1, 1 and then 192 with good 0, 1, 2 or 1, 3 and so forth, and so you'll end up with something which has got the same grid size as the input and 192 channels in the output. So that's a really good way to you know: either reduce the dimensionality or increase the dimensionality of an input without changing the grid size. That's normally what we use 1x1 cons for. So here we've got a 1x1 conf and then we've got another one-by-one conf and then they add it together and then there's a third path, and this third path is not added this third path - it's not actually explicitly mentioned, but it's concatenated right and so actually there Is a form of ResNet which is basically identical to resonate, but we don't do plus we do concat right and that's called a dense net. Alright. So it's just a resonate where we do concat instead of plus and that's an interesting approach, because then the kind of the identity path is literally being copied right. So you kind of get that that that flow through all the way through and so as we'll see.

Next week, that tends to be good for, like segmentation and stuff, like that, where you really want to kind of keep the original pixels and the first layer of pixels and the second layer of pixels and touched so concatenate. Another than adding branches is, is a very useful thing to do, and so here we're concatenate in this branch and this this branch is doing something interesting, which is it's doing. First of all, the 1x1 con and then a 1 by 7 and then a seven by one. So what's going on there so what's going on, there is basically what we really want to do is do a seven by seven conch. The reason we want to do a seven by seven con is that if you've got multiple paths, each of which has different kernel sizes, then it's able to look at. You know different amounts of the image, and so like the original inception network had like a 1x1 or 3x3 a 5x5 seven by seven kind of getting concatenated in together. Something like that, and so, if we can have a seven by seven filter, then we get to kind of look at a lot of the image at once and create a really rich representation and so actually the stem of the inception network. That is the the first few layers of the inception network actually also used. You know this kind of seven by seven cond, because you start out with this 224 by 224 by three, and you want to turn it into something: that's like 112 by 112 by 64. So, by using a 7x7 Conniff, you can get a lot of information in each one of those outputs to get those 64 filters.

But the problem is that 7x7 conv is a lot of work. You've got 49 kernel, values, 2 x 49 inputs for every input pixel across every channel, so the compute is crazy.

4. 00:22:15

- **Facebook AI Similarity Search (FAISS)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You know you can kind of get away with it, maybe for the very first layer and in fact the very first layer, the very first con was reza net - is a seven by seven con. But i'm not so for inception for inception. They don't do a seven by seven comma. Instead, they do a one by seven, followed by a seven by one and so to explain the basic idea of the inception networks or all the different versions of it. That you have a number of separate paths which have different convolution widths in this case conceptually. The idea is, this is a one-by-one convolution with, and this is going to be a seven convolution with and so they're looking at different amounts of data, and then we combine them together, but we we don't want to have a seven by seven plunge throughout the network, Because it's just too computationally expensive, but if you think about it, if we've got some input coming in right and we have some big filter that we want and it's it's too big to deal with and what could we do right? So let's say, let's just to make it a little bit less drawing that's two five by five. What we can do is to create two filters, one which is 1 by 5 1, which is 5 by 1 or 7 or whatever on line. So we take our activations, the previous layer and we put it through the 1 by 5. We take the activations out of that and put it through the 5 by 1 and something comes out. The other end.

Now what comes out the other end? Well, rather than thinking of it as first of all, we take the activations, then we put it through the 5 by 1. Then we put it through the 5, but then we put it through the 1 by 5, 1 by 5, then the 5 by 1. What if, instead, we think of these 2 operations together and say what is a 5 by 1 dot production of one by five dot product do together right and effectively right? You could take a 1 by 5 and a 5 by 1, and the outer product of that is going to give you a 5 by 5 right now that you can't create any possible 5 by 5 matrix by taking that product right. But there's a lot of 5 by 5 matrices that you can create, and so the basic idea here is: you know when you think about the order of operations and I'm going to go into the detail of this if you're interested in more of the theory here, You should check out Rachel's numerical linear algebra course, which is basically a whole course about this stuff, but conceptually the idea is that very often the the computation you want to do is actually more simple than an entire 5x5 convolution, very often that the term we use In linear algebra is that there's some lower rank approximation, in other words, that the 1 by 5 and the 5 by 1, combined together that 5 by 5 matrix, is nearly as good as the 5 by 5 matrix.

You really, ideally would have computed if you were able to, and so this is very often the case in practice right just because the nature of kind of the real world is that the real world tends not to be. Is you know it tends to have more structure? You know than kind of randomness, so the cool thing is, if we replace our seven by if we replace our seven by seven conf, where the one by seven and a seven by one right, then this has basically for each cell. It's got 14 by input channel by output, Channel dot products to do, whereas this one has 49 to do okay. So it's just going to be a lot faster and we have to hope that it's going to be nearly as good. It's certainly capturing as much widths of information by definition. So if you're in student learning more about this specifically in a deep learning area, you can google for

factored convolutions. The idea was come up with three or four years ago. Now it's probably been around for long river. That was when I first saw it and yeah it turned out to work really well and the inception network uses it quite widely. They actually use it in their in their stem. It's it's interesting! Actually, we've talked before about how we tend to kind of add-on. We tend to say like this: it's main like backbone. You know like when we have ResNet 34, for example, we kind of say: oh, this is main backbone, which is all of the convolutions, and then we've talked about how we can add on to it a custom head right, and that tends to be like a mac.

Spalling layer and a fully connected layers and whether the you know it's actually kind of better to talk about the the backbone is containing kind of two pieces. One is the stem and then the other is kind of the main backbone. And the reason is that the thing that's coming in remember: it's only got three

5. 00:28:15

- **The BiLSTM Hegemony**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Channels, and so we want some sequence of operations, it's going to expand that out into something richer generally, something like 64 channels and so in ResNet. The stem is just super simple. It's a seven by seven cons straight to one followed by a Strad to Emax port yeah. I think that's it. If memory serves correctly an inception, they have a much more complex stem with multiple paths getting combined and cabin aided, including factoid comms, as one by seven and seven by one and now I'm kind of interested in what would happen if you stopped like a resident standard, Resonate on top of an inception instead, for instance like I think that would be a really interesting thing to try, because, like an inception, stem is kind of quite a carefully engineered thing, and this thing of like how do you take your three channel input and turn It into something richer seems really important, and all of that work seems to have got thrown away for ResNet. We like ResNet, it works really well, but what if we put, you know or a dent in it? What if we put the dense net backbone on top of an inception stem or what, if we replaced the seven by seven cons with a 1 by 7 and 7 by 1, factored conf? You know standard business, I don't know, there's lots of things we could try and I think it'd be really interesting. So there's some more thoughts about potential research directions. Ok, so that was kind of my little bunch of random stuff section.

Moving a little bit closer to the actual main topic of this, which is what I used image enhancement, I'm going to talk about a new paper briefly, because it's it really connects what I just discussed with what we're going to discuss next and the new paper. Well, it's not that new. Is it no it's a year old, it's a paper on progressive dance which came from Nvidia, and the progressive Ganz paper is really neat. It basically sorry Rachel. Yes, we have a question. One-By-One Kampf is usually called a network within a network. In the literature, what is the intuition of such a name know? Networking network is more than just a one by one time. It's part of it. I am, and we don't. I don't think there's any particular reason to look at that said, I'm aware of okay, so the the progressive can basically takes this idea of actually gradually increasing the image size. It's the only other direction, I'm aware of where people have actually gradually increase the image size, and it kind of surprises me, because this paper is actually very popular and very well known and very well liked. And yet people haven't taken

the basic idea of gradually increasing the image size and use it anywhere else, which shows you, the general level of creativity. You can expect to find in the deep learning research community.

Perhaps so they start with four by four, like they really go back start with a four by four again, like literally they're, trying to create like replicate four by four pixel and then eight by eight and so here's the 8 by 8 pixels. This is the celeb. A data set so we're trying to recreate pictures of celebrities, and then they go 65, 16 and then 32 and then 64 and then 128 and then 256 and one of the really nifty things they do is that as they increase size, they also add more layers To the network right, which kind of makes sense right, because if you're doing a more of a resin, Ettie type thing, you know then you're spitting out something which hopefully makes sense at each grid cell size. And so you should be able to kind of layer stuff. On top - and they do another nifty thing where they kind of add a skip connection when they do that and they gradually change the linear, interpolation parameter that moves it more and more away from the old 4x4 Network and towards the new 8x8 Network. And then, once this totally moved it across they throw away that extra connection. So it's it the details, don't matter too much, but it it uses the basic ideas. We've talked about gradually increasing the image size, it's kind of skip, connections and stuff. But it's a great paper to study because a you know: it's like one of these rare things where they've like good engineers, actually built something that just works in a really sensible way. Now it's not surprising.

This actually comes from Nvidia themselves right so in video, don't do a lot of papers and it's interesting that when they do they build something, that's so thoroughly practical and sensible, and so I think it's a great paper to study. You know if you want to kind of like put together lots of the different things we've learned. You know, and there aren't many re-implementation of this. So like it's an interesting thing, you know to project and you maybe you could build on and find something else. So here's what happens next, we eventually go up to 102 4 by 1 or 2, 4 and you'll see that the images are not only getting higher resolution but they're getting better, and so, when I prove 1 or 2 4 by 184. I'm going to see if you can guess which one of the next page is fake, they're, all fake. That's the next stage right you go up up up up up up up and then BOOM. Okay, so like dance and stuff they're getting crazy - and some of you may have seen this during the week yeah, so this video just came out and it's a speech by Barack Obama and let's check it out, so my Jordan Peele. This is a dangerous time. Moving forward, we need to be more vigilant with what we trust from you nourish. It's time we need to rely on trusted news sources. They sound basic, but how we before so as you can

6. 00:35:00

- **Implementing the BiLSTM, and Grammar as a Foreign Language (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

See they've used this kind of technology to literally move Obama's face in the way that Jordan peels face was moving and, like you, basically have all the techniques you need now to do that. So is that a good idea? So this is the bit where we talk about what's most important, which is like now that we can like do all this stuff. What should we be doing, and how do we think about that and the Tod our version is, I actually don't know recently. A lot of you saw the founders of this, the spacy prodigy folks founders of explosion, a I had to to talk and Matthew and Ennis. I went to dinner with them afterwards and we basically spent the entire

evening talking debating arguing about you know what does it mean? They're companies like ours, of building tools that are democratizing access to tools that can be used in harmful ways, and you know they're incredibly thoughtful people and we, I wouldn't say we didn't agree. We just couldn't. We just couldn't come to a conclusion ourselves. So I'm just going to lay out some of the questions and point to some of the research and when I say research, most of the actual literature review and putting this together was done by Rachel. So thanks Rachel. Let me start by saying the models we build are often pretty shitty in ways which are not immediately apparent, and you won't know how shitty they are unless the people that are building them with you, a range of people and the people that are using them with You or a range of people so, for example, a couple of wonderful research is Chemnitz at Stanford and rest joy.

Is she oh she's at Microsoft? Now she wasn't Stanford, okay, joy! Is it from PhD from MIT, so joy and Timna did this really interesting research where they looked at some basically off-the-shelf face, recognizes one from face plus plus, which is a huge Chinese company, IBM's and Microsoft's, and they looked for a range of different face types? I'm, generally speaking, you know the Microsoft one in particular was incredibly accurate and last the face type happened to be dark-skinned. When suddenly it went, you know 25 times worse, you know, got it wrong. Nearly half the time and for somebody to a big company like this, to release a product that, for like a very, very large percentage of the world, basically doesn't work, is more than a technical failure right. It's a really deep failure of understanding. What kind of team needs to be used to create such a technology and to test such a technology, or even an understanding of who your customers are yeah? Some of your customers have dark skin, yes, Rachel. I was also gon na add that the classifier is all did worse on women than on men, shocking, yeah yeah as funny like actually Rachel tweeted about something like this the other day, and some some guy was like. What's this all about, you know like what are you saying that we, like you know, don't you know about like people, people made cards for a long time.

You saying you need women to make cars and Rachel pointed out like well. Actually, yes, for most of the history of car safety, women in cars have been far far more at risk of death than men in cars, because the men created male looking feeling sized crash-test dummies, and so car safety was literally not tested on women, size bodies. So the fact you know, like you know, shitty product management with a total failure of diversity and understanding is not new to our field, and I would say that was comparing impacts of similar strength, men and women. Mm-Hmm yeah. I don't know why, like whenever you say something nice on Twitter like Rachel, has to say this because anytime, you say something like some Twitter there's, like 10 people, who'll be like all. You have to compare all these other things is, if, like we didn't know that so yeah I mean yeah other things. You know our very best. Most famous systems do like Microsoft's face recognizer or Google's language translator. You turn she is a doctor. He is a nurse into Turkish, and quite correctly, both the pronouns become. Oh, because there's no gendered pronouns in Turkish so go the other direction. I'll be a doctor. I don't know how to say that one for Christmas and what does it get turned into? He is a doctor. She is a nurse so, like we've got these kind of like biases, built into tools that we're all using every day and again people they go.

It's just showing us what's in the world and well: okay, there's lots of problems with that basic assertion, but, as you know, machine learning, algorithms, love to generalize right and so because they love to generalize. This is one of the cool things about you guys knowing the technical details now, because they love to generalize when you see something like 60 % of people cooking our in the pictures, they used to build this model, and then you actually run

the model on a Separate set of pictures, then 84 percent of the people they choose as cooking women rather than the correct 67 percent, but which is like a really understandable thing for an algorithm to do as it took a biased input and created a more biased output. Because you know for this particular loss function, you know that's kind of where it ended up, and this is a really common kind of a really common kind of model. Amplification, okay, so this stuff matters right. It matters in ways more than just you know, awkward translations or, like you know, black people's photos not being classified correctly or you know, maybe there's some there's some wins too, as well like you know, horrifying surveillance everywhere and maybe won't work on black people right. But yes or it'll be even worse because it's horrifying surveillance and it's flat-out racist and wrong.

Okay, that, sir, but but let's go deeper right like what hat like that, the four always say about human failings, humans such generally, you know that there's there's a long history of civilization and societies, creating kind of layers of human judgment which avoid hopefully the most horrible Things happening, and sometimes companies which love technology think let's throw away the humans and replace them with technology like Facebook did right. So, let's let two or three years ago, a couple years ago, Facebook literally got rid of their human editors like this is in the news at the time and they were replaced with algorithms and so now, as algorithms, that put all the stuff on your on your Newsfeed and human editors right at the loop. What happened next many things happen next, one of which was a massive horrifying genocide. Menma babies getting torn out of their mothers are right under fires, mass rape, murder and an entire people exiled from their homeland. Okay, I'm not gon na say that was because Facebook did this, but what I will say is that when the leaders of this horrifying project are interviewed, they regularly talk about how everything they learnt about the disgusting animal behaviors of Rangers that need to be thrown off. The earth they learnt from Facebook right because the algorithms just want to feed you more stuff.

That gets you clicking, and so, if you get told these people that don't look like you and you don't know a bad people and here's what the story's about the bad people and then you start clicking on them and then they feed you more of those things And next thing you know you have this like extraordinary cycle and people have been studying this right. So, for example, some we've been told a few times. People click on our fastai videos and then the next thing recommended to them is like conspiracy, theory, videos from Alex Jones, and then you know continues there because you know humans, click on things that shocked us and surprise us and horrify us right and so at So many levels you know this decision has had extraordinary consequences which we're only beginning to understand, and again this is not to say this particular consequence is because of this one thing, but to say it's entirely unrelated would be clearly ignoring all of the evidence and information That we have right so this is really kind of the key takeaway is to think like what are you building and how could it be used right? So lots and lots of effort now being put into face detection, including in our course right. We've been spending a lot of time, thinking about how to recognize stuff and where and there's lots of good reasons to want.

7. 00:45:30

- **Reminder on how RNN's work from Lesson #5 (Part 1)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

To be good at that, you know for improving crop yields and agriculture for improving diagnostic and treatment, planning and medicine for improving your logo, sorting, robot system, whatever right, but it's also being widely used in in surveillance and propaganda and disinformation. And you know again, it's like the question is like well. What do I do about that? I don't exactly know right, but it's just definitely at least important to be thinking about it. Talking about it, and sometimes you can do really good things. For example, meetup comm did something which I would put in the category of really good thing, which is they recognized early, a potential problem, which is that more men who are tending to go to their meet us and that was causing their collaborative filtering systems, which you're All familiar building now to recommend more technical content to men, and that was causing more men to go to more technical content, which was causing the recommendation systems to suggest more technical content to men right and this kind of runaway feedback. Loop is extremely common. When we interface the algorithm and the human together, so what it made up, do they intentionally made the decision to recommend more technical content to women right? Not because of some. You know highfalutin idea about how the world should be, but just because

8. 00:47:20

- **Why Attentional Models use “such” a simple architecture**
- **& “Tacotron: a Fully End-To-End Text-To-Speech Synthesis Model” (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

That makes sense right that the runaway feedback loop was a bug right. There are women that want to go to tech meetups, but when you turn up for a tech make up and it's all men and you don't go, and then it recommends more to men and so on and so forth right. So so I made up made us a really strong product management decision here, which was too not do what the algorithm said to do. Unfortunately, this is rare, most of these runaway feedback loops, for example, in predictive policing, where algorithms tell policemen where to go, which very often is more black neighborhoods, which end up crawling with more policemen, which leads to more arrests, which has assistance job more policemen to go. To more black, neighborhoods and so forth, so this problem of algorithmic bias is now very widespread and, as algorithms become more and more widely used for specific policy decisions, judicial decisions, day-to-day decisions about just who to give what offer to this just keeps becoming a bigger problem. Right and so, and some of them are really things that the people involved in the product management decision should have seen at the very start, didn't make sense and were unreasonable under any definition of the term, for example this stuff that I'd gone pointed out. These were questions that were used to decide, which was the sentencing guidelines. This software is used for both pretrial so who it was required to post bail.

