Fastai Lesson 8a

Infrastructure
Broadcasting

Next week:
Sat. 7/13 Lesson 8b

TWiML Fastai Meetup
Sat. 06 Jul 2019
Joseph Catanzarite

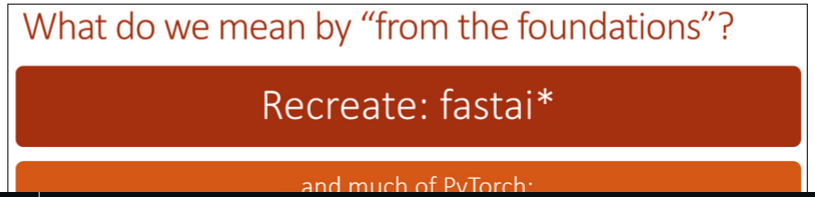
Part 2: Deep Learning from the Foundations

Welcome to *Part 2: Deep Learning from the Foundations*, which shows how to build a state of the art deep learning model from scratch. It takes you all the way from the foundations of implementing matrix multiplication and back-propogation, through to high performance mixed-precision training, to the latest neural network architectures and learning techniques, and everything in between. It covers many of the most important academic papers that form the foundations of modern deep learning, using "*code-first*" teaching, where each method is implemented from scratch in python and explained in detail (in the process, we'll discuss many important software engineering techniques too). Before starting this part, you need to have completed Part 1: Practical Deep Learning for Coders.

The first five lessons use Python, PyTorch, and the fastai library; the last two lessons use Swift for TensorFlow, and are co-taught with Chris Lattner, the original creator of Swift, clang, and LLVM.

The purpose of *Deep Learning from the Foundations* is, in some ways, the opposite of part 1. This time, we're not learning practical things that we will use right away, but are learning foundations that we can build on. This is particularly important nowadays because this field is moving so fast. In this new course, we will learn to implement a lot of things that are inside the fastai and PyTorch libraries. In fact, we'll be reimplementing a significant subset of the fastai library! Along the way, we will practice implementing papers, which is an important skill to master when making state of the art models.

In the remainder of this post I'll provide a quick summary of some of the topics you can expect to cover in this course—if this sounds interesting, then click on lesson 8 in the "Part 2" section of the sidebar over on the left. And if you have any questions along the way (or just want to chat with other students) there's a very active forum for the course.



Overview

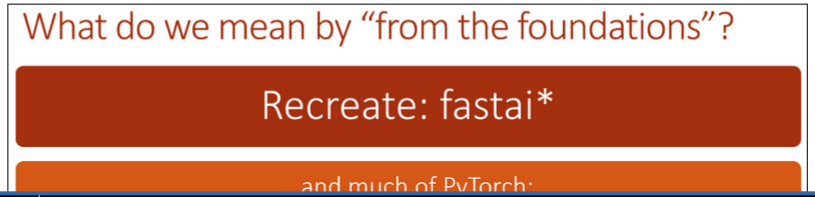
Part 2: Deep Learning from the Foundations

Welcome to *Part 2: Deep Learning from the Foundations*, which shows how to build a state of the art deep learning model from scratch. It takes you all the way from the foundations of implementing matrix multiplication and back-propogation, through to high performance mixed-precision training, to the latest neural network architectures and learning techniques, and everything in between. It covers many of the most important academic papers that form the foundations of modern deep learning, using "*code-first*" teaching, where each method is implemented from scratch in python and explained in detail (in the process, we'll discuss many important software engineering techniques too). Before starting this part, you need to have completed Part 1: Practical Deep Learning for Coders.

The first five lessons use Python, PyTorch, and the fastai library; the last two lessons use Swift for TensorFlow, and are co-taught with Chris Lattner, the original creator of Swift, clang, and LLVM.