So these are people that haven't even been convicted, as well as for sentencing and for who gets parole, and this was upheld by the Wisconsin Supreme Court last year. Despite all the flaws, okay, so whether you have to stay in jail because you can't pay the bail and how long your sentences for and how long you stay in jail for depends on what your father did. Whether your parents stayed married, who your friends are and where you live right now turns out these algorithms are actually terribly terribly bad, so some recent analysis showed that they're, basically worse than chance, but even the the company's building them were confident on these were statistically accurate. Correlations does anybody imagine there's a

world where it makes sense to decide like what happens to you, based on what your dad did. You know so a lot of this stuff. You

9. 00:50:15

- **Continuing on Spelling_bee_RNN notebook (Attention Model), from Lesson 12**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Know at the basic level is obviously unreasonable and a lot of it just fails in these ways that you can see empirically that these kind of runaway feedback loops must have happened, and these over generalizations must have happened. You know, for example, these are the kind of cross tabs that anybody working in these fields in any field is using. Algorithm should be preparing right, so prediction of likelihood of reoffending for black versus white defendants, like we can just calculate this very simply of the people that were labeled high-risk but didn't reopen. There were twenty three point: five percent white, but about twice that african-american. Where else, those that were labeled law risk but did riaf end, was like half the white people and only twenty percent of the african-american right so like this is the kind of stuff. We're least you know if you're taking the technologies we've been talking about and putting the production in some kind of in any way right or building a an API for other people or providing training for people or or whatever right. Then at least make sure that the that what you're doing can be tracked in a way that people know if some things you know people know what's going on - that's at least they're informed. Okay, I think it's a mistake, in my opinion, to assume that people are our evil. You know and and trying to break society right like I think I would. I prefer to start with an assumption of like okay.

If people are doing dumb stuff, it's because they don't all right, so you know at least make sure that they have this information, and I find very few ml practitioners thinking about what is the information they should be presenting in their interface. You know and then often I'll talk to data scientists, who will kind of say, like oh the stuff, I'm working one doesn't have a societal impact. It's like really like, like a number of people who think that what they're doing is entirely pointless come on. You know. Otherwise, you think people are paying you to do it for a reason. That's going to impact people in some way. Okay, so think about what that is. The other thing I know is a lot of people involved here are hiring people, and so, if you're hiring people, you know, I guess you're all very familiar with the first day. I philosophy now, which is the basic premise that - and I think it comes back to this idea - that I don't think about people on the whole were evil. I think they need to be informed and have tools right. So we're trying to have us give as many people the tools as possible that they need and particularly we're trying to put those tools in the hand of a more the hands of a more diverse range of people. So if you're involved in hiring decisions, perhaps you can keep this kind of philosophy in mind as well, that if you're, if you're, not just hiring a wider range of people, but also promoting a wider range of people and providing like really appropriate Career Management for a Wider range of people well, apart from anything else, your company will do better.

It actually turns out that more diverse teams are more creative and tend to solve problems more quickly and better than most diverse teams, but also you know you might avoid these kind of awful screw-ups, which you know that one level are bad for the world and another Level, if you ever get found out, they can also destroy your company. Also, they can destroy

you or at least make you look pretty bad in history. A couple of examples - one is, you know, going right back to the Second World War IBM basically provided all of the infrastructure necessary to track the Holocaust, so they had a these are the forms that they used and they say had different code. For you know, Jews were Asian gypsies for 12 death and the gas chambers were six and they all went on these punch cards. You can go and look at these punch cards and museums now, and this has actually been reviewed by a Swiss judge who said that IBM's technical assistance facilitated the task of the Nazis and the commission of the crimes against humanity. And you know it's interesting to read back the history. You know from these times to see like what was going through the minds of people at IBM at that time, and you know what what was clearly going through. The minds was like the opportunity to show technical superiority, the opportunity that I test out their you know their new systems, it's it's it's you know and the course the extraordinary amount of money that they were making um you know and when, when you do something which At some point down, the line turns out to be a problem, even if you were told to do it that can turn out to be a problem for you.

Personally, for example, you will remember the diesel emissions scandal in VW know. Who is the one guy that went to jail? It was the engineer right just doing his job okay. So if all of this stuff about actually you know not sucking up the world, isn't enough to convince you, they can up your life too right. So if you, if you do something that turns out to cause problems, even though somebody told you to do it, you can absolutely be held criminally responsible and you're. Certainly like look at the, but there's no Cogan. You know, I think a lot of people now know the name Aleksandr Cogan. He was the guy that handed over the Cambridge analytic er data he's a Cambridge academic, now, a very famous Cambridge academic, the world over for doing his part to destroy the foundations of democracy. Right so you know this is probably not how we want to go down in history all right. So let's have a break before we do read short question on a different topic. Yes, in one of your tweets, you said, drop out is patented. I think this is about wavenet patent from Google. What does it mean? Can you please share more insight on this subject? Does it mean that will help to pay to use drop out in the future? Yeah? Okay, good question: let's talk about that after the break and so let's come back at 7:40. The question before the break was about patents.

What does it mean? So so I guess the reasons coming up was because I wrote a tweet this week, which I think was like three words and said: drop out is patented. The patent holders is Geoffrey Hinton. So what isn't that great inventions all of our patents right? And so you know my answer is no. You know. Patents have gone wildly crazy. The amount of things that are patentable that we talk about every week would be dozens like it's so easy to come up with a little tweak, and then you know if you turn that into a patent, to stop everybody from using that little tweak for the next 14 years - and you end up with a situation we have now where everything is patented in 50 different ways. And so then you get these patent trolls, who have made a very, very good business out of looking better, basically buying lots of shitty little patents and then suing anybody who accidentally turned out. Did that thing you know like putting rounded corners on buttons. You know so who was it? Oh, this there's Apple

10. 00:58:40

- **Building the Attention Layer and the ‘attention_wrapper.py’ walk-through**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Suits Samsung or something I don't remember so yeah, so what does it mean for us that a lot of stuff is patented in deep learning? I don't know it's like one theory like a lot of the one of the main people doing. This is Google and people from Google who replied to this patent tend to assume that on Google's doing it because they wanted to have it defensively. So if somebody Sue's them they'll be like don't sue us we'll see you back, because we have all these patents. The problem is that, as far as I know, they haven't signed, what's called a defensive patent pledge. So, basically, you can sign a legally binding document that says our patent portfolio will only be used in defense and not offense, and even if you believe, all the management of Google would never turn into a patent. Troll you've got to remember that. You know management changes right and like to give a specific example. I know the you know the somewhat recent CFO of Google, you know has a much more. You know kind of aggressive stance towards the PNL and I don't know - maybe maybe she might decide that they should start monetizing their patents or maybe the you know the group that made that patent might get spun off and then sold to another company. That might end up in private equity hands and decide to monetize the patents or whatever like so, I think it's a problem.

There has been a big shift legally recently away from software patents actually having any legal standing. So it's possible that these will all end up thrown out of court, but you know the reality. Is that anything, but a big company is unlikely to have the financial ability to defend themselves against one of these huge patent trolls. So I think it's a problem. I don't know like it you you can't you can't avoid using patented stuff. If you write code like most, I wouldn't be surprised if most lines of code you write, have patents on them. So actually, funnily enough, the best thing to do is not to study the patents, because if you do and you infringe knowingly, then it's the penalties are worse. So the best thing to do is to like put your hands in your ears, sing, a song. You know and get back to work so that thing about it said about dropouts patented. Forget I said that you, don't you don't know that he's given that bit? Okay, this is super fun artistic style, we're gon na kind of go a bit retro here, because this is actually the kind of original artistic style paper and there's been a lot of updates to it. A lot of different approaches - and I actually think kind of in many ways the original is the best we're going to look at some of the newer approaches as well that I actually think the original is a terrific way to do it, even with everything that's gone Since, let's just jump to the code, so this is the style transfer, no four.

So the idea here is that we want to take a photo. We've got to take a photo of this bird and we want to create a painting that looks like van Gogh. Painted the picture of the bird thank off then go quite a bit of the stuff that I'm doing by the way uses an image net. You don't have to download the whole of image net for any of the things I'm doing, there's an image net sample on file start fast. Today I slash data which has like I don't know a couple of gig, and it should be plenty good enough for everything. We're doing if you want to get really great results, you can grab image net. You can download it from cab all the I'm Carol. It's the localization. Competition actually contains all of the classification data as well all right so, and you know if you've got room. It's good to have a copy of imagenet because it comes in handy all the time. So I just grab the bird out of my imagenet folder and there is my Buddha and so what I'm going to do is I'm going to start with this picture and I'm going to try and make it more and more. Like a picture of this bird painted by Van Gogh and the way I do, that is actually very simple, you're all familiar with it, we will create a loss function which we'll call f, yeah and the loss function is going to take as input a picture and Spit out as output a value and the value will be lower.

If the image looks more like a the bird photo painted by Van Gogh having written that loss

function, we will then use the pytorch, gradient and optimizers gradient times the learning rate and we're not going to update any weights. We're going to update the pixels of the input image to make it a little bit more like a picture which would be a bird painted by Van Gogh and we'll stick it through the loss function again to get more gradients, and do it again and again, and That's it so it's like identical to how we solve every problem, like you know, I'm a one-trick pony right. This is my only trick. Ok, tap, create a loss. Function, use it to get some gradients x, learning rates to update something always before we've updated weights. In a model, but today we're not going to do that, they're going to update that pixels in the input, but it's no different at all all right, we're just taking the gradient with respect to the input rather than respect to the weights. Okay, that's it so we're nearly done. Let's do a couple more things. Let's mention here that there's going to be two more inputs to our loss function. One is the picture of the bird birds. Look like this. Okay and the second is an artwork by Van Gogh. They look like this, oh and, of course, go okay, so, and by having those as inputs as well. That means we all be able to rerun the function later to make it look like you know, a bird painted by money or a jumbo jet painted by van Gogh or whatever right.

So those are going to be the three three inputs, and so initially as we discussed input here, this is going to be the first time I've ever found the rainbow pen useful, there's going to be that there's also: okay, some random noise. Okay. So we start with some random noise use the loss function, get the gradients, make it a little bit more like a bird painted by Van Gogh and so forth. Okay, so the only outstanding question which you know - I guess we can talk about briefly - is how we calculate how much our image looks like a bird, this bird painted by Van Gogh. Okay, so let's split it into two parts: let's put it into a part, called the content loss and that's going to return a function, a value, that's lower. If it looks more like the bird, not just any bird, the specific bird that we have coming in okay and then, let's also create something called the style loss and that's going to be a lower number. If the the image is more like van goths style. Okay, so there's one way to do the content loss, which is very simple. We could look at the pixels of the output, compare them to the pixels of the bird and to a mean square error addemup. So if we did that, I ran this for a while. Eventually, our image would turn into an image of the bird. You should try it right. You should try this as an exercise. Try to use the optimizer implied torch to start with a random image and turn it into another image by using mean squared error, pixel loss.

Okay, not terribly exciting, but that would be step one. The problem is, even if we already had our style loss, function, working beautifully and then presumably what we're going to do is we're going to add these two together right and then one of them will multiplied by some lambda so like adjust, some number we'll pick to Adjust how much style versus how much content right so assuming we had a style loss or we picked some sensible lambda. If we use two pixel wise content loss, then anything that makes it look more like Van Gogh and less like the exact photo. The exact background, the exact contrast lighting everything will decrease. The content was, which is not what we want right. We wanted to look like the bird, but not in the same way right. It's still gon na have the same two eyes in the same place and be the same kind of shape and so forth, but not the same representation. So what we're going to do is this is going to shop here we're going to use a neural network, all right, they're, going to use a neural network. I totally meant that to be black and a came out green, it's always a black box ever mind and we're going to use the vgg neural network, because that's what I used last year - and I didn't have time to see if other things worked, so you can Try that yourself during the week and the vgg network is something which takes in an input and sticks it through a number of layers and I'm just going to treat these as just the convolutional layers.

There's obviously value there. And if it's a bgg with batch norm which most are today, then it's it's also got better on men, there's some X pulling and so forth, but that's fine. What we could do is we could take one of these convolutional activations and then, rather than comparing the pixels of this bird, we could instead compare the vgg layer, five activations of this, to the vgg layer, five activations of our original birth or layer, six or layer. Seven or whatever, so why might that be more interesting? Well, for one thing: it wouldn't be the same bird right. It wouldn't be exactly the same because we're not checking the pixels we're checking some later set of activations, and so what are those latest sets of activations contained right? Well, assuming that's after some max pooling. They contain a small agree right. So it's less specific about where things are and rather than containing pixel color values they're more like semantic things like, is this kind of like an eyeball or is this kind of furry, or is this kind of bright or is this kind of reflective, or is this Laying flat whatever right so, we would hope that there's some level kind of semantic features through those layers where, if we get something a picture that that matches those activations, then any picture that matches those activations looks like the bird. But it's not the same representation of the bird.

So that's what we're going to do! That's what our content loss is going to be, and people generally call this a perceptual loss right which, because, like it's, really important in deep learning that you always create a new name for every obvious thing. You do right. So if you compare two activations together, you're doing a perceptual was okay. So so that's it! A Content. Loss is going to be a perceptual loss and then we'll do the style loss later. So, let's start by trying to create a bird that initially is random noise and we're going to use perceptual loss to create something that is bird-like. But it's not this better. Okay. So, let's start by saying they don't do 28 by 288, like we've, because pretty good to do one bird there's going to be no GPU memory problems right. So I was actually disappointed that I realized that I picked a rather small important image. It'd be fun to try this with something much bigger to create a really grand scale piece. The other thing to remember is: if you are like production izing, this you could like do a whole batch at a time. So people sometimes complain about this. This approach, gaddy's, is the lead author, the gaddy's style transfer, approaches being slow and I don't agree, it's low. It takes a few seconds and you can do a whole batch in a few seconds anyway.

So we're going to stick it through some trance, as per usual, transforms for vgg, 16 model, and so remember the transform class has a dunder call method, so we can treat it as if it's a function right. So if you pass an image into that, then we get the transformed image right so like try not to treat the fastai and ply torch infrastructure as a black box, because, like it's all designed to be like really easy to use in a decoupled way, all Right so this idea of that transforms are just callable, x' ie things that you can do with parentheses comes from pipe torch and we totally plagiarized the idea so with with torch vision or with fastai. You basically you're transforms are just color balls and the whole pipeline of transforms is just a callable. So now we have something of 3 by 2, 88 by 2 88, because pytorch likes the channel to be first and as you can see, it's been turned into a square for us, it's being normalized to 0 1 or that normal stuff. Ok, now we're creating a random image. Ok and here's something I discovered trying to turn this into a picture of anything. It's actually really hard. I found it very difficult to actually get an optimizer to get reasonable gradients that went anywhere and just as I thought,

11. 01:15:40

- **Impressive student's experiment with different mathematical technique on Style Transfer**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

I was going to run out of time for this class and really embarrassed myself. I realized the key issue. Is that pictures? Don't look like this? They have more smoothness, so I turned this into this by just kind of blurring it a little bit. I used a median filter that basically it's like a like a median pooling, effectively right and as soon as I change it from this to this, it immediately started training really well, ok, so it's like a number of little tweaks. You have to do to get these things to work is kind of insane, but there's that here is a little booty alright, so we start with a random image, which is at least somewhat smooth. Okay and I found that my bird image had a standard deviation of pixels, that was about half of this, so I mean about half of this mean, so I divided it by two just trying to make it a little bit easier for it to match. I don't know if it matters turn that into a variable, because this image remember we're going to be modifying those pixels with an optimization algorithm. So anything that involved in the loss function needs to be a variable, and specifically it requires a gradient because we're actually updating the image. Okay, all right, so we now have a mini batch of one three channels: 288 by 288, random noise, we're going to use for no particular reason the thirty-seventh layer of vgg. If you print out the vgg Network, you can just type in their member score, vgg and prints it out.

You'll see that this is a you know, kind of mid to late stage layer. So we can just grab the first 37 layers and turn it into a sequential model, and so now we've got a subset of heg that will spit out some mid layer activations, and so that's that's what the models going to be. So we can take our actual bird image right and we want to create a mini batch of one. So remember if you slice in numpy with none, also known as NP, you axis it

12. 01:18:00

- **Translate English into French, with Pytorch**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Introduces a new unit axis in that point, so in here I want to create an axis of size, one to say this is a mini batch of size, one alright, so slicing with none, just like I did here, has sliced with none to get this one's one Unit axis at the front, okay, sis, so then we turn that into a variable and this one doesn't need to write big, be updated. So it's we use DV to say you don't need gradients for this guy, and so that's going to give us our our target. Activations, okay, so we've basically taken our bird image turn it into a variable stuck it through our model to grab the thirty seventh layer. Activations and that's our target right is that we want our content loss to be this set of activations here. So then, we're going to create an optimizer, we'll go back to the details of this in a moment, but we're going to create an optimizer and we're going to step a bunch of times going 0. The gradients call some loss function, loss top backward. No, so that's the high-level version and I'm going to come back to the details in a moment, but the key thing is that the loss function, we're passing in that randomly generated image, the optimization image or actually the variable of it right. So we passed that to our loss function, and so it's going to update this using the loss function and the loss function is the mean squared error loss, comparing our current optimization image passed through our vgg to get the intermediate activations and comparing it to our target.