The purpose of *Deep Learning from the Foundations* is, in some ways, the opposite of part 1. This time, we're not learning practical things that we will use right away, but are learning foundations that we can build on. This is particularly important nowadays because this field is moving so fast. In this new course, we will learn to implement a lot of things that are inside the fastai and PyTorch libraries. In fact, we'll be reimplementing a significant subset of the fastai library! Along the way, we will practice implementing papers, which is an important skill to master when making state of the art models.

In the remainder of this post I'll provide a quick summary of some of the topics you can expect to cover in this course—if this sounds interesting, then click on lesson 8 in the "Part 2" section of the sidebar over on the left. And if you have any questions along the way (or just want to chat with other students) there's a very active forum for the course.



Run the notebooks on the cloud

Or, run the notebooks on your own computer

How to install fastai v1 on Windows 10

Medium article by Pierre Guillou

https://medium.com/@pierre_guillou/how-to-install-fastai-v1-on-windows-10-ca1bc370dce4

Installation Guide on fastai/README

https://github.com/fastai/fastai/blob/master/README.md#conda-install

Fast AI Video Viewer - Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://course.fast.ai/videos/?lesson=8

SFFCU Google fastai install colab Colab | fast.ai course v3 FastaiForum Foundations NLP-TF 2019Part2 TWiML&AI CVPR NLP-fastai RXRX.AI SIIM-ACR doc.ai CS231 CS224N

esson 8 (2019) - Deep Learning from the Foundations

Watch later Shar

What do we mean by "from the foundations"?

Recreate: fastai*

...and much of PyTorch:
matrix multiply, torch.nn, torch.optim, Dataset, DataLoader

Python	Python stdlib	Non-data science modules	PyTorch array creation, RNG, indexer	fastai.datasets	matplotlib

* but we'll make it *even better*!

MORE VIDEOS

Search transcript

2:06:40

esson 8 (2019) - Deep Learning from the Foundations

Watch later | Shar

But why?...