Activations, okay, just like we discussed okay and we'll run that a bunch of times and we'll print it out, and we have our bird but not the representation of the boat. Okay. So there it is so a couple of new details here. One is we had optimizer lb-ft, yes, anybody who's done. I don't know exactly what courses they're in, but certain parts of math and computer science courses comes into deep learning discovers we use all this stuff like Adam and the SGD, and always assume that nobody in the field knows the first thing about computer science and immediately Says, oh of any of you guys tried using the FDS, there's basically a long history of a totally different kind of algorithm for optimization that we don't use to train neural networks and of course the answer is actually the people who have spent decades studying neural networks. Do know a thing or two about computer science and it turns out these techniques on the whole. Don't work very well, but it's actually going to work well for this, and it's a good opportunity to talk about an interesting algorithm for those of you that haven't studied this type of optimization algorithm at school. So BFGS is one of the names Broyden favor. I can't remember anyway, initials are for different people at the L stands for limited memory, so it's really just quote: VFDs limited memory BFGS and it's an optimizer so as an optimizer. That means that there's some loss function and it's going to use some gradients.

To I mean not all optimizes use gradients, but all the ones we use do here's gradients to find a direction to go and try to make the loss function, go lower and lower by adjusting some parameters. Yeah this just an optimizer, but it's an interesting kind of optimizer, because it does a bit more work than the ones we're used to on each step and so specifically, okay, Facebook, okay, so the way it works is. It starts the same way that we used to, which is we just kind of pick somewhere to get started, and in this case we've picked like a random image, as we saw and as per usual, we we calculate the gradient, but we then don't just take a Step, but what we actually do is as well as finding the gradient. We also try to find the second derivative, so the direct second derivative says how fast is the gradient change, so the gradient is how fast of the function change. The second derivative is how fast as a gradient change. In other words, how curvy is it right and the basic idea is that if you know that it's like not very curvy, then you can probably jump further. But if it's very curvy, then you probably don't want to jump as far and so in in higher dimensions. The gradients called the Jacobian and the second derivative is called the Hessian you'll, see those words all the time that that's what they mean. Okay, again, mathematicians have to invent your words for everything as well they're, just like deep learning researchers.

So it may be a bit more snooty, so with BFGS we're going to try and calculate the second derivative and then we're going to use that to figure out kind of what direction to go and and how far to go all right. So it's less of a kind of a wild jump into the unknown. Now the problem is that actually calculating the hessian. The second derivative is almost certainly not a good idea, because in each possible direction that you can add for each direction that you're measuring the gradient in you also have to calculate the hessian in every direction. It gets ridiculously big so, rather than actually calculating it, we take a few steps and we basically look at how much the gradients changing as we do each step and we approximate the Hessian using that little function. Right and again, this seems like a really obvious thing to do, but nobody thought of it until someone. Well, surprisingly, a long time later, keeping track of every single step you take takes a lot of memory, so duh don't keep track of every step. You take just keep the last ten or twenty and the second bit there. That's the L to the l-bfgs, so a limited memory BFGS means keep the last ten or 20 gradients use that to approximate the amount of curvature and then use the curvature in gradient. To estimate what direction to travel and how far, and so that's normally not a good idea in deep learning for a number of reasons, you know it's obviously more work to do than a kind of an atom or an SGD update,

number Z, more memory memory is Much more of a big issue when you've got a GPU to store it on an hundreds of millions of weights. But, more importantly, the mini-batches super bumpy, so figuring out like curvature, decide exactly how far to travel is kind of polishing.

Turds, as we say, is that an American expression, or just an Australian Australian thing, a bit English there to do in a certain sense, yeah, obviously yeah, oh yeah, yeah polishing, turns you get the idea and also, interestingly, actually take using the second derivative information. It turns out is like a magnet for saddle points, so there's some interesting theoretical results. That basically say it's actually sends you towards nasty flat areas of the function. If you use second derivative information, so normally not a good idea, but in this case we're not optimising weights, we're optimizing pixels. So all the rules, change and actually turns our BFGS does make sense and because it does more work each time you know it's a kind of a different kind of optimizer. The API is a little bit different in plight. Watch as you can see here, when you say optimizer dot step, you actually pass in the loss function, okay and so my log. So my loss function is to call step with a particular loss function, which is my activation loss right and, as you can see you don't so you don't inside the loop, you don't say step step-step right, but rather it looks like this. So it's a little bit different and you're. Welcome to try and rewrite this to use. Sgd it'll still work it'll just take a bit longer.

I haven't tried it with SGD I'd, be interested to know how much longer it takes okay, so you can see the loss function going down the mean squared error between the you know, activations at layer 37 of our vgg model for our optimized image versus the target. Activations and remember the target activations were the vgg applied to our Albert, so make sense right so we've, okay, so we've now got a Content loss. Now one thing I'll say about this content loss. Is we don't know which layer it's going to work best, so it'd be nice if we were able to experiment a little bit more and the way it is here is annoying. Maybe we even want to use multiple layers? Okay, so, rather than like lopping off all of the layers are for the one we want. Wouldn't it be nice if we could somehow like grab the activations of a few layers as it calculates now. We already know one way to do that back when we did SSD. We actually wrote our own network, which had a number of outputs. Remember like the different convolutional layers. We spat out a different like icon thing, but I don't really want to go and like add that to the torch vision, ResNet model, especially not if, like later on, I want to try you know, then I want to try the torch vision, vgg model, and then I want to try an S and at a model I don't to go into all of them and like change their outputs right, besides, which I'd like to easily be able to turn certain activations on and off with demand.

So we briefly touched before this idea that pytorch has these fantastic things called hooks. You can have forward Hawks that, let you plug anything you like into the forward path of a calculation or a backward walk. So that's you plug anything. You like into the backward pass, so we're going to create the world's simplest forward hook, and this is one of these things that, like almost nobody knows about so like almost any code you find on the internet that implements style transfer will have all kinds of horrible Hacks, rather than using forward walks, but with four books, it's really easy so to create a forward hook, you just create a class right and the class has to have something called hook: function: okay and your hook function is going to receive the module that you've hooked. It's going to receive the input for the forward pass and it's going to receive the target, and then you do whatever the hell you like. So what I'm going to do is I'm just going to store the output of this module in some attribute? That's it all! Right so this can actually be called anything you like, but hook function seems to be the standard, because you can see what happens here in the constructor. Is I store inside some attribute? The result of this is going to be the layer that I'm gon na hook. You go

module register forward hook and pass in the function that you want to be called when this module, when it's when it's forward method is called. So when it's forward method is called, it will call self dot hook function which will store the output in an attribute cord features.

Okay, so now what

13. 01:31:20

- **Translate English into French: using Keras to prepare the data**
- **Note: Pytorch latest version now supports Broadcasting**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

We can do is we can create a vgg as before right and let's set it to not trainable. So we don't waste time and memory, calculating gradients for it and let's go through and find out, let's find all of the max pool layers. Alright. So, let's go through all of the children of this module and if it's a max pool layer, let's spit out index minus one. So that's going to give me the layer before the map, sport and so in general. The layer before and that's pool or the layer before us dry to cons is a very interesting layer. But because it's like it's the most, you know complete representation. We have at that grid cell size back because the very next layer is changing the grid. Okay, so that seems to me like a good place to grab the for content loss from, is you know the best most semantic, most interesting content we have at that grid size. So that's why I'm going to pick those indexes so Helia. Those are the indexes of the last layer before each max poor in vgg, so I'm going to grab this one here, 22, just no particular reason just to try something else. So I'm going to say sorry this one here 32, so I'm going to say, block ends. 3, that's maybe 32, so children, vgg indexed to block ends 3, will give me the 30 second layer of vgg as a as a module right and then, if I call the save features, constructor, it's going to go self cork equals 30. Second layer of the GG register forward hook hook, function, ok, so now, every time I do a forward pass on this bjg model.

It's going to store the 30 second layers. Output inside sf dot features. So we can now say see here. I'm calling my vgg Network, but I'm not strong it anywhere. I'm not saying you know: activations equals vgg of my image, I'm calling it throwing away the answer and then grabbing the features that we stored in our SF in our safe features, object right. So that way, this is now going to contain one. So like I've done. If this is a forward plus now that's how you do a forward pass and ply torch, you don't say dot forward, you just use it as a callable and using as a callable on an NN dot module automatically calls forward. That's how plat watch modules, what okay? So we call it as a koala ball that ends up calling our forward hook, that forward hook, stores the activations in SF dot features, and so now we have our target variable just like before, but in a much more flexible way. These are the same four lines of code. We had earlier I've just stuck them into a function, okay, and so it's just giving me my popped up, my random image to optimize and an optimizer to optimize that image. This is exactly the same code as before, so that gives me these, and so now I can go ahead and do exactly the same thing right. But now I'm going to use a different loss, function, activation lost number two which doesn't say out, equals mV GG again. The calls MV chuchita to a forward pass throws away the results and grabs. Sf top features.

Okay - and so that's now, my 30 second layer activations, which I can then do my MSA loss on you - might have noticed the last time the last loss function in this one, both multiplied by

a thousand. Why are they multiplied by a thousand again? This was like all the things that were trying to get this lesson to not work correctly. I didn't used to have the a thousand yeah. It wasn't training by lunchtime today. Nothing was working after days of trying to get this thing to work and finally kind of just randomly noticed like gosh. The last functions like the numbers are really low. You know like 10 e neg 7 and I just kind of thought: well what, if they weren't so low, so I multiplied them by a thousand and instead of working. So why did it not work because we're doing single precision, floating point right and single precision floating point ain't, that precise and particularly once you're kind of getting gradients that are kind of small and then you're multiplying around the learning rate can be kind of small, and You end up with a small number and if it's so small, they could get rounded to zero and that's what was happening and my model wasn't ready. Okay, so I'm sure there are better ways: I'm multiplying by a thousand whatever it works. Fine, like it doesn't matter what you multiply a loss function by, because all you care about is its is its direction. It's relative size right and, interestingly, like this is actually something similar we do for when we were training imagenet, we were using half precision floating point, because the Volta tents, of course require that and it's actually a standard practice.

If you want to get the half precision floating point to Train, you actually have to multiply the loss function by a scaling factor and we were using a thousand and twenty four or five twelve, and I think fast. Ai is now the first library that has all of the tricks necessary to train in half precision floating point built-in. So if you now, if you have a lucky enough to have a Volta or you can pay for a p3, if you've got a learner object, you can just say learned: half and it'll now just magically train correctly position floating point that built into the model. Data objects as well, it's all automatic and pretty sure, no other library does that okay. So this is just doing the same thing on a slightly earlier layer and you can see that the bird looks you know the later layer. You know doesn't look very bird-like at all, but you can kind of tell it's bird slightly earlier layer, more bird-like right and hopefully that makes sense to you that earlier layers are getting closer to the they're getting closer to the pixels. You know it's a it's! A smaller grid size well, there's this little more grid cells. Each cell is smaller, smaller receptive field, less complex semantic features, so the earlier we get the more it's going to look like a bit and in fact the paper has a nice picture of that, showing various different layers and kind of zooming into this house.

They're trying to make this house look like this picture, and you can see that later on, it's pretty messy and earlier on. It looks like this okay, so this is just doing what we just did and I will say, like one of the things I've noticed in our study group is anytime. I say to somebody to answer a question. Yes, if we're anytime, I say, read:

14. 01:38:50

- **Writing and running the 'Train & Test' code with Pytorch**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

The paper there's a thing in the paper that tells you the answer to that question. There's always this shock. Look: okay, read the paper me the paper, but seriously the papers have like they've done these experiments and drawn the pictures like there's all this stuff in the papers like it doesn't mean you have to read every part of the paper right, but at least look at The pictures right so check out the gaddy's paper. It's got nice pictures. Okay, so they've done the

experiment for us. They basically did this experiment, okay, but it looks like they didn't go as deep. They just got some earlier ones. Okay, the next thing we need to do is to create style loss all right, so we've already got the loss, which is how much like the bird is it now we need how like this painting style? Is it and we're going to do nearly the same thing? Okay, we're going to grab the activations of somewhere now. The problem is that the activations of some layer - let's say it - was a five by five layer way. Of course, there are no five by five layers up to 24 by 24 bit corporate head five by five. I, whatever nine to say, totally unrealistic sizes but never mind so there, here's some activations and we could get these activations both per hour, the image we're optimizing and for our van Gogh. Thank you and let's look at our van Gogh painting there. It is night I downloaded this from Wikipedia and I was wondering what is taking so long to load.

It turns out that the the Wikipedia version I downloaded was thirty thousand by thirty thousand pixels. Okay, that's pretty cool they've got this like serious gallery quality archive stuff. There I didn't know it existed, so don't try and run a neuron in on that totally killed my Jupiter notebook, okay, so yeah. So we can do that for our van Gogh image and we can do that for our optimizer image. And then we can compare the two and we would end up creating an image that looks. You know, content like the painting, but it's not the painting, that's not what we want. We want something with the same style, but it's not the painting, it doesn't have the contrary. So we actually want to throw away all of the spatial information right, we're not trying to create something that looks that has a moon here and stars here and because it's a church here and whatever right, we don't want any of that. So how do we throw away all the spatial information? What we do is let's grab so there are like, in this case they're, like nineteen faces on this right, like nineteen slices. So let's grab this top slice. Okay, let's grab that top slice. So that's going to be a five by five matrix: okay and now, let's flatten it so now, we've got a twenty five long vector now, in one stroke, we've thrown away the you know the bulk of the spatial information by flattening it all right.

Now, let's grab a second slice, alright, so another another Channel and do the same thing: okay, so here's channel one flattened here's channel two flattened and they've both got $ 25 and now, let's take the dot product which we can do with at in dump. I and so the nine the dot products going to give us one number. What's that number? What is it telling us? Well, assuming this is kind of somewhere around the middle activation. You know the activations are somewhere around the middle layer of the vgg network. We might expect some of these activations to be like how textured is the brushstroke and some of them to be like how bright is this area? And some of them to be like? Is this part of a house or a part of a circular thing or other parts to be? You know how dark is this part of the painting and so this a dot product? Remember it's basically a correlation right. If, if, if this element - and this

15. 01:44:00

- **NLP Programming Tutorial, by Graham Neubig (NAIST)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Element are both highly positive or both highly negative. It gives us a big result right we're also, if they're the opposite gives a small result. If they're both close to zero, it gives no result, so it's basically a dot product as a measure of how similar these two things are right, and so, if the activations of channel 1 and channel 2, you know how similar that it basically says. Let's

give an example: let's say this: first, one was like how textured are the brushstrokes, and this one here that say was like how kind of diagonally oriented are the brushstrokes right and if and if both of these were high together and both of these were high Together then, it's basically saying, oh anywhere, that there's more textured brush strokes. They tend to be diagonal right or another. Interesting one is what would be the dot product of c1 with c1, so that would be basically the the two normal that the sum of the squares of that channel, which, in other words, is basically just on average. How so, let's go back. I screwed this up. Channel 1 might be texture and channel, two might be diagonal, and this one here would be cell 1, comma 1, and this cell here would be like cell, say, 4, comma, 2, and so sorry, what I was should have been saying is: if these are both high At the same time - and these are both high at the same time, then it's saying grid cells would have texture tend to also have diagonal. So sorry I drew that all wrong. The idea was right.

It has drew it all wrong so yeah, so this number is going to be high when grid cells that have texture also have diagonal and when they don't they don't so that's C: 1 dot product C 2, where else C 1 dot product C 1 right is Basically, as we said like the two norm, effectively squared or the sum of the squares of C 1 sum over, I of c1 squared - and this is basically saying how, in how many grid cells is the textured channel active and how active is it so? In other words, see one dot product see, one tells us how much textured painting is going on and see two dot product see two tells us how much diagonal paint strokes is going on, and you know maybe see three is you know? Is it bright, colors so see three dot product see three would be you know. How often do we have bright colored cells? So what we could do, then is we could create a 25 by 25 matrix containing every one channel 1 channel 2 channel 3 channel 1 channel 2 channel 3, so you're not channel man. It's been a long day. 19. There are 19 channels, 19 by 19, okay, channel 1 channel 2 channel 3 channel 19 channel 1 channel 2 channel 3 did it do it channel 19, okay, and so this would be the dot product of channel 1 or channel 1. This would be the drug product of channel 2 with channel 2 and so forth. After flattening yeah and like we discussed mathematicians, have to give everything a name, so this particular matrix where you flatten something L out and then do the doctrine.

All the dot products is called a gram matrix

16. 01:48:25

- **Question: “Could we translate Chinese to English with that technique ?”**
- **& new technique: Neural Machine Translation of Rare Words with Subword Units (Research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

And I'll tell you a secret like most deep learning: practitioners either don't know or don't remember all these things like what is a grand matrix if they ever did study at university, they probably forgot it because they had a big night afterwards and the way it works In practice is, like you realize, oh, I could create a kind of non spatial representation of how the channels correlate with each other and then, when I write up the paper, I have to go and ask around and say like does this thing have a name and Somebody be right, isn't it the gram matrix and you go and look it up that it is right, so don't think like you have to go and study all of math. First, you use your intuition and common sense, and then you worry about what the math is called later. Normally sometimes it works the other way not with me, because I can do a mess

okay, so this is called the gram matrix and, of course, if you're, a real mathematician is very important that you, you say this as if you already always knew it was a Gram matrix and you kind of just go: oh yes, we just calculate the gram matrix, that's really important, so the grand matrix then is this kind of map of the diagonal is perhaps the most interesting right. The diagonal is like, which channels are the most active and then the off diagonal is like which channels tend to appear together and overall.

If two pictures have the same style right, then we're expecting that some layer of activations they will have similar Gramm matrices, because if we found the level of activations that capture a lot of stuff about like paint, strokes and colors and stuff, then the diagonal alone might Even be enough and like that's another interesting homework assignment, if somebody wants to take, it, is try doing gaddy's style transfer, not using the ground matrix, but just using the diagonal of the gray matrix, and that would be like a single line of code to change that. I haven't seen it tried and I don't know if it would work at all, but it might work fine, Christine Christine. Okay, yes, Christine you've tried it and it works most of the time except when you have funny pictures where you need two styles to appear in the same spot. So it sounds like grass in one half and like a crowd in one half, and you need the two styles. Ah cool you're still gon na take your homework. Is that okay, Christine says she'll, do it for you? Okay? So let's do that. So here's our painting I've tried to resize the painting, so it's the same size as my bird picture. So that's all this is just doing so. I make the yeah so there it is. It doesn't matter too much which bit I use as long as it's got. Lots of a nice style in it, I grab my optimizer and my random image just like before and this time I call save features for all of my block ends and that's going to give me an array of save features, objects, one for each module.

That appears the layer before IMAX Paul so now, because because this time I want to play around with different activation layer styles or more specifically, I want to let you play around with it. Okay, so now I've got a whole array of them. So now I call my vgg module on my my image again: yeah, I'm not going to use that yeah. Okay, ignore that line.i style image; sorry style images, my van Gogh painting, so I take my style image. Put it through my transformations to create my transform style image. I turn that into a variable put it through the forward pass of my vgg module, and now I can go through all of my save features, objects and grab each set of features and notice. I call clone right because I don't I've caused like later on. If I call my vgg object again, it's going to replace those contents, I haven't quite thought about whether this is necessary. If you take it away and it's not that's fine but us as being careful. So here's now an array of the activations at every block and player. So here you can see all of those shapes and you can see like being able to like whip up a list. Comprehension really quickly. It's really important in your Jupiter fiddling around because you really want to be able to like immediately see you know. Here's my channel sixty four one, four, six, five, twelve and you can see here the grid size having as we would expect because all of these appear just before M export. So so do a Graham MSC loss.

It's going to be the MSE loss on the ground. Matrix of the input versus the gram matrix of the target and the ground matrix is just the matrix multiply of X with X, transpose where X is simply equal to my input where I've flattened the batch and channel axes all down together, and I already got one Image so you can kind of ignore the batch part right, it's basically channel and then everything else which in this case is the height and width, is the other dimension, because there's now it'll be channel by width and then, as we discussed, we can then just do The matrix multiply of that by its transpose and just to normalize it we'll divide that

17. 01:54:45

- **Leaving Translation aside and moving to Image Segmentation,**
- **with the "The 100 layers Tiramisu: Fully Convolutional DenseNets" (research)**
- **and "Densely Connected Convolutional Networks" (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

By the number of elements, it would actually be more elegant if I had said / import, dot, Nam, elements that were in the same thing: okay and then again, this kind of keep me tiny numbers. So I multiply it by a big number to make it something more sensible. Okay, so that's basically my loss right. So now my style loss is to take my image to optimize, throw it through vgg forward, pass grab an array of the features in all of the safe features objects and then call my Graham MSC loss on every one of those layers. Okay and that's going to give me an array, and then I just add them up now you could add them up with different weightings. You could add up subsets whatever right in this case, I'm just grabbing all of them pass that into my optimizer as before, and here we have a random image in the style of Van Gogh, which i think is kind of cool and again gaddy's has done it For us here is different layers of random image in the style of Van Gogh, and so that the first one as you can see, the activations are simple, geometric things not very interesting at all. The later eight layers are much more interesting, so we kind of have a suspicion that we probably want to use later layers largely for our style laughs. If we wanted to look good all right, I added this. What was it this save features, dot close, which just calls remember, I stored the hook here and so Hawk. Don't remove, gets rid of it and it's a good idea to get rid of it, because otherwise, you know you can potentially just keep using memory all right.

So, at the end I go through each of my so features. Objection was it so style transfer is adding the two together with some weight, so there's not much to show grab my optimizer grab my image and now my combined loss is the MSE loss at one particular layer. My style loss set all of my layers sum up. The stay losses add them to the content loss, the content lost. So I'm scaling actually the style loss a scaled already by 1e6, and this one is one two, three, four, five, six, okay, so actually they're they're, both the scaled. Exactly the same, add them together and again you could try weighting the different style losses or you could maybe remove some of them whatever. So this is the simplest possible version, train that and like holy, it actually looks good. So I think that's yeah. I think that's pretty awesome, you know again, you know the main takeaway here is: if you want to solve something with a neural network, all you got to do is set up a loss, function and then optimize something all right and the loss function is something which A lower number is something that you're happier with because then, when you optimize it, it's going to make that number as low as you can and it'll do. What's your, what did its do? Okay? So here we can't what we didn't come up with. Gettys came up with a loss function that does a good job of being a smaller number.

When it looks like the thing we want it to look like, and it looks like the style of the thing we want to be in the style of that's all. We had to do right, we like what it actually comes to it. You know, apart from implementing gram, MSE loss, which was like six lines of code. If that that's our loss function, pass it to our optimizer, wait about five seconds and we're done and remember. We could do a batch of these at a time, so we could wait five seconds and sixty-four of these will be done yeah. So I think that's really interesting and and since this paper came out, you know it's really inspired a lot of interesting work to me, though, most of the

interesting work hasn't happened yet because to me the interesting work is the work where you combine human creativity with These kinds of tools - you know - and you know I haven't seen much in the way of tools that you can download or use where you know that the artist is in control and can kind of do things interactively it's interesting talking to the guys at the Google Magento project, which is kind of their creative value project, all of the stuff they're doing with music, is specifically about this. It's building tools that musicians can use to perform in real time, and so you'll see much more of that on the music space thanks to magenta.

If you go to their website, there's all kinds of things where you can like press the buttons to like actually change the drum beats or melodies or keys or whatever, and you can definitely see like Adobe an Nvidia, it's kind of starting to release. You know little prototypes and started you through this spit. You know this kind of like creative AI explosion hasn't happened yet I think we have pretty much all the technology we need, but no one's like put it together into a thing and said. Like look at the thing I built and look at the stuff that people built with my thing, you know so that's just a huge area of opportunity. So the paper that I mentioned at the start of class in passing the one where we can add Captain America's shield to arbitrary paintings, basically used this technique right that the trick was, though some minor tweaks to make the kind of the pasted Captain America shield blending Nicely right but like that, would that papers only a couple of days old so like that would be a really interesting project to try, because you you can use all this code. You know it really does leverage this approach and then you could start by you know making the content image be like the painting with the shield and then the style image could be the painting without the shield and like that would be a good start. And then you could kind of see what specific problems they try to solve and there's painting in this paper to make it better.

But you know you could you could have a start on it right now? Okay, so let's make a quick start on the next bit, which is yes, Rachel, say two questions earlier. There are a number of people that expressed interest in your thoughts on pyro and probabilistic programming yeah. So you know tensor flows now, but this particular tends to flow probability or something there's yeah, there's a bunch of probabilistic programming frameworks out there. I I think they're intriguing, you know, but as yet unproven in the sense that, like I haven't, seen anything done with any probabilistic programming system, which hasn't been done better without them. The basic premise is that it allows you to create more of a moral of how you think the world works and then like plug in the parameters. So back when I used to work in management consulting 20 years ago, we used to do a lot of stuff where we would use a spreadsheet and then we would have these Monte Carlo simulation and plugins at risk, and one crystal ball. I don't know if there still exists decades later, but basically they would let you like change a spreadsheet cell. To say this is not a specific value, but it actually represents a distribution of values. With this mean and the standard deviation or it's got this distribution, and then you would like hit a button and the spreadsheet would recalculate a thousand times pulling random numbers from those distributions and show you like the distribution of your outcome.

That might be some. You know profit or market share or whatever, and we used them all the time back. There apparently think that feel that a spreadsheets a more obvious place to do that kind of work, because you can kind of see it all much more. Naturally, but I don't know we'll see at this stage - I I hope it turns out to be useful, because I find it very appealing and it kind of appeals to, as I say, the kind of work I used to do. A lot of there's actually whole practices around this stuff. They used to call systems dynamics which really was built on top of this

kind of stuff, but no it's it's not quite gone anywhere. Hey then there is a question about pre training for generic style transfer. Yes, I don't think you can pre train for a generic style, but you can pre train for a generic photo for a particular style which is where we're going to get to. Although it may end up being a homework, I haven't decided, but I'm going to do all the pieces and one more question is: please ask him to talk about multi-gpu, oh yeah, I haven't had a slide about that. It's events actually we're about to about to get it. So, yes, okay! So before we do just another interesting picture from the Geddes paper, they've got a few more just didn't fit in mice in my slide here, but different convolutional layers for the style, different style to content ratios and here's the different images.

Obviously this isn't then go anymore. There's a different combination, so you can see like if you just do like wall style. You don't see any image. If you do all you know lots of content that you use low enough convolutional layer, it looks okay, but the backgrounds kind of dumb. So you kind of want somewhere around here - or here I guess anyway, so you can play around with it and experiment, but also use the paper to help guide you. Actually. I think I might work on the math now and we'll talk about multi, GPU and and super resolution next week, because I think that this is from the the paper and like one of the things I really do want you to do after we talk about a Paper is to read the paper and then ask questions on the forum. Anything that's not clear, but there's kind of like a key part of this paper, which I wanted to talk about and discuss how to interpret it. So we're going to be the paper says: we're going to be given an input image X and this little thing means it's whatnot. It means it's a vector Rachel, but this one's a matrix. I guess it could mean either yeah. I don't know - maybe it's anyway. So normally small letter, bulk means vector or a small letter with Dubey on top means vector they can both mean vector and normally big letter means matrix or small letter with two doobies on top means matrix. In this case, our image is a matrix. We are going to basically treat it as a vector, so maybe we're just getting ahead of ourselves.

So we've got an input image X and it can be encoded in a particular layer of the CNN by the filter responses. So the activations, your responses are activations right. So hopefully, that's something you all understand, that's basically what a CNN does is it produces layers of activations Alea has a bunch of filters right, which produce a number of channels right and so this year says that layer number L has capital our filters and again this Capital does not mean matrix, so I don't know math notation is so inconsistent, so capital NL distinct filters that layer, L, which means it has that also that many feature Mouse right so make sure you can see that this letter is the same as list letter. That's you've got to be very careful to read the letters and recognize it's like snap. You know that's the same letter as that. Okay, so obviously NL feature maps are in our filters, filters create and our feature maps or channels h1 is of size M. Okay. So I can see this is where the this is, where the unrolling is happening. Each map is of size M little L right. So this is, like you know, m square bracket, L in numpy, notation, it's the elf layer, so m for the elf layer and the the size is height times: width; okay, so we flattened it out. So the responses at that layer, L, can be stored in a matrix, F and now the old grows at the top for some reason. So this it's not f, ^ else.

It's just another indexing we're just moving around fun, and this thing here where we say it's an element of R. This is a special I mean in the real numbers n times M. This is saying that the dimensions of this is n by M right. So this is really important, like you, don't move on it's just like with pytorch, making sure that you understand the rank and size of your dimensions. First same with math right, you did. These are the bits where you stop and think. Why is it n by M right? Okay, so n is a number of filters. M is height by width right. So do you remember that thing where we did viewer batch x, channel, comma, minus 1 right here that is okay, so

try to map the code to the math, so f is f is X. If I was nicer to you, I would have used the same letters as the paper, but I was too busy getting this damn thing working to do that carefully, so you can go back and rename it as capital F, okay, and this is why we moved the L to the top is because we're now going to have some more indexing right so like, where else your numpy or apply torch. We index things by square brackets and then lots of things with commas between the approach in math is to like surround your letter by little letters all around it: okay and just throw them up there everywhere. So here F, L is the elf layer of F and then I J is the activation of the I filter at position, J of layer, yeah right, so position J is cut up to size M, which is up to size height by width. This is the kind of thing that be easy to get confused.

Like often you'd see an IJ and assume that's like indexing into a position of an image like height by width, but it's totally not. Is it okay, it's indexing into channel by flattened image right, and it even tells you it's the I filter the earth channel in the J's position in the flattened out image in layer. L right so you you're, not gon na, be able to get any further in the paper unless you know, unless you understand what F is okay, so that's. Why, like these are the bits where you stop and make sure you're comfortable all right. So now the content lost, so I'm not going to spend much time on, but basically we're going to just check out the values of the activations versus the predictions squared right. So there's our content, loss, okay and the style loss will be much the same thing. But using the grande matrix G, okay - and I really wanted to show you this ones, I think it's super. This is sometimes I really like things you can do in math note and they're things that you can also generally do in J and APL, which is this kind of this implicit loop going on here right. What this is saying is there's a whole bunch of values of I and a whole bunch of values of J and I'm going to define G for all of them and there's a whole bunch of values of L as well. I'm going to define G for all of those as well right and so for all of my G at every L of every I at every J. It's going to be equal to something, and you can see that something has an i and a J and an L right so matching these, and it also has a K and that's part of the sum so what's going on here.

Well, it's saying that my grand ma[illegible]tre -- ks in layer, l for the eighth channel well there's out channels anymore in the eighth position in one axis and the j DH position and another axis is equal to my F matrix, so my flattened out matrix for the Eighth channel in that layer versus the jafe channel and the same layer, and then I'm going to sum over I'm going to say see. This K in this case are the same letter right. So we're going to take the k, DH position and multiply them together and then add them all up all right. So that's exactly what we just did before when we calculated our own matrix right. So like this, there's a lot going on because of some like to me very neat. Notation right, which is there are three implicit loops or going on at the same time, plus one explicit loop in the Sun. And then they all work together to create this grand matrix for every layer. That's so let's go back and see if you can match this yeah. So so, oh that's kind of happening all at once, which i think is pretty great. Okay, so that's it so next week we're going to be looking at a very similar approach, basically doing style transfer all over again, but in a way where we were actually going to train a neural network to do it for us rather than having to do the Optimization and we'll also see that you can do the same thing to do super resolution and we're also going to go back and revisit some of that SSD stuff, as well as doing some some segmentation.

So if you're, if you've forgotten SSD, might be worth doing a little bit of revision this week, all right thanks, everybody see you next week, [Applause,],

Lesson 14: Super resolution; Image segmentation with Unet

Outline

In this final lesson, we do a deep dive into super resolution, an amazing technique that allows us to restore high resolution detail in our images, based on a convolutional neural network. In the process, we'll look at a few modern techniques for faster and more reliable training of generative convnets.

We close with a look at image segmentation, in particular using the Unet architecture, a state of the art technique that has won many Kaggle competitions and is widely used in industry. Image segmentation models allow us to precisely classify every part of an image, right down to pixel level.

We hope you enjoyed your deep learning journey with us! Now that you've finished, be sure to drop by the forums to tell us how you're using deep learning in your life or work, or what projects you're considering working on now.

Video Timelines and Transcript

1. 00:01:25

- **Time-Series and Structured Data**
- **& "Patient Mortality Risk Predictions in Pediatric Intensive Care, using RNN's" (research)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Welcome to the last lesson, lesson: 14: we're going to be looking at image segmentation today, amongst other things, but before we do a bit of show-and-tell from last week, Elena Ali did something really interesting, which was she tried finding out what would happen if you did cycle Gain on just three or 400 images - and I really like these projects where people just go to Google Image Search. You know using the the API you're one of the libraries out there. Some of our students have created some very good libraries for interacting with Google images. Api download a bunch of stuff that they're interested in, in this case some photos and some stained glass windows and yeah with three or four hundred photos of that she trained a model. She trained actually a few different models. This is what I particularly liked and, as you can see, with quite a small number of images, you know she gets some very nice stained-glass effects. So I thought that was a interesting example of yeah, using pretty small amounts of data that was readily available, that she was able to download pretty quickly and there's more information about that on the forum if you're interested yeah. It's it's interesting to wonder about what kinds of things people will come up with with this kind of generative

model, it's clearly a great artistic medium, it's clearly a great medium for forgeries and bakeries. I wonder what other kinds of things before will realize they can do with these kind of generative models.

I think audio is going to be the next big area and also very like interactive type stuff that Nvidia, I just released a paper showing a interactive kind of photo repair tour where you just like brush over an object, and it replaces it with. You know a deep learning generated replacement very nicely. Those kinds of interactive tools, I think, would be very interesting too. So before we talk about segmentation, we've got some stuff to finish up from last time, which is that we look at doing style transfer they actually directly. Optimizing pixels - and you know like with most of the things in part two: it's not so much that I'm wanting you to understand style transfer per se, but the kind of idea of optimizing your input directly and using activations as part of a loss function is really The key kind of takeaway here so it's interesting, then, to kind of see the photos effectively the follow-up paper you know not from the same people, but the paper that kind of came next in the in the sequence of these kind of vision, generative models with this One from Justin, Johnson and folks at Stanford, and it it actually does the same thing style transfer, but it does it in a different way. Rather than optimizing, the pixels we're going to go back to something much more familiar and optimize some weights, and so specifically we're going to train a model which learns to take a photo and translate it into a photo on this in the style of a particular artwork.

So each ComNet will learn to produce one kind of style. Now it turns out that getting to that point there's an intermediate point, which is, I actually think, kind of more more useful and Texas halfway there, which is something called super resolution, so we're actually going to start with super resolution because then we'll build on top of Super resolution to finish off the style, transfer, ComNet, based I'll transfer and so super resolution is where we take a low res image, we're going to take 72 by 72 and upscale up to a larger image 288 by 288. In our case, trying to create, you know a higher res image that looks as real as possible, and so this is a pretty challenging thing to do, because at 72 by 72, there's not that much information about a lot of the details and the cool thing is That we're going to do it in a way as we tend to do with vision models which is not tied to the input size. So you could totally then take this model that and apply it to a 288 by 288 image and get something that's four times bigger on each side so 16 times bigger than that, and and often it even kind of works better at that level, because you're really Introducing a lot of a lot of detail into the finer details and you could really print out a high resolution, print of something which earlier on was pretty big so later.