Really experiment

Understand it by creating it

Tweak everything

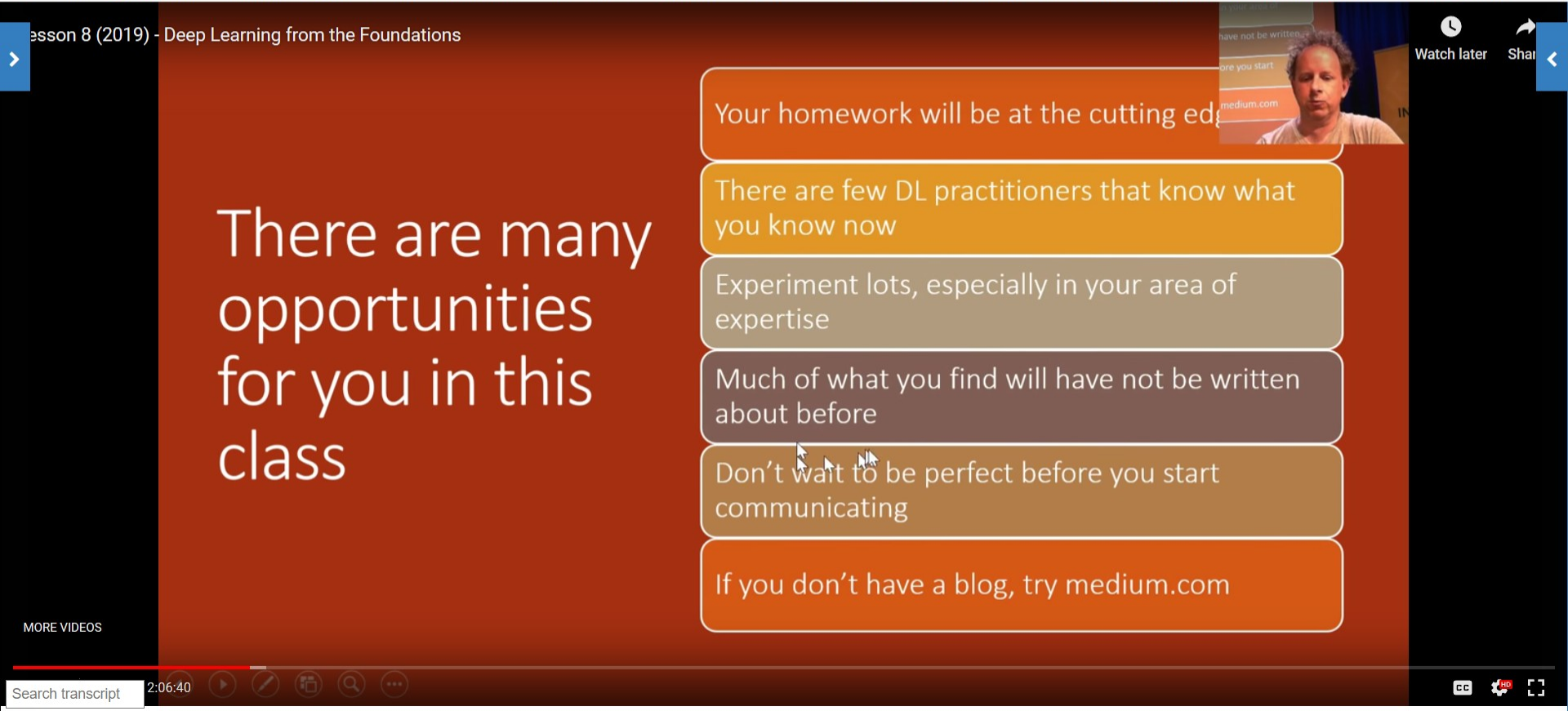
Contribute

Correlate papers with code

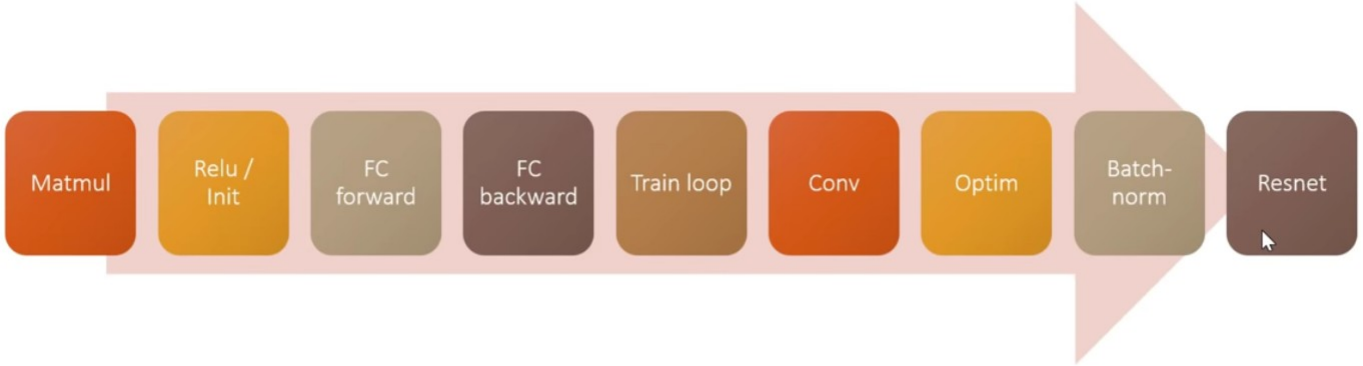
MORE VIDEOS

Search transcript

2:06:40


esson 8 (2019) - Deep Learning from the Foundations
Watch later
There are many opportunities for you in this class
Your homework will be at the cutting edg
There are few DL practitioners that know what you know now
Experiment lots, especially in your area of expertise
Much of what you find will have not be written about before
Don't wait to be perfect before you start communicating
If you don't have a blog, try medium.com
MORE VIDEOS
Search transcript
2:06:40