So this is the notebook or enhance, and it is a lot like that kind of CSI style enhancement where we're going to take something that appears like the information is just not there and we kind of invent it, but the confidence going to learn to invent it. In a way, that's consistent with the information that is there, so, hopefully you know it's kind of inventing the right information. One of the really nice things about this kind of problem is that we can create our own data set as big as we like without any labeling requirements, because we can easily create a low res image from a high-res image just by down sampling our images. So something I would love some of you to try doing the week would be to through other types of imaged image, translation where you can invent kind of late labels invent your dependent variable, for example, D skewing. You know so either recognize things that have been rotated by 90 degrees or better still, that have been rotated by five degrees and straighten them. Colorization, so turn make a bunch of images into black-and-white and learn to put the color back again. Noise reduction, you know, maybe do a really low quality, JPEG, save and learn to put it back to how it should have been, and so forth or yeah, maybe taking something.

That's like in a 16 color palette and put it back to a higher color palette. I think these things are all interesting because they can like be used to take.

You know pictures that you may have taken back on crappy old digital cameras before there are high resolution, or you may have scanned in some old photos that kind of faded or whatever you know. I think it's really useful thing to go to do and also it's, but it's a good project, because it's like really similar to what we're doing here but different enough that you'll come come across some interesting challenges on the way, I'm sure so I'm going to use Some imagenet again again, you don't need to use all of image net at all. I just happened to have it lying around? You can download the one percent sample of image net from faster faster. They are. You can use any set of pictures.

2. 00:07:30

- **Time-Series with Rossmann Store Sales (Kaggle)**
- **& 3rd place solution with "a very uncool NN ;-)".**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You have lying around honestly and in this case, as I say, we don't really have labels per se, so I just got a give everything a label of zero just so we can use it with our existing infrastructure more easily. Now because I'm in this case pointing at a folder that contains all of image net, I certainly don't want to wait for all of image net to finish to run an epoch. So here I'm just most of the time I would set keep percent to like one or two percent, and then I just generate a bunch of random numbers and then I just grab those keeps those which are less than 0.02 and so that lets be quickly subsample. My rows, all right, so we're going to use vgg 16 and vgg 16 is something that we haven't really looked at in this class. But it's a very simple, very simple model where we take and normal as you apply three-channel input and we basically run it through a number of 3x3 convolutions and then from time to time we put it through a 2x2 Maps pool and then we do a few More 3x3 convolutions max Paul, so on so forth, and then this is kind of our backbone. I guess and then we we don't do an average pooling layer in a deputy of average pooling layer. After a few of these, we end up with this. You know 7x7 grid as usual. I think it's about 7 by 7 by 512, something like that and so, rather than average polling.

We do something different, which is reflect the whole thing, so that spits out a very long vector of activations of size, 7 times 7 times 512 memory says correctly, and then that gets fed into two fully connected layers, each one of which has 409 6 activations and Then one more fully connected layer, which has, however, many classes. So if you think about it, the weight matrix here is huge. It's you know, 7 by 7 by 512 by 409 6, and it's because of that weight matrix, really that V gge went out of favor. Pretty quickly because it takes a lot of memory and takes a lot of computation and it's really slow and there's a lot of redundant stuff going on here, because really those 512 activations are not that specific to which of those 7x7 grid cells they're in right. But when you have this entire weight matrix here of every possible combination, it treats all of them, uniquely right and so that'll can also lead to generalization core ones, because there's just a lot of weights and so forth. My view is that there's you know that the approach that's used in every modern Network, which is here we do an adaptive average pulling care, ask that we be known as a global average calling for in fastai. We didn't really do a concat adaptive, concat pooling, which spits it straight down to a 512 long

activation. I think that's throwing away too much geometry.

So to me, probably the correct answer is somewhere in between and will involve some kind of um factored convolution or some kind of tensor decomposition, which yeah, maybe some of us can think about in the coming months. So for now, anyway, we've gone from one extreme, which is the adaptive average pulling to the other extreme, which is this huge, flattened, collocation player. So a couple of things which are interesting about vgg that make it still useful today. The first one is that there's there's more interesting layers going on here with with most modern networks, including the resident family. We, the very first layer generally, is a seven by seven pawns or something similar, which means we and that's tried to right, which means we throw away half the grid size straight away, and so this little opportunity to use the the fine detail, because we never do Any computation with it - and so that's a bit of a problem for things like segmentation or super resolution models, because the fine detail matters right. We actually want to restore it, and then the second problem is that the adaptive average polling layer entirely throws away the geometry in the last few sections, which means that the rest of the moral doesn't really have as much interesting kind of learning. That geometry - is it otherwise might, and so, therefore, for things which dependent on position any kind of localization based approach, tor, anything that requires generative model is going to be less effective.

So one of the things I'm hoping you're hearing is I describe this. Is that probably none of the existing architectures are actually ideal? We can invent a new one. Then, actually, I just tried inventing a new one over the week, which was to take the the vgg head and attach it to a resonate that one and interestingly I found. I actually got a slightly better classifier than a normal ResNet, but it also was something with a little bit more useful information. You know it took, I don't know five or ten percent longer to Train, but nothing worth worrying about yeah. I think you know. Maybe we couldn't in Resident replace this, as we've talked about briefly before this very early convolution, with something more like an inceptions dam which has a bit more computation. I think there's definitely room for some nice little tweaks to these architectures so that we can build some models which are maybe more versatile. You know at the moment people tend to build architectures that just do one thing: they don't really think you know. What am i throwing away in terms of opportunity? Because that's that's how publishing works you know you published, like they've, got the state of the art and this one thing rather than you have created something that's good at lots of things so um.

So, for these reasons we're going to use vgg today, even though it's it's ancient and it's missing lots of great stuff. One thing we are going to do, though, is use a slightly more modern version, which is a version of vgg where batch norm has been added. After all, the convolutions, and so in fastai. Actually, when you ask for a vgg Network, you always get the best norm, one because that's basically always what you want. So this is actually very Gigi with batch mode and there's a 16 in the 19. The 19 is way bigger and heavier and doesn't really is really any better. So we no one, really uses it. Okay, so we're going to go from 72 by 72. Lr is low resolution input, size, low resolution, we're going to initially scale it up by x, 2. We're the batch size of 60 or to get a two times 72, so one by 44 by 144 output. So that's gon na be our stage stage. One we'll create our own data set for this, and the data set it's very worthwhile. Looking inside the faster I dot data set module and seeing what's there because, just about anything you'd want, we probably have something: that's almost what you want. So in this case I want a data set where my X's are images, and my y's also images. So there's already a files data set we can inherit from where the x's are images, and then i just inherit from that and i just copied and pasted the get x and turn that into get y. So i just opens an

image.

So now I've got something where that X is an image and the y is an image and in both cases, what we're passing in is an array of flower names. I'm going to do some data augmentation, obviously with all of image net. We don't really need it, but this is mainly here, for you know anybody who's using smaller data sets to make the most of it. Random dihedral is referring to every possible 90-degree rotation and plus optional left/right flipping. So, though, the dihedral group of eight symmetries, normally, we don't use this transformation for image net pictures because, like you, don't normally flip blobs upside down, but in this case we're not trying to classify whether it's a dog or a cat. We're just trying to keep the general structure of it. So, actually, you know every possible flip is a reasonably sensible thing to do for this problem, so a creative validation set in the usual way, and you can see I'm kind of like using a few more slightly lower level functions. Generally speaking, I just copy and paste them out of the faster source code to you know find the bits I want so here's the bit, which takes an array of validation, set indexes and one or more arrays of variables and simply splits. So, in this case, the into a training and a validation set, and this into a training novella bit sorry, it yeah the training, validation set.

You give us our X's and our whites now, in this case the Train are the X and the y

3. 00:18:00

- **Implementing the Rossman solution with Keras + TensorFlow + Pandas + Sklearn**
- **Building Tables & Exploratory Data Analysis (EDA)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

At the same import image and our output image of the same we're going to use transformations to make one of them lower resolution, so that's why these are the same. The same thing, okay, so the next thing that we need to do is to create our transformations as per usual and we're going to use this transform Y parameter like we did for bounding boxes, but rather than use, transform type dot, coordinate we're going to use, transform Type pixel, and so that tells our transformations framework, that your Y values are images with normal pixels in them, and so anything you do the X. You also need to do so. The way do the same thing: okay and you need to make sure any data augmentation transforms you use, have the same parameter as well. Okay, so you can see the possible transform types. You basically you've got classification which we're about to use the segmentation in the second half of today coordinates no transformation at all or pixel. Alright, so once we've got a dataset class and some X & amp Y training and validation sets, there's a handy. Little method called get datasets, which basically runs that constructor over all the different things that you have to return all the datasets. You need in exactly the right format to pass pass to a model data constructor as a constructor, in this case the image data constructor.

So we're kind of like going back under the covers of fastai, a little bit and building it up from scratch, and you know in the next few weeks this will all be wrapped up and refactored into something that you can do in a single step. In fastai, but the point of this is to learn, you know a bit about going under the covers, so something we've briefly seen before is that when we take images in, we transform them, not just the data augmentation that we also move the channels dimension up To the start, we subtract the mean divided by the standard deviation

whatever. So, if we want to be able to display those pictures that have come out of our data sets or data loaders, we need to denormalize them, and so the model data objects. Data set has ad norm function. That knows how to do that. So I'm just going to give that a short name for convenience. So now I'm going to create a function that can show an image from a data set and if you pass in something saying this is a normalized image, then won'ting on it. Okay, so we can go ahead and have a look at that you'll see here. We've passed in size low-res as our size for the transforms and size high-res, as this is something new the size Y parameter. Okay, so the two bits are going to get different sizes, and so here you can see the two different resolutions of our X and our Y for a whole bunch of fish.

Okay, as you know, as per usual plot subplots, to create our two plots and then we can just use the different axes that came back to the stuff next to each other. So we can then have a look at a few different versions of the data transformation and there you can see them being clicked in all different directions. Okay, so let's create our model, so we're going to have an image coming in small image coming in and we want to have a big image coming out, and so we need to do some computation between those two to calculate what the big image would look like And so essentially, there's kind of two ways of doing that: computation. We could first of all do some up sampling and then do a few straight one kind of layers to do lots of computation or we could first do lots of straight one layers to do other computation and then at the end, do some up sampling. We've got to pick the second approach because we want to do lots of computation on something smaller, because it's much faster to do it. That way, and also like all that computation we get to kind of leverage during the up sampling process. So that's sampling. We know a couple of possible ways to do that we can use transposed or fractionally straited convolutions or we can use nearest neighbor up sampling, followed by a one by one conf and then in in the kind of do lots of computation section.

We could just have a whole bunch of 3x3 coms right, but in this case particular it seems likely that ResNet blocks are going to be better because really the output and the input are very, very similar right. So we really want a kind of a flow through path that allows as little fussing around as possible, except kind of a minimal amount necessary to do our super resolution. And so, if we use ResNet blocks, then they have an identity path already right so, like you can imagine the most simple version where it does like a you know: bilinear sampling, kind of approach or something it could basically just go through identity box all the way Through and then in the up sampling blocks just learn to take the averages of the inputs and get something that's like not too terrible. So that's what we're going to do we're going to create something with five ResNet blocks and then for each 2x scale-up. We have to do we'll, have 1/2 sampling look so they're all going to consist of obviously, as per usual convolution layers, possibly with activation functions after many of them. So I kind of like to put my standard convolution block into a function, so I can refactor it more easily as per usual, I just won't worry about passing in padding and just calculate it directly as kernel size over two.

So one interesting thing about a little comic block here is that there's, no, that's not, which is pretty unusual for ResNet type models and the reason there's no batch norm it because I'm stealing ideas from this fantastic recent paper, which actually won a recent competition in super Resolution performance and to see how good this paper is: here's kind of a previous state of the art there's SR ResNet right and what they've done here is they've zoomed way in to a an up, sampled kind of natural or fence. This is the original and you can see in the kind of previous best approach, there's a whole lot of distortion and blurring going on right or else in their approach. It's it's nearly perfect. Alright. So like it was a really big step up this paper. They call their model, EDS are enhanced deep residual networks and they did two things

differently to the kind of previous standard approaches. One was to take the ResNet blocks. This is a regular resident block and throw away the better. Not so why would they throw away the veteran or well the reason they would throw away? The batch norm is because batch norm changes stuff, and we want a nice straight through path that doesn't change stuff, okay, so the idea, basically here is like hey. If you don't want to fit all with the input more than you have to, then don't force it to have to calculate things like batching on parameters so throw away the and the second trick we'll see shortly.

Alright. So here's a con with no batch norm and so then we're going to create a residual block containing as per usual, two convolutions and, as you see in their approach they'd, even they don't even have a rail you after their second conf. Okay. So that's why I've only got activation on the first one. So a couple of interesting things here: one is that

4. 00:27:15

- **Digress: categorical variable encodings and "Vtreat for R"**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

This idea of like having some kind of main ResNet path like conv, relia cons and then turning that into a rail you block by adding it back to the identity. It's something we do so often I kind of factored it out into a tiny little module called res sequential, which simply takes a bunch of layers that you want to put into your residual path. Turns that into a sequential model. Runs it and then adds it back to the input right. So, with this tat all module, we can now turn anything like conf activation cons into a resonate lock just by wrapping it in res sequential okay. But that's not quite all I'm doing because, like normally a res block, just has that and it's forward, but I've also got that what's risco Briscoe is the number zero point one? Why is it there? I'm not sure anybody quite notice, but the short answer is that the guy who invented batch norm also somewhat more recently, did a paper in which he showed for, I think, the first time the ability to Train imagenet in under an hour and the way he did It was fire up, lots and lots of machines and have them work in parallel to create really large batch sizes. Now, generally, when you increase the batch size by order n, you also increase the learning rate by order n to go with it. So generally, a very large batch size training means very high learning rate training as well, and he found that with these very large batch sizes of like 8,000 plus or even up to 32,000, that at the start of training, his activations would basically go straight to infinity And a lot of other people have found that we actually found that when we were competing in dawn bench both on the sofa and the imagenet competitions, that you know, we really struggled to make the most of even the eight GPUs that we were trying to take Advantage of because of these kind of challenges with these larger batch sizes and taking advantage of them so something that a Christian found.

This research was that if he in the ResNet blocks, if he multiplied them by some number smaller than one something like point one or point two, it really helped stabilize training start and that's kind of weird, because let mathematically it's kind of identical right, because, obviously, whatever I'm multiplying it by here. You know I could just scale the weights by the opposite amount here and have the same number. Okay. So, but it's kind of like we're not dealing with

5. 00:30:15

- **Back to Rossmann solution**
- **& "Python for Data Analysis" (book)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Abstract math, you know we're dealing with, like you know, real optimization problems and different initializations and learning rates and whatever else, and so the problem of kind of whites disappearing off into infinity. I guess generally is really about that they're kind of the discrete and finite nature of computers in in practice, partly and so often yeah. These kind of little tricks can can make the difference. Alright. So in this case, we're just kind of toning things down based at least based on our initial initialization, and so there probably other ways to do this. For example, one approach from some folks at Nvidia called Lars le RS, which I briefly mentioned last week, is an approach which uses discriminative learning rates calculated in real time. Basically, looking at the ratio between the gradients and the activations to scale all learning rates by layer, and so they found that they didn't need this trick to scale it scale up the batch sizes, a lot, maybe a different initialization, which would be all that's necessary. The reason I mentioned this is not so much because I think a lot of you are likely to want to train on massive clusters of computers, but rather that I think a lot of you want to train models quickly, and that means using high learning rates and Ideally getting super convergence and I think these kinds of tricks are the tricks that we'll need to be able to get super convergence across more different, architectures and so forth, and you know other than Leslie Smith. No one else is really working on super convergence other than some fastai students nowadays.

So these kind of things about how do we train at very, very high learning rates? We're going to be have to be the ones who figure it out? Because, as far as I can tell, nobody else cares yet so so I think you know looking at the literature around, you know: training imagenet in one hour or more recently, there's now a train image net in 15 minutes. These papers actually tell, I think, have some of the tricks to allow us to train things at home learning rates, and so here's one of them, and so, interestingly other than the train image net1 our paper, the only other place I've seen this mentioned was in this Pds, our paper and it's really cool because, like I know, people who win competitions, I just find them to be very pragmatic and well-read. You know lucky they actually have to get things to work, and so this paper describes an approach which actually worked better than anybody else's approach, and they did these pathetic things like throw away batch norm and use this little scaling factor which almost nobody seems to know About and stuff like that, ok, so that's where the point one comes from. So basically, our super-resolution ResNet is down and do a convolution to go from our three channels to 64 channels, just to rich nut the space a little bit then also we've got actually a cannot 5h lots of these res blocks and we're just going to keep remember Every one of these res blocks is strike one, so the grid size doesn't change.

The number of filters doesn't change. It's just 64. All the way through well do one more convolution and then we'll do our app sampling by however much scale we asked for and then something I've added, which is a little idea, is just one batch norm here, because it kind of felt like it might be helpful. Just to scale the last layer and then finally a comb to go back to the three channels we want, so you can see that's basically here's lots and lots of computation and then a little bit of our sampling, just like we kind of described so the only Other piece here, then, is, and I also just dimension - you know, as you can see, as I'm tending to do now. This whole thing is done by creating just a list with layers and then at the end, turning that

into a sequential model, and so my forward function is as simple as can be, so here's our app sampling and up sampling is a bit interesting because it is Not doing either of these two things, so let's talk a bit about up sampling here is a picture from the paper from not from the competition winning paper, but from this original paper and so they're saying hey, our approach is so much better, but look at their Approach, it's got goddamn artifacts in it. Alright, these just pop up everywhere, and so one of the reasons for this is that they use transposed convolutions and we all know don't use transposed convolutions. Okay, so here are transposed convolutions.

This is from this fantastic convolutional arithmetic paper that was shown also in the Theano Docs. If we're going from the blue is the original image, so 3x3 image up to a 5x5 image right or a 6x6. If we added a layer of padding, then all a transpose convolution does, is it uses a regular 3x3 cons, but it sticks white. You know zero pixels between every pair of pixels, alright, so that makes the input image bigger and when we run this convolution life over. It therefore gives us a larger output. Okay, but I mean that's obviously stupid because when we get here, for example, of the nine pixels coming in eight of them, a zero so like we're just wasting a whole lot of computation and then on the other hand, if we're slightly off over here, then

6. 00:36:30

- **What Jeremy does everytime he sees a 'date' in a structured ML model**
- **& other tips**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Four of our nine and on zero, but yet we only have one filter like one kernel to use, so it can't like change depending on how many zeros are coming in, so it has to kind of be suitable for both and it's just not possible right. So we end up with these artifacts, so one approach we've learnt to make it a bit better, is to not put white things here, but instead to copy this pixels value to each of these three locations. Alright, so that's a just a nearest neighbor up sampling. That's certainly a bit better all right, but it's still pretty crappy, because now still when we get to these nine here, four of them are exactly the same number all right and when we move across one then now we've got. You know a different situation entirely right and so to on where we are so in particular, if we're here, you know, there's going to be a lot less repetition. So again we have this problem where there's like wasted computation and too much structure in the data and it's going to lead to RFS again. So up sampling is better than transposed convolutions, it's you know better to copy them, rather than replace them with zeros. But it's still not quite good enough, so instead we're gon na do the pixel shuffle. So the pixel shuffle is an operation in this sub pixel convolutional neural network and it's a little bit mind-bending, but it's kind of fascinating, and so we start with our input.

We go through some convolutions to create some feature Maps for a while, until eventually we get to layer - and I we go to this layer - I minus one which has n I minus one feature Maps we're going to do another 3x3 cons and our goal here is To go from a 7x7 grid cell we're going to go a 3x3 up scaling, so we're going to go up to a 21 by 21 grid cell. So how do we what's another way? We could do that to make it simpler. Let's just pick one face, just one filter, so we'll just take the topmost filter and just do a convolution over that just to see what happens and what we're going to do is we're going to use a convolution where the kernel size

is. Is the number of filters is nine times bigger than we strictly speaking need? So if we needed 64 filters, we're actually going to do 64 times nine filters. Why is that right, and so uh here are: is the scale effect uh? So three right, so a squared 3 squared is 9. So here are the nine filters to cover one of these input layers, one of these input slices, but what we can do is we started with seven by seven and we turn it into seven by seven by nine right. Well, the output that we want is equal to 7 times 3 by 7 times 3, so in other words, there's an equal number of pixels here activations here, as there are our activations here, so we can literally reshuffle these seven by seven by nine activations to create This 7 by 3, by 7 by 3 Matt. And so what we're going to do is we're going to take one little kind of tube here on the top left hand of each grid and we're going to put the purple one up in the top left.

And then the blue one one to the right and then the light blue one one to the right of that and then the slightly darker blue one and the middle of the far left, the green one in the middle and so forth. So each of these nine cells in the top left are going to end up in this little 3x3 section of our grid and then we're going to take. You know 2, comma 1 and take all of those 9 and move them to these 3 by 3. Part of the grid and so on and so forth, right and so we're going to end up having every one of these 7 by 7 by 9 activations inside this 7 by 3 by 7 by 3 image. So the first thing to realize is yes, of course, this works under some definition of works, because we have a learn: herbal convolution here and it's going to get some gradients which is going to do the best job it can of filling in the correct activation such That this output is the thing we want alright, so the first step is to realize: there's nothing particularly magical here. You know we can. We can create any architecture we like. We can move things around any. How we want to - and you know our wipes in the convolution - will do their best to do all we asked the real question is: is it good idea you know, is this an easier thing for it to do? You know and a more flexible thing for it to do, then the transposed convolution or the up sampling, followed by one by one month, and the short answer is yes, it is, and the reason it's better in short, is that the convolution here is happening in the Low resolution, seven by seven space, which is quite efficient, where else, if we first of all up sampled and then did our cons, then our con would be happening in the 21 by 21 space, which is a lot of computation right and furthermore, as we discuss there's A lot of replication in redundancy in the nearest neighbor sample version, so they actually show in this paper they actually, in fact, I think they have a follow-up technical note, where they kind of provide some more mathematical details as to exactly what work is being done and Show that the work really is more efficient, this way.

Okay. So that's what we're going to do all right, so we're going to have for our

7. 00:43:00

- **Dealing with duration of special events (holidays, promotions) in Time-Series**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

App sampling, we have two steps. The first will be a three by three cons with R squared times more channels than we originally wanted, and then a pixel shuffle operation which moves everything in each grid cell into the little by our grids that are located throughout here. Okay, so here it is it's one line of code right and so here's the cons from number of in to number of filters out times four, because we're doing a scale to that sample all right. So two squared is four. So that's our convolution, and then here is our pixel. Shuffle it's built into high touch

pixel shuffle is the thing that moves each thing into its right spot, so that will increase will up sample by a scale factor of two, and so we need to do that. Log base two scale time, so, if scale is for two times to go two times two, you go okay. So that's what this up sample here does great guess what that does not get rid of the checkerboard patterns. We still have checkerboard patterns, so I'm sure in great fury and frustration this same team from Twitter. I think this is back when they used to be at a startup called magic Pony that Twitter thought came back again with another paper saying: okay, this time we've got rid of the checkerboard okay. So so why do we still have? As you can see here, you still have a checkerboard right and so the reason we still have a checkerboard even after doing this is that when we randomly initialize this convolutional kernel at the start, it means that each of these nine pixels in this little 3x3 grid Over here are going to be totally randomly different, but then the next set of three pixels will be randomly different to each other, but will be very similar to their corresponding pixel in the previous 3x3 section.

So we're going to have repeating 3x3 things all the way across and so then, as we try to learn something better. It's starting from this like repeating 3x3 starting point, which is not what we want right. What we actually would want is for these three by three pixels to be the same. To start with, so to make these three by three pixels the same, we would need to make these nine channels the same here right for each filter and so the solution and his paper is very simple. It's that when we initialize this convolution that start when we randomly initialize it, we don't totally randomly initialize it. We randomly initialize one of the the r-squared sets of channels, and then we copy that to the other R squared, so they're all the same, and that way initially each of these three by threes will be the same, and so that is called I CNN, okay and That's what we're going to use in a moment so before we do. Let's take a quick look, so we've got this super resolution resinate, which just does lots of computation. You know with lots of ResNet blocks and then it does some up sampling and gets our final. Three channels out and then to make life faster, we're going to run this in parallel. One reason we want to run it in parallel is because Jurado told us that he has six GPUs, and this is what his computer looks like right now, and so I'm sure anybody who has more than one GPU has had this experience before.

So how do we get? How do we get these men working in together? All you need to do is to take your pytorch module and wrap it with N n data parallel, okay and once you've done that it copies it to each of your GPUs and will automatically run it in parallel. It scales pretty well done to two GPUs. Okay to three GPUs, better than nothing to four GPUs and beyond that performance does two go backwards: the by default. It will copy it to all of your GPUs. You can add an array of GPUs. Otherwise, if you want to avoid getting in trouble, for example, I have to share our box with you net and if I didn't put this here, then she would be yelling at me right now. Well, maybe you know, or according my plus, so this is how you avoid getting into trouble with you net. So one thing to be aware of here is that once you do this, it actually modifies your module. So if you now print out your module, let's say previously, it was just an endless sequential now, you'll find it's an N in dots as Crenshaw embedded inside a module called module right, and so, in other words, if you save something which you had n end updated Paralleled and then tried and load it back into something that you hadn't and end up beta paralleled, it'll say it doesn't match up, because one of them is embedded inside this module attribute and the other one isn't. It may also depend even on which GPU IDs, you have had a coffee too, so two possible solutions.

One is don't save the module M, but instead save the module, attribute m dot module because that's actually the the non data parallel bit or always put it on the same GPU IDs and then use

data parallel and load and save that every time that's what I Was using this will be an easy thing for me to fix automatically in fastai and I'll. Do it pretty soon so it'll look for that module, attribution and deal with it automatically, but for now we have to do it manually. It's probably useful, to know. What's going on behind the scenes anyway? Alright, so we've got our module. You know I find it overrun like 50 or 60 percent faster on a 1080 TI. If you're running on voltar, it actually paralyzes a bit better. There's a there. Much faster ways to parallel parallel lives, but this is like a super super easy way all right, so we created our learner in the usual way. We could use MSA loss here. So that's just going to compare the pixels of the output to the pixels. You know that we expected and we can run our learning rate finder and we can train it for awhile and here's our input and here's our output, and you can see that what we've managed to do is to train a very advanced residual convolutional net work. That's learnt to blur things. Why is that? Well, because it's what we asked for, we said to minimize MSE loss right, an MSA lost between pixels, really, the best way to that is just average. The pixels I eat a blur.

So that's why pixel lusts, no good! So we want to use our perceptual loss, so let's try perceptual us right so with perceptual loss, we're basically going to take our vgg network and just like we did last week we're going to find the block index just before we get a max ball. Okay. So here are the ends of each kind of block of the same grid size and if we just print them out as we'd expect, every one of those is a value module, and so in this case, these last two blocks are less interesting to us. They're kind of the grid size there is small enough. You know kind of coarse enough that it's not as useful for super resolution, so we're just going to use the first three and so just to save unnecessary computation. We're just going to use those first 23 layers or vgg we'll throw a way to look at the rest, we'll stick it on the GPU. We're not going to be training! This speech EG model at all, we're just using it to compare activations so we'll stick it in

8. 00:52:00

- **Using 'inplace=True' in .drop(), & a look at our final 'feature engineering' results**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Eval mode and we will set it to not trainable okay, just like last week, we will use a save features class to through a forward hook, which saves the output activations at each of those layers, and so now we've got everything we need to create our perceptual Loss so as I call it here, feature loss plus right and so we're going to pass in a list of layer IDs. You know the layers where we want the content loss to be calculated and array of weights a list of weights for each of those layers. So we can just go through each of those layer IDs and create an object which is going to store, which is you know, I've got the book function forward, hook, function to store the activations and so in our forward. Then we can just go ahead and call the forward pass of our model with the target. So the target is the hi, whereas image we're trying to create okay, and so the reason we do. That is because that's going to then call that book function and store in soft save features the activations. We want right now we're going to need to do that for our confident output as well right. So we need to clone these, because otherwise the confident output is going to go ahead and just plop up what I already had. Okay. So now we can do the same thing for the confident output, which is the input to the loss function, and so now we've got those two things we can zip them all together, along with the weights, so we've

9. 00:53:40

- **Starting to feed our NN**
- **& using 'pickle.dump()' for storage encodings**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Got inputs targets and weights, and then we can do the l1 loss between the inputs and the targets and multiply by the layer weights. The only other thing I do is, I also grab the pixel loss right, but I weight it down quite a bit. Okay and most people don't do this - I haven't seen papers that do this, but in my opinion it's maybe a little bit better because you've got you know the perceptual content lost activation stuff, but you know the really finest level. It also cares about the individual pixels. Okay, so that's our last function. We create our super resolution, ResNet telling it how much to scale up by and then we're going to do our I see in our initialization of that pixel shuffle convolution right. So there's really like it's. This is very, very boring code. I actually stole it from from somebody else. Like literally all it does is just say: okay, you've got some weight, tensor X, that you want to initialize, so we're going to treat it as if it had shape divided by so number of features divided by scale. Squared features in practice so, like you know this might be 2 squared before, because we actually want to copy you know we want to just keep set with them and then copy them four times. So we divide it by four and we create something of that size and we initialize that with by default timing, normal initialization, and then we just make scale squared copies of it.

Okay and the rest of its just kind of moving axes around a little bit. All right, so that's kind of return, a new weight matrix where each each initialized sub kernel is repeated R, squared or scale squared times, so that details don't matter very much. All that matters here is that I just looked through to find what was the actual layer. The cone flower, just before the pixel shuffle and stored it away, and then I called I see an R on its weight matrix to get my new weight matrix and then I copied that new weight matrix back into that layer. Ok, so, as you can see, I went to quite a lot of trouble in this exercise to really try to implement all the best practices right and I kind of tend to do things a bit one extreme or the other. I show you like a really happy version that only slightly works or I go to the enth degree to make it work really well right, and so this is a version where I'm claiming that this is pretty much a state of the art implementation. It's a competition. Winning or at least my reimplementation of a competition winning approach and the reason I'm doing that is because I think, like this is one of those rare papers where they actually get a lot of the details right and I kind of want you to get a feel Of what does it feel like to get all the details right and you know remember getting the details right - is the difference between this hideous blurry mess. You know, and this really pretty exquisite result.

Okay, so so we're gon na have a to do potato parallel on that again, we're going to set our criterion to be feature loss using our vgg model grab the first few blocks, and these are assets of layer weights that I found worked pretty well. Do a learning rate finder fit it for a while, and I fiddled around for a little while trying to kind of get some of these details right, but here's the my favorite part if the paper is what happens next now that we've done it for scale equals To progressive resizing right, so progressive resizing is the trick that let us get the best single computer result for image net training on one bench. It's this idea of starting small gradually making bigger. I only know of two papers that have used this idea. One is the progressive resizing of gans paper, which allows training a very high resolution gains and the

other one is. The ideas are, and the cool thing about progressive resizing is not only are your earlier. Epochs assuming you've got two by two smaller four times faster. You can also make the batch size - maybe three or four times bigger, but more importantly, they're - going to generalize better because you're feeding your model different sized images during training right. So we were able to Train like half as many epochs for imagenet as most people. So our epochs were faster and they were fewer of them.

So progressive resizing is something that you know, particularly if your training from scratch, I'm not so sure, if it's useful for fine-tuning transfer learning, but if you're training from scratch, you probably want to do nearly all the time. So the next step is to go all the way back to the top right and change to full scale. Thirty-Two batch size right, like restart so I saved the model before I do that, go back and that's why there's a little bit of fussing around in here with reloading, because what I needed to do now is I needed to load my saved model back in. But there's a slight issue, which is: I now have one more up sampling layer than I used to have to go from two by two to four by four. My little. My little loop here is now looping through twice not once, and therefore it's added an extra convent, an extra pixel shuffle. So how am I going to load in weights for a different network? And the answer is that I use a very handy thing in pytorch, which is if I call that this is what this is. Basically, what learned load calls behind the scenes load state kicked if I pass this parameter strict equals false. If I pass in this parameter strict equals false that it says. Okay, if you can't fill in all of the layers just fill in the lay as you can so after loading the model back in this way, we're going to end up with something where it's loaded in all the layers that it can and that one comp layer.

That's new is going to be randomly initialized, all

10. 01:00:45

- **“Their big mistake” and how they could have won #1**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Right and so then I freeze all my layers and then unfreeze that upsampling part right and then use I CNR on my newly added extra layer right and then I can go ahead and wear again and so then the rest is the same. So, if you're trying to replicate this, don't just run this top to bottom. Okay realize it involves a bit of jumping around okay yeah, the longer you train, the better it gets. I ended up training it for about 10 hours, but you'll still get very good results. Much more quickly if you're, less patient, and so we can try it out - and here is the result. Here is my pixelated bird and look here: it's like totally random e pixels and here's the upsampled version. It's like, it's literally invented color, it coloration, but it figured out what kind of bird it is right and it knows what the feathers are metal look like, and so it has imagined a set of feathers which are compatible with these exact pixels, which is like genius. Like saying here at there's no way you can tell what these blue dots are meant to represent, but if you know that this kind of bird has an array of feathers here, you know that's what they must be right and then you can figure out where the Feathers would have to be such that when they were pixelated they'd end up in these pots, all right, so it's like literally reverse engineered, like given its knowledge of this exact species of bird, how it would have to have looked to create this output, and so this Is like so amazing, it also knows from all the kind of signs around it that this area here was was almost certainly blurred out.

So it's actually reconstructed blurred vegetation and you know if it if it hadn't have done all of those things it wouldn't have got. Such a good loss function right because in the end it had to match. You know the activations saying like oh there's a feather over here and it's kind of fluffy looking and it's you know in this direction and all that all right. Well, that brings us to the end of super resolution. Don't forget to check out the ask Jeremy anything's threaded and we will do some Astro me anything after the break but see you back here a quarter to eight okay, so we are going to do. Ask Jeremy anything Rachel will tell me the most voted up of your questions. Yes, Rachel. What are the future plans for fastai in this course? Will there be a part three? If there is a part three, I would really love to take it. Oh I'm not! Quite sure I always had to guess I hope there'll be some kind of follow-up. Last year after part, two one of the students started up a weekly book club going through the Ian Goodfellow deep learning book and Ian actually came in and presented quite a few of the chapters and other people like there's somebody an expert who presented every chapter. That was really that was like a really cool part. Three and four that extended will depend on I'm you, the community, to come up with ideas and help make them happen and yeah, and I'm definitely keen to to help. I've got a bunch of ideas, but I'm nervous about saying them because I'm not sure which ones will happen and which ones won't, but the more support I have in making things happen that you want to happen from you, the more likely they are.