Steps to a basic modern CNN model



Road map for the first few lessons

It's time to start reading papers

Theorem 4.1. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Our Theorem 4.1 implies when the data features are sparse and bounded gradients, the summation term can be much smaller than its upper bound $\sum_{i=1}^{d} \|g_{1:T,i}\|_2 << dG_\infty\sqrt{T}$ and $\sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} << dG_\infty\sqrt{T}$, in particular if the class of function and data features are in the form of section 1.2 in (Duchi et al., 2011). Their results for the expected value $\mathbb{E}[\sum_{i=1}^{d} \|g_{1:T,i}\|_2]$ also apply to Adam. In particular, the adaptive method, such as Adam and Adagrad, can achieve $O(\log d\sqrt{T})$, an improvement over $O(\sqrt{dT})$ for the non-adaptive method. Decaying $\beta_{1,t}$ towards zero is important in our theoretical analysis and also matches previous empirical findings, e.g. (Sutskever et al., 2013) suggests reducing the momentum coefficient in the end of training can improve convergence.

Finally, we can show the average regret of Adam converges,

Corollary 4.2. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$$

- Even familiar stuff looks complex in a paper!
- Papers are important for deep learning beyond the basics, but hard to read
- Google for a blog post describing the paper
- Learn to pronounce Greek letters

sebastianruder.com/optimizin

Adam

Adaptive Moment Estimation (Adam) [15] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

Familiarize with Math symbols and Greek alphabet

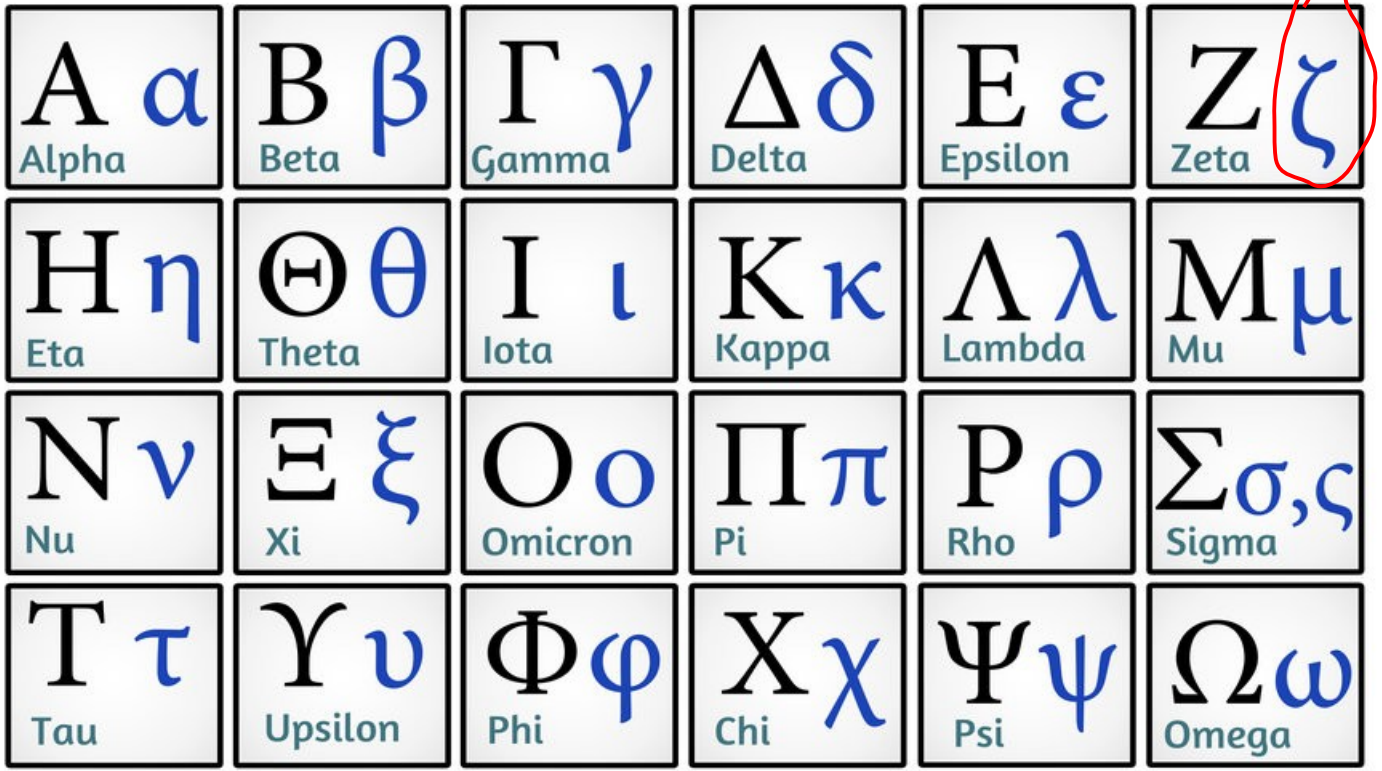
https://en.wikipedia.org/wiki/List_of_mathematical_symbols