What was your experience like starting down the path of entrepreneurship? Have you always been an entrepreneur, or did you start at it start out at a big company in transition to a startup? Did you go from academia to startups or startup stack edenia? No, I was definitely not an academia, I'm totally a fake academic. I I

11. 01:05:30

- **Splitting into Training and Test, but not randomly**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

I started at McKinsey and company, which is a strategy firm when I was 18, which meant I couldn't really go to university, so I didn't really turn up and then yeah spent eight years in business, helping really big companies on strategic questions. I always wanted to be an entrepreneur planter on you spend two years in McKinsey. Only thing I really regret in my life was not sticking to that plan and wasting eight years instead. So two years would have been perfect, but yeah. Then I went into burner. Ship started two companies in Australia and the best part about that was that I didn't get any funding. So all the money that I made was mine or the decisions were mine and my you know and my partner's you know I focused entirely on on profit and product and customer and service, whereas I find in San Francisco. I'm glad you know I'm glad I came here, and so the two of us from you know came here for cable and EMI and raised. You know ridiculous amount of money, eleven million dollars for this really new company. That was really interesting, but it's also really distracting. You know trying to worry about scaling and VC's wanting to see what your business development plans are and also just not having any real need to actually make a profit and yeah. So I had a bit of the same problem at analytic where I again raised a lot of money: fifteen million dollars pretty quickly and yeah a lot of distractions, so yeah.

I think you know trying to bootstrap your own company and focus on making money by selling something at a better profit, and then you know plowing that back into the company. It

worked really well right because within like five years, you know we were making a profit from three months in and within five years, we're making. You know enough for profit, not just to pay all of us in their own wages, but also to see my bank account growing and after ten years sold it for a big chunk of money, not enough that a VC would be excited. But enough that I didn't have to worry that money again, you know, so I think yeah bootstrapping a company, something which people in the Bay Area's don't seem to appreciate how good our idea, that is, if you were 25 years old today and still know what you Know where which of you looking to use AI? What are you working on right now are looking to work on in the next two years.

12. 01:08:20

- **Why they modified their Sales Target with 'np.log()/max_log_y'**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

You should ignore the last part of that I won't even answer it doesn't matter where I'm looking like what you should do is leverage your knowledge about your domain. So, like one of the main reasons we do, this is to get people who have backgrounds in whatever recruiting you know. Oil field surveys, journalism, activism, whatever right and solve your problems, it'll be really obvious to you. What real problems are and it'll be really obvious to you, what data you have and where to find it? Those are all the bits that, for everybody else, that's really hard. So people who start out with like oh, I know deep learning now I'll, go and find something to apply it to basically never succeed. Where else people who are like. Oh, I've been spending 25 years doing specialized recruiting for legal firms, and I know that the key issue is this thing, and I know that this piece of data totally solves it, and so I'm just going to do that now and I already know who to call Who actually start selling it to? You know they're the ones who who tend to win so yeah and and and you know, if you, if you've done nothing but like academic stuff, that it's more, maybe about your your hobbies and interests. You know so everybody has hobbies. The main thing, I would say is: please don't focus on building tools for data scientists to use or for software engineers to use, because every data scientist knows about the market of data scientists, whereas only you know about the market. For you know, analyzing oil survey world walks, or you know, understanding, audiology studies or whatever it is that you do given what you've shown us about applying transfer learning from image recognition to NLP. There looks to be a lot of value in paying attention to all of the developments that happen across the whole machine learning field and that, if you were to focus in one area, you might miss out on some great advances in other concentrations.

How do you stay aware of all the advancements across the field while still having time to dig in deep to your specific domains? Yes, awesome. I mean that's kind of the message of this course. One of the key messages this course yeah. It's like lots of good works being done in different places, and people are so specialized. Most people don't know about it like if I can get stated that results in an LP within six months of starting to look at NLP, and I think that says more about NLP than it does about me. Thankfully, so yeah it's kind of like the Entrepreneurship thing. It's like you pick that the areas you see that you know about and kind of transfer stuff like. Oh, we could use deep learning to solve this problem or, in this case, like

13. 01:11:20

- **A look at our basic model**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

We could use you know this idea can compute a vision to solve that problem, so things like trendier may transfer learning, i'm sure there's like a thousand things opportunities for you to do in other fields to do what Sebastian and I did in NLP with NLP classification. So the short answer to your question is the way to stay ahead of, what's going on would be to follow my feed of Twitter favorites, and my approach is to and follow lots and lots of people on Twitter and put them into the Twitter favorites. For you, like literally I every time I come across something interesting, I click favorite, and there are two reasons I do it. The first is that when the next course comes along, I go through my favorites to find which things I want to study. The second is so that you know you can do the same thing and then you know which do you go deep into it almost doesn't matter like I find every time I look at something it turns out to be super interesting and important. So tiss like pick something which is like you feel like solving, that problem would be actually useful to some reason and it doesn't seem to be popular which is kind of the opposite. With what everybody else does. Everybody else works on the problems which everybody else is already working on, because they're, the ones that seem popular - and I don't know - I can't quite understand this chain of thinking - but it seems to be very common - is deep learning and overkill to use on tabular data When is it better to use deep learning instead of machine learning on tabular data? Is that a real question, or did you just put that there so that I would point out that Rachel, Thomas just wrote an article? So yes, so Rachel's just written about this and and original, and I spent a long time talking about it and the short answer is: we think it's great to use deep learning on tabular data actually of all the rich, complex, important and interesting things that appear in Rachel's, Twitter stream, covering everything from the genocide of the Inga through to latest ethics violations in AI companies.

The one by far that got the most attention and engagement from the community was their question about. Is it called tabular data or structured data? So ya ask computer Pires people how to name things and you'll, get plenty of interest yeah and there's some really good links here to stuff from instacart and pinterest and other folks in this area. Any of you that went to the data institute conference will have seen Jeremy Stanley's presentation about the really cool work they did and instacart. Yes, we're sure, so I relied heavily on lessons 3 and 4 from part one and writing this post. So yes, much of that may be familiar to you. Yeah Rachel asked me during the post like how to tell whether you should use the decision tree ensemble like GBM or random forest or or a neural net, and my answer is, I still don't know nobody to my nobody. I'm aware of has done that research and any particularly meaningful way, so there's a question to be answered there. I guess

14. 01:14:45

- **Training our model and questions**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

My approach has been to try to make both of those things as accessible as possible through the faster your library, so you can try them both both and see what works. Yes, that's what I do, and that is it for the top voted questions. Thank you. Okay. So, just quickly to go from super resolution to style transfer, it's kind of. Oh, I think I miss the one on reinforcement, learning and reinforcement. Learning popularity has been on a gradual rise in the recent past. What's your take on reinforcement, learning, which fast day I consider covering some ground and popular RL techniques in the future? I'm still not a believer in reinforcement learning. I think it's a an interesting problem to solve, but it's not at all clear that we have a good way of solving this problem. So the problem - it really is the delayed credit problem. So you know I want to learn to play pong. I've moved up or down and three minutes later I find out whether I won the game of pong which actions I took were actually useful and so to me the idea of calculating the gradients of those inputs with respect. You know the app so the gradient of the output with respect to those inputs. The credit is so delayed that those derivatives don't seem very interesting and there's been, you know, kind of been. I get this question quite regularly in every one of these four courses.

So far, I've always it the same thing I'm rather pleased, but finally, recently there's been some results showing that actually, basically random search, often does better than reinforcement learning. So basically, what's happened is very well-funded. Companies with vast amounts of computational power, throw all of it at

15. 01:16:45

- **Running the same model with XGBoost**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Reinforcement learning problems and get good results and people then say: oh it's because of the reinforcement learning rather than the vast amounts of compute power, or they use extremely thoughtful and clever algorithms, like a combination of convolutional, neural nets and Monte Carlo tree search like they did With the alpha girl stuff to get great results and people incorrectly say: oh that's because of reinforcement learning when it wasn't really reinforcement learning at all. So I'm very interested like in solving these kind of more generic optimization type problems rather than just prediction problems and that's what these delay to credit problems tend to look like, but I don't think we've yet got good enough best practices that I have anything on ready To teach and say, like I've got to teach you this thing, because I think it's still going to be useful next year, so we'll keep watching and yeah see see what happens. Ok, so we're going to now turn the super resolution Network, basically into a style transfer network, and what do this pretty quickly? We basically already have something so here's my input image and I'm going to have some loss function and I've got some neural net again. So, instead of a neural net, that does a whole lot of compute and then does up sampling at the end. Our input, this time is just as big as our output, so we're going to do some down sampling first and then our compute and then our settlement. Okay, so that's the first change.

We're going to make is going to add some down sampling so since tried to convolution layers to the front of our network. The second is, rather than just comparing YC and X, to the same thing here right, so we're going to basically say our input. Image should look like itself by the end, and so specifically we're going to compare it by checking it through vgg and comparing it at one of the content at one of the activation layers. And then its style should look like some

painting which brought to it. Just like we did with the gaddy's approach by looking at the grand Matrix correspondence at a number of layers, so that's basically it and so that that ought to be super straightforward. It's really just combining two things: we've already done, and so all this code starts identical, except we don't have high res and low res. We just have one size 256. Well, this is the same. My model is the same. One thing I did here is I I made I did not do any kind of fancy best practices for this one at all, partly because there doesn't seem to be any like there's been very little, follow up in this approach compared to the super resolution stuff and We'll talk about why in a moment so you'll see this is like much more normal looking. You know I've got batch norm layers. I don't have the scaling factor here.

You know I don't have a pixel shuffle, that's just using a normal up sampling, followed by

16. 01:20:10

- **“The really, really, really weird things here !”**
- **& end of the Rossmann competition;-)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

One by one cons, so it's kind of it's just more normal one thing they mentioned in the paper. Is they had a lot of problems with zero, padding, creating artifacts and the way they solve? That was by adding 40 pixels of reflection padding at the start. So I did the same thing and then they used zero padding in there convolutions in there red blocks. Now, if you've got zero padding in your convolution in your red blocks, then that means that you're, the two parts of your resin, it won't add up anymore, because you've lost a pixel from each side on each of your two convolutions. So my my red sequential, has become res sequential Center and I've removed the last two pixels on each side of those good sus, okay, so other than that. This is basically the same as what we had thought. So then we can bring in our starry night picture. We can resize it, we can throw it through our transformations just to make the method a little bit easier for my brain to handle. I took my transformation image which, after transfer our transform style in after transformations of three by 256 by 256, and I made a mini batch. My batch size is 24 24 copies of it. Now it just makes it a little bit easier to do the kind of batch arithmetic without worrying about some of the broadcasting they're not really 24. Copies are used, MP, dot broadcast to to basically fake 24 piece. Okay, so just like, before we create a V Gigi grab the last block this time we're going to use all of these layers.

So we keep everything up to the forty-third layer, and so now our combined lasts is going to add together a content loss for the third block, plus the gramm loss for all of our blocks with different weights. And so the gram loss against that kind of going back to everything being is like normal as possible. I've gone back to using MSA here. Basically, what happened as I had a lot of trouble getting this to train properly, so I gradually removed trick after trick and eventually just went. Ok, that's good, I believe make it as as bland as possible. Last week's Graham matrix was wrong by the way it only worked for a batch size of one, and we only had a batch size of one. So that was fine. I was using matrix multiply, which meant that every batch was being compared to every other batch. You actually need to use batch matrix multiply, which does a matrix multiply per batch. Ok, so that's something to be aware of there. Ok, so so I've got my grand matrices. I do my MSE loss between the gram matrices. I weight them my style weights, so I create that ResNet so create my style. My combined loss passing in the vgg Network, passing in the block IDs passing in the transformed starry night

image, and so you'll see the very start. Here. I do a forward pass through both vtg model with that starry night image, in order that I can the features for it right now notice.

It's really important now that I don't do any data augmentation, because I've saved the style features for a particular. You know non Augmented version, and so, if I augmented it, it might make some minor problems, but that's fine, because I've got all of imagenet to deal with. I don't really need to do data augmentation anyway. Okay, so I've got my loss function and I can go ahead and fit and there's really nothing flavor here at all. At the end, I have my some layers equals false, so I can see what each part looks like and see that there is some we balanced and I can finally pop it out. So I mentioned that should be pretty easy, and yet it took me about 4 days because it just I just found this incredibly fiddly to actually get it to work so like when I finally got up in the morning. I said to Rachel guess what they're trained correctly Rachel was like. I never thought that was going to happen it just it just looked awful all the time and it's really about getting the exact right mix of content, lossless, a style loss of the mix of the layers of the style loss and that the worst part was it Takes a really long time to train the damn CNN, and I don't didn't really know how long to train it before before I decided it wasn't doing well like, should I just train it for longer or what and I don't know all the little details didn't seem To like slightly change it, but just like it would totally fall apart all the time.

So I kind of mentioned this partly to say like just remember. The final answer you see here is after me, driving myself, crazy or weak over nearly always not working until. Finally, the last minute it finally does for even for things which just seemed like they couldn't pass be difficult, because that is combining two things we already have working. The other is like to be careful about how we interpret what authors claim yeah, so it was so fiddly getting this style transfer to work and like after doing it, it left me thinking. Why did I bother? Because now I've got something that takes hours to create a network that can turn any kind of photo into one specific

17. 01:26:30

- **Taxi Trajectory Prediction (Kaggle) with “another uncool NN” winner**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Style, it just seems very unlikely. I would want that for anything like about the only reason I could think that being useful would be to like do some art and stuff on a video, rather to turn every frame into some style like it's incredibly interesting to what to do. But you know when I looked at the paper that you know their tables saying like oh we're a thousand times faster than the Gattis approach, which is like it's just such an obviously meaningless thing to say in such an incredibly kind of misleading thing to say, because It ignores all the hours of training for each individual style, and I don't know I find this frustrating because, like a groups like this, Stanford group clearly know better or ought to know better, but still, I guess the academic community. He kind of encourages people to make these ridiculously grand plans and it also completely ignores this incredibly sensitive fiddly training process. So you know this paper was just so well accepted when it came out. You know I remember everybody getting on Twitter and being like wow. You know these Stanford people have found this way of doing style, transfer a thousand times faster and clearly you know the people saying this would like all like top researchers in the field. Clearly, like none of them actually understood

because nobody said you know, I don't see why this is remotely useful and also I tried it and it was incredibly fiddly to get it all to work, and so it's not until like what is this they're, like eighteen months Later or something that I finally coming back to it and kind of thinking like wait a minute, this is kind of stupid, and so so this is the answer I think to the question of like well, why haven't people don't follow ups on this to like create Really amazing best practices and better approaches like with a super resolution, part of the paper, and they I think the answer is because it's done so.

I think this part of the paper is clearly not fun. You know and it's been improved and improved and improved, and now we have great super resolution and I think we can derive from that great noise reduction. Great colorization great, you know, slant, removal, great, interactive and effective or whatever. So I think, there's a lot of really cool techniques here and it's also delivering a lot of stuff that we've been learning and getting better and better at okay. So then, finally, let's talk about segmentation. This is from the famous some cam vid data set, which is a classic example of an academic segmentation data set, and basically you can see what we do is we start with a picture they're. Actually video frames in this data set like here and we construct. We have some labels, where they're not actually colors, that each one has an ID and the IDS of math, two colors so like red might be, one purple might be two like pink might be three, and so all the buildings. You know one class or the cars or another class or the people or another class or the road is another class, and so what we're actually doing here is multi-class classification for every pixel, okay, and so you can see sometimes that model class specification really is quite Tricky there's you know like like these branches. Well, though, sometimes the labels are really not that great. You know this is very coarse, as you can see so here at traffic lights so forth.

So, but that's what we're going to do we're going to do. This is segmentation, and so it's a look like bounding boxes. Okay, but you know rather than just finding you know a box around each thing: we're actually going to label every single pixel with its plus and really that's actually a lot easier because it fits our CNN style so nicely that we've. Basically, we can create any CNN where the output is an N by M grid containing the integers from 0 to C, where there are C categories, and then we can use cross-entropy loss with a softmax activation and we're done right so like I could actually stop the Class there and you can go and use exactly the approaches. You've learnt in like lessons 1 & amp, 2 and you'll get a perfectly. Okay result. Ok, so the first thing to say is like this is not actually a terribly hard thing to do, but we're going to try and do it really well, and so, let's start by doing it, the really simple way and we're going to use the CAG or carvanha Competition, so you could go cable car Varner to find it. You can download it with the caracal api as per usual. Now, basically, there's a train folder containing a bunch of images which is the independent variable and a train masks, folder, there's the dependent variable and they look like this. Here's the here's one of the independent variable and here's one of the dependent variable okay.

So in this case, just like cats and dogs we're going simple rather than doing multi-class classification we're going to do binary classification, but of course multi-class is just the more general version. You know categorical, cross-entropy, your binary, cross-entropy, okay, so there's no differences conceptually. So we've got this is just you know, zeros and ones where else this is a regular image. So, in order to do this well, it would really help to know what cars look like right, because you know really we just what to do is figure out. This is a car and its orientation and then color, you know, put white pixels where we expect the part of e based on the picture and their understanding of what cars will plug. The original data set came with these CSV files as well. I don't really use them for very much other than getting the list of

images from them. Each each image after the car ID has a 0 1, 0 2, etc of which I've printed out all 16 of them for one car and, as you can see, if basically those numbers are the 16 orientations of one car. That is, and I don't think anybody in this competition actually used this orientation information. I believe they all kick the cars images just treated them separately. These images are pretty big, like over a thousand by thousand in size, and just opening the JPEGs and resizing them is slow, so I processed them all. Also OpenCV can't handle gif files.