Symbols based on equality [edit]

Symbol in HTML	Symbol in TeX	Name / Read as / Category	Explanation	Examples
=	=	equality / is equal to; equals / everywhere	$x = y$ means x and y represent the same thing or value.	$2 = 2$ $1 + 1 = 2$ $36 - 5 = 31$
≠	≠ \ne	inequality / is not equal to; does not equal / everywhere	$x \neq y$ means that x and y do not represent the same thing or value. (*The forms !=, /= or <> are generally used in programming languages where ease of typing and use of ASCII text is preferred.*)	$2 + 2 \neq 5$ $36 - 5 \neq 30$
≈	≈ \approx	approximately equal / is approximately equal to / everywhere	$x \approx y$ means x is approximately equal to y. *This may also be written* ≃, ≅, ~, ♎ (Libra Symbol), *or* ≒.	$\pi \approx 3.14159$
		isomorphism / is isomorphic to	$G \approx H$ means that group G is isomorphic (structurally identical) to group H.	$Q_8 / C_2 \approx V$

https://en.wikipedia.org/wiki/List_of_mathematical_symbols

Greek Alphabet and Symbols



Detexify LaTeX handwritten symbol recognition - Mozilla Firefox

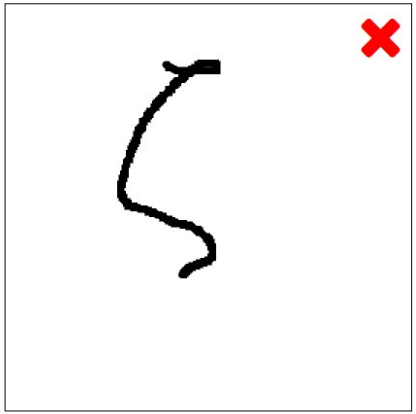
File Edit View History Bookmarks Tools Help

detexify.kirelabs.org/classify.html

SFFCU Google fastai install1 fastai install2 colab Colab | fast.ai course v3 FastaiForum Foundations NLP-TF 2019Part2 TWiML&AI CVPR NLP-fastai

Detexify

classify | symbols



Score: 0.08138207974610712
\zeta
mathmode

Score: 0.09904512225282196
\usepackage{ upgreek }
\upzeta
mathmode

Score: 0.10159323421165987
\xi
mathmode

Score: 0.10659579735043749
\varsigma
mathmode

Score: 0.10665310334005068
\langle
mathmode

Want a Mac app?

Lucky you. The Mac app is finally stable enough. See how it works on Vimeo. Download the latest version here.