So I converted them yes, Rachel, the question: how would somebody get these masks for training, initially Mechanical Turk or something yeah yeah, just a lot of boring work. You know probably some tools that help you with a bit of edge, snapping and stuff so that the human can kind of do it roughly and then just find you in the bits it gets wrong. Yeah, these kinds of labels are expensive. You know, and so one of the things I really want to work on is deep learning enhanced interactive labeling tools because you know yeah, so I've got a little section here that you can run. If you want to. You probably want to which converts the gifts into pngs, so just open it up with PIL and then save it as PNG, because open CV doesn't have give support and, as per usual for this stuff, I do it with a thread pool. So I can take advantage of parallel processing and then also create a separate directory train 128 and train masks 128, which contains the 128 by 128 resized versions of them, and this is the kind of stuff that keeps you sane. If you do it early in the process, so anytime you get a new data set, you know seriously think about creating a you know smaller version to make life fast anytime. You find yourself waiting on your computer. You know try and think of a way to create a smaller version, so yeah after you grab it from cowboy. You probably want to run this stuff by way to have lunch come back and when you're done you'll have these smaller directories, which we're going to use here 128 by 128 pixel versions to start with, so here's a cool trick if you use the same access object To plot an image twice and the second time you use alpha, which, as you might know, means transparency in the computer vision world, then you can actually plot the mask over the top of the photo, and so there here's a nice way to see all the masks On top of the photos for all of the cars in one group, this is the same match.

Files data set we've seen twice already. This is all the same code we used and here's something important, though, if we had something that was in the training set, go to this image and then the validation had that image. That would kind of be cheating, because it's the same Cup. So we use a contiguous set of car IDs and since each set is a set of 16, we make sure that's evenly divisible by 16, so we make sure that our validation set contains different car IDs to our training set. This is the kind of stuff, but you've got to be careful of on Kaggle. It's not so bad you'll know about it, because you'll submit your result and you'll get a very different result on your leaderboard compared to your validation set, but in the real world you won't know until you put it in production and send your company bankrupt and lose Your job, so you might want to think carefully about your validation set in that case, so here we're going to use, transform type classification, it's basically the same as transform type dot pixel. But if you think about it, we with a pixel version. If we rotate a little bit, then we probably want to, like average, the pixels in between the two but the classification. Obviously we we don't, we use nearest neighbor, so the slight difference there also for classification. You know lighting, doesn't kick in normalization to the dependent variable. Okay, they're already square images, so we don't have to do any cropping.

So here you can see different versions of the Augmented. You know move around a bit when they're rotating a bit and so forth. Yeah. I get a lot of questions kind of like during our study group and stuff about like how do I debug things and fix things that aren't working and like a

I

18. 01:38:00

- **“Start with a Conv layer and pass it to an RNN” question and research**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Never have a great answer other than like every time I fix a problem is because of stuff like this, that I do all the time you know I just always print out everything as I go, and then the one thing that I screw up always turns out To be the one thing that I forgot to check along the way so yeah, the more of this kind of thing you can do the better if you're, not looking at all of your intermediate results, I mean I have troubles okay, so, given that we want something That knows what cars look like: we probably want to start with a pre trained, imagenet network, so we're going to start with ResNet 34 and so with confident builder. We can grab our resin at 34 and we can add a custom head, and so the custom head is going to be something that up samples, a bunch of plants and we're going to do things really done for now, which is we're just going to do a Commons pose to addy batch norm, value, okay, and so here's like this is what I'm saying you could any of you could have built this without looking at any of this, or at least like you, have the information from previous classes. There's nothing new at all. Okay and so at the very end, we have a single filter - okay and now that's going to give us something which is batch size by 1 by 128 by 128. But we want something which is batch sized by 128 by 128. So we have to remove that unit axis, so I've got a lambda layer here.

Lambda layers are incredibly helpful right because without the lambda layer here, which is simply removing that unit axis by just indexing into it at 0, without the lambda layer, I would have to have created a custom class with a custom forward method and so forth. But by creating a lambda layer that does like the one custom bit, I cannotice chocolate from the sequential and so that just makes life easier, so the pytorch people are kind of snooty about this approach. Lambda is actually something that's part of the first day. I library not part of the pad torch library and, like literally people on the pad watch discussion board like yes, we could give people this. Yes, it is only a single line of code, but they never like encourage them to use sequential too often so there you go okay, so so this is that custom head right, so we're gon na have a rest at 34 that goes down sample and then a Really simple custom here that very quickly up samples, and that hopefully, will do something and we're going to use accuracy with a threshold of 0.5 to print out metrics, and so after a few a pops we've got 96 percent accurate. Okay, so is that good is 96 percent, accurate, good and hopefully the answer to your question. That question is, it depends, what's it for right, and the answer is qivana wanted this, because they wanted to be able to take their car images and cut them out and paste them on.

You know exotic monte-carlo backgrounds or whatever that's multicolor the place, the simulation. So to do that you need a really good mask, but you don't want to like leave the rearview mirrors behind or, like you know, kind of, have one wheel missing or include a little bit of background or something that would look stupid. So you would beat something very good, so only having 96 percent of the pixels correct doesn't sound great, but we won't really know until we look at it. So let's look at it. So there's the correct version that we want to cut out. That's the 96 % expert version, okay, so like. Where do you look at it? You guys oh yeah,

getting 90 %, 96 % of the pixels. Accurate is actually easy because, like all the outside bits, not care at all the inside bit is car and really really interesting. Bit is the edge okay. So we need to do better, so, let's unfreeze, because what we've done so far is trained the customer here. Okay, let's do more, and so after a bit more, we've got 99.1 percent. Okay, so is that good? I don't know, let's take a look, and so actually no, it's totally missed the rearview vision mirror here and

19. 01:42:40

- **The 100-layers Tiramisu: Fully Convolutional Densenets, for Image Segmentation (Lesson 13 cont.)**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

Missed a lot of it here and it's clearly got an edge wrong here, and these things are totally going to matter when we try to cut it out. So it's still not good enough. So let's try upscaling and the nice thing is that when we upscale to 512 by 512 make sure you decrease the batch size because you'll run out of memory, you know here's the true ones, it's quite a lot more. This is all identical. It's quite a lot! More information Everett go on, so our accuracy increases to 99.4 % and things keep getting better, but we've still got quite a few little black blocky bits. So let's go to 124 by 124 down to batch size of four. This is pretty high res now and train a bit more. Ninety nine point, six ninety-nine point eight, and so now, if we look at the masks, they're actually looking, not bad, okay, that's looking pretty good right. So can we do better, and the answer is yes, we can so we're moving from the carbonyl nut book to the carbonyl unit. Notebook now and the unit Network is quite magnificent right. You see with that previous approach. Our pre-trained imagenet network was being squished down. All the way down to seven by seven and then expand it out all the way back up to you know. Well, it's two to four go to seven by seven, but one or two four is going quite a bit bigger and then expanded out again or this way, which means it has to somehow store all the information about the much bigger version in the small version right And actually most of the information about the bigger version was really in the original picture anyway, so it doesn't seem like a great approach, this squishing and I'm squishing.

So the unit idea comes from this fantastic paper where, like it was literally invented in this, you know very domain-specific area of biomedical image segmentation, but in fact basically, every cattle winner in in anything even vaguely related to segmentation has ended up using unit as one of These things that, like everybody in Carroll, knows, is the best practice, but in more of an academic circles like even now, this has been around for a couple of years. At least a lot of people still don't realize. It's like this is by far the best approach and here's the basic idea: here's the downward path right, where we basically start start at five 72 by 5, 3. 2. In this case, and then kind of half the grid size, half the grid, size, half the grid, size, half the grid, size right and then here's the upward path where we double the grid size, double double double double. No, but the thing that we also do is we take. You know at every point where we've half the grid size, we actually copy those activations over to the upward path and and concatenate the together, and so you can see here these red blobs of maps, polling operations, the green blobs are upward sampling and then these gray Bits here are copying right, and so we copy and can cat so basically, in other words, the input image after a couple of poems is copied over to the output concatenated together, and so now we get to use all of the informations gone through all the down And all the app plus also a slightly modified version

of the input, pixels right and a slightly modified version of one thing down from the input pixels because they came out through yes right.

So we have like all of the richness of going all the way down and up, but also like a slightly less coarse version and a slightly less cost version, and then this really kind of simple version and they can all be combined together. Okay and so that's unit, it's such a cool idea, so here we are in the in the carvanha unit notebook all this is the same code as before and at the start I've got a simple up sample version, just to kind of show you again the the Non unit version this time, I'm going to add in something called the dice metric dice is very similar, as you see to jacquard or I over you, it's just a minor difference. It's basically intersection over Union with a minor tweak and the reason we're going to use dice is that's the metric that the caracal competition used and it's kind of it's a little bit harder to get a high score than a high accuracy. Because it's really looking at. Like what the overlap of the correct pixels are with with your pixels, but it's pretty similar so in the Carroll competition people that we're doing okay, we're getting about ninety nine point: six dice and the winners for about nine nine point: seven days. So here's our standard up sample this is all as before, and so now we can check our dice metric, and so you can see on dice metric we're getting like nine six, eight at 128 by 128, and so that's not great okay. So so, let's try unit and I'm calling it unit ish because that's per usual, I'm creating my own someone hacky version right kind of trying to keep things as similar to what you're used to as possible and doing things that I think, makes sense.

And so there should be plenty of opportunity for you to at least make this more authentically unit by looking at the exact kind of grid sizes and like see how here, the size is going down a little bit so they're, obviously not adding any padding and then They're doing here, they've got some cropping going on there's a few differences right, but one of the things is because I want to take advantage of transfer learning. That means I can't quite use unit. So here's another big opportunity is what, if you create the unit down path and then add a classifier on the end and then train that on imagenet and you've, now got an image net trained classifier, which is specifically designed to be a good backbone for unit right And then you should be able to now come back and get pretty close to winning this old competition is actually not that old. It's fairly recent competition, because you know that pre train network didn't exist before. But if you think about like what Yolo v3 did it's basically that right, they create a darknet, they pre trained it on image net and then they used it as the basis for their bounding boxes. So again, this kind of idea of free training - things which are designed not just for classification but to find for other things, is just something that nobody's nobody's done yet and let's weave.

But as we've shown, you know, you can train image net for 25 bucks in three hours now so and if people in the community are interested in doing this, you know hopefully I'll have credits. I can help you with as well. So if you do, you know the work to get it set up and give me a script. I can probably run it for you so for now, though, we don't have that so we're going to use resonate so so we're basically going to start with this. Let's see with get base and so base is our base network, and that was defined back up for this first section right, so get base is going to be something that calls whatever this is, and this is resna 34, so we're going to grab our ResNet 34 And cut model is the first thing that our confident builder does. It basically removes everything from the adaptive pulling onwards, and so that gives us back the backbone of resna, 34. Okay, so get base is going to give us okay and then we're going to take that rest at 30 for backbone and turn it into a. I call it a unit 34. So what that's going to do is it's going to save that ResNet that we passed in and then we're going to use a forward hook just like before to save the results at the second fourth, fifth and sixth blocks, which, as before, is the basically before each Straight to convolution, then we're going to create a bunch

of these things. We're calling unit blocks and the unit block basically says so. These unit blocks are these things.

These are unit blocks, so the the unit block tells us. You know we have to tell it how many things are coming from the from the kind of previous layer that we're up sampling how many are coming across and then how many do we want to come out right, and so the amount coming across is entirely defined By whatever the base network was right, it's like whatever, whatever the downward path was, we need that many layers, and so this is a little bit awkward. Actually, one of our master students here kerim, has actually created something called dynamic unit that you'll find in fastai unit dynamic unit, and it actually calculates this all for you and automatically creates the whole unit from your base model. It's got some minor quirks still that I want to fix by the time the videos out, it'll definitely be working, and I will at least have a notebook showing how to use it and possibly add additional video. But for now you know you'll just have to go through and do it yourself. You can easily see it just by once, you've got a reson it you can just go. You know just type in its name and it'll print out all the layers, and you can see how big, how many activations there are in each block, or you can even have a printed out for you for each for each block automatically anyway. I just did this manually, and so the unit block is works like this, so you said: okay right, this penny coming up from the previous layer.

I've got this penny coming across this X, I'm using across across from the the downward path. This is the amount I want coming out now. What I do is they then say: okay, we're going to create a certain amount of convolutions from the upward path in a certain amount from the cross path, and so I'm going to be concatenated them together. So, let's divide the number we want out by two right and so we're going to have our cross convolution, take our cross path and create number out /. And then the upward path is going to be a common transpose to D right because we want to increase up sample and again here, we've got the number n divided by two and then at the end I just concatenate those together alright, so I've got an upward Sample I've got a cross convolution, I can catenate the two together yeah and so that's all a unit block is, and so that's actually a pretty easy module to create, and so then, in my forward path, I need to pass to the forward of the of the Unit block the upward path and the cross plus, so the upward path is just wherever I'm up to so far right, but then the cross path is whatever the value is of whatever the activations are that I stored on the way down right. So as I come up, it's the last set of saved features that I need first and in as I gradually keep going up further and further and further.

Eventually it's the first set of features. Okay, and so there are some more tricks, we can do to make this a little bit better, but this is this is a good stuff right. So if we try this, so the simple up, sampling approach, looked horrible right and had a dice of nine six. Eight now you net with everything else identical except we've been out got these unit blocks, has a dice of nine eight five right, sir. That's like we've kind of halved the error with with everything else, exactly the same and more the point, you can look at it. This is actually looking somewhat car-like compared to our non unit equivalent, which is just a block now, because you know try to do this through down and up paths. Just it's just asking too much. You know where else when we actually provide the downward path pixels at every point, it can actually start to create something karush. So at the end of that, we'll go dot close to again remove those s. Fs features taking up GPU memory, go to a smaller batch size, a higher size, and you can see the dice coefficients really going up. This is just so notice here. I'm learning I'm loading in right, the 128 by 128 version of the network. Okay. So we're doing this progressive resizing trick again, so that gets us 99 3 and then unfreeze to get to 99 4, and you can see it's now. Looking

pretty good.

Okay go down to a batch size of 4, so 102 for load-in. What we just did with the 512 Texas, 299 5 unfreeze takes us to 99, we'll call that 99 six, five nine nine and, as you can see, that actually looks good right in accuracy terms. Ninety-Nine point eight two, you know you can see. This is looking like something you could just about used to cut out. I think, too, you know at this point there's a couple of minor tweaks. We can do to get up to ninety-nine point seven, but really the key thing then I think, is just maybe to do a you know a few bit of smoothing, maybe or a little bit of post-processing. You can go and have a look at the carvanha winners. Blogs and see some of these tricks but, as I say, the difference between where we are at ninety nine point: six and what the winners got of

20. 01:58:00

- **Building and training the Tiramisu model**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

99.7, you know, is it's not heaps and so really that just the you net on its own pretty much pretty much solves our problem. Okay, so that's it so. The last thing I wanted to mention is now to come all the way back to bounding boxes, because you might remember, I said out, our bounding box model was still not doing very well on small objects. So hopefully you might be able to guess where I'm going to go with this, which is that for the bounding box model, remember how we we had at different grid cells. We spat out outputs of their model and it was those earlier ones with the small grits sizes that weren't very good, for how do we fix it? You net it right. Let's have an upward path with cross connections right and so then we're just going to do a unit and then spit them out of that. The now does those finer grid cells have all of the information of that path and that path and that path and that path for leverage. Now, of course, this is deep learning, so that means you can't write a paper saying we just used you net for bounding boxes. You have to invent a new word, so this is called feature: pyramid networks or fbms, okay and like that literally the paper. This is part of the retina net paper, which is actually a it's used in the retina net paper. It was you, it was created an earlier paper, specifically of Olympians and like if memory serves correctly, they did briefly cite the unit paper, but they kind of made it sound like it was this vaguely slightly connected thing that maybe some people could consider slightly useful, but It really F P ends as units okay.

I don't have an implementation of it to show you, but you know it'll be a fun thing, maybe for some of us to try and some of us have already some I haven't yet, but I know some of the students have been trying so to get it Working well on the forums so yeah interesting thing to try. So I think a couple of couple of things to look at after this class, as well as the other things I mentioned, would be playing around with FP ends and also maybe trying caroms dynamic unit. They would both be interesting things to look at all right, so so you guys have all been through fourteen lessons of me talking at you now. So I'm sorry about that. Thanks for putting up with me, you know, I think it's it. It's you're gon na find it hard to find people who actually, as know them as much about training, neural networks and practice, as you do, it'll be really easy for you to overestimate. How capable all these other people are an underestimate, how poor you are, and so, like the main thing I'd say is like please practice, please just because you don't have this constant thing, getting you to come back here every

Monday night. Now it's very easy to kind of lose that momentum so find ways to keep it. You know you know, organize a study group, you know, or a book a reading group or get together some friends and work on a project or you know, do something more than just deciding.

I want to keep working on X like it's gon na need to involve problem unless you're the kind of person who's super motivated, and you know that whenever you decide to do something, it happens. That's not me right. It's like, I know something to happen. I have to like say yes, David in October. I will absolutely teach that course, and then it's like okay, if it actually writes a material, let's see only way, I can get stuff to happen. So we've got a great community there on the forums. If people have ideas for ways to make it better, please tell me you know if you think you can help with you know, if you want to create some new forum or moderated in some different way or whatever it is. Let me know right: you can always PM me and there's a lot of projects going on through github as well lots of stuff so yeah. I hope to see you all back here at something else and thanks so much for joining me on this journey. [Applause,]

21. 02:02:50

- **ENet and LINKNet models: better than the Tiramisu ?**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)

22. 02:04:00

- **Part 2: conclusion and next steps**

(autogenerated subtitles follow, may contain gibberish/bad format - please proofread to improve - remove this note once proofread)