Restriction: In addition to the LaTeX command the unlicensed version will copy a

http://detexify.kirelabs.org/classify.html

Exporting jupyter notebooks as Python modules

First notebook 00_exports.ipynb:

```
!python notebook2script.py 00_exports.ipynb
```

Converts 00_exports.ipynb to exp\nb_00.py, a python module

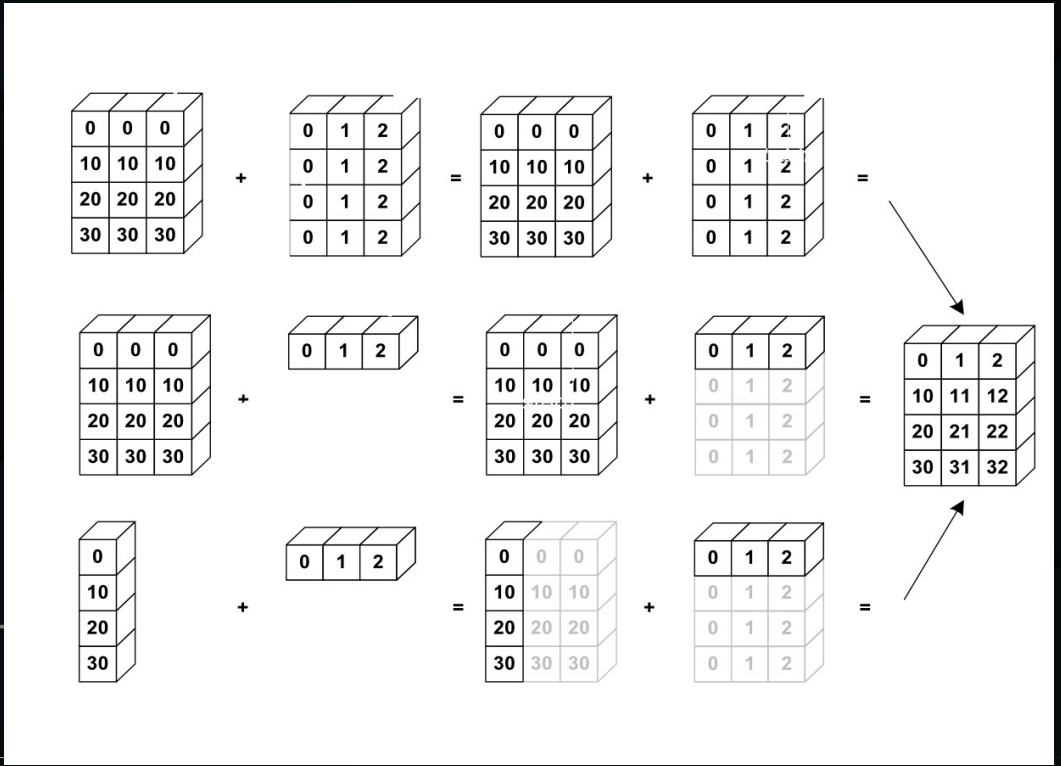
Next notebook 01_matmul.ipynb:

```
from exp.nb_00 import *
```

Each notebook imports material from the previous notebook, in the form of a file exp\nb_xx.py

Broadcasting

Extends array operation syntax via implicit rules



https://mathematica.stackexchange.com/questions/99171/how-to-implement-the-general-array-broadcasting-method-from-numpy

Broadcasting Rules

Rule 1: If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded with ones on its leading (left) side.

Rule 2: If the shape of the two arrays does not match in any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.

Rule 3: If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

https://jakevdp.github.io/PythonDataScienceHandbook/02.05-computation-on-arrays-broadcasting.html

`00_exports.ipynb` and `01_matmul.ipynb`

Play with them until you understand as much as you can!

Also, have a look inside

`notebook2script.py`

to see what it’s doing

Run through the notebooks

For next time

Set up your environment to run notebooks!

Study the second half of Lesson 8, on forward and backward propagation

Work through the associated notebook `02_fully_connected.ipynb`

Read section 2.2 in the “Delving Deep Into Rectifiers” paper

`https://arxiv.org/abs/1502.01852`