openSAP
Enterprise Deep Learning with TensorFlow

Week 05 Unit 01

00:00:08 Welcome to Week 5 of the openSAP course "Enterprise Deep Learning with TensorFlow". The topic for this week is "Industry Applications of Deep Learning".

00:00:17 My name is Oliver Kroneisen. I am the vice president of Financials Operations Development in LoB Finance.

00:00:24 And in this unit I will give you an overview on machine learning from an application point of view. Of course, inspired a bit by my finance background.

00:00:35 You will learn in this unit which application aspects are important to select for valid machine learning cases. I want to apply the paradigm of feasibility, desirability, and viability

00:00:50 to validate the machine learning use cases you have in mind. And I want to talk about which criteria you should check in order to make a "go" or "no-go" decision

00:01:00 if you want to go with a specific machine learning case. Let's have a look at the magic intersection of feasibility, viability, and desirability.

00:01:14 The specific machine learning use cases you have in mind can be classified according to those criteria. In feasibility, we check whether the use case you have in mind can technically be solved by machine learning.

00:01:30 So that includes of aspects of: "Will you be able to build it?" In desirability, we will also look to the customer point of view.

00:01:39 So will the customer want a solution based on machine learning? And is it also a worthwhile endeavor for customers and the software producers to embark on this use case?

00:01:53 In viability, we will also check whether there is a valid use case and a business case for customers to adopt the machine learning application.

00:02:03 And will the customer also be interested in paying for it? Now let's look in more depth at the visibility side,

00:02:11 which is usually the first thing you want to check. First of all, if you want to check the feasibility of machine learning,

00:02:20 you have to check the data sources available to train your machine learning algorithm, because only when you have data to train the algorithm can it derive proper results.

00:02:34 The usual starting point is structured data, such as database tables. And you would ask yourself whether you know the relevant fields that you want to include

00:02:44 in the machine learning algorithm. However, in many cases there's also unstructured data you want to consider.

00:02:51 Here it's important that you have an idea how you can convert this unstructured data, for example, images of certain documents, into structured data.

00:03:02 What's also important is: "Do you have enough historical decision data that humans based their decisions on in order to train your algorithm?"

00:03:12 Because that's like a baby that you want to educate with this data to produce predictions and good results.

00:03:20 And what's very important is: "Are customers willing and allowed to share this data with you?" And: "How long will it take until you get this data?"

00:03:31 That's a topic that's sometimes underestimated, and you have to consider this in your project plan.

00:03:38 Now in the second step, you would look at: "How do humans process the problem in real life?" And you would ask yourself whether the users can readily derive the solutions with the data that you have available.

00:03:56 What could be critical is if users in real life have a certain gut feeling to base their decisions upon, but there may be no evidence in recorded data.

00:04:07 Or they might rely on their social network and personal connections to make decisions. It may be very hard to apply machine learning here,

00:04:17 and you may think about or rethink whether you want to add some more data to your database and selected data fields. Okay, let's look at the desirability of your machine learning case.

00:04:30 Of course only comparative advantages are convincing customers to apply machine learning. So there are two possibilities to consider here.

00:04:39 The first one is: "Can your machine learning algorithm solve a problem that otherwise could not be handled or solved by humans at all?"

00:04:48 This would be a strategic differentiator and the best possible case for your machine learning use case. Or: "Could your machine learning use case solve a problem that otherwise would be done by a huge amount of human workforce?"

00:05:05 In that case, you may have a cost differentiator. Of course, both things could also come together.

00:05:11 If it's about cost differentiation, the crucial question is: "How many people do this work at the customer?"

00:05:19 Just to give you some examples: Many customers in finance have large financial shared service centers,

00:05:25 with hundreds, sometimes thousands, of people working in them. When we build the SAP Cash Application powered by SAP Leonardo,

00:05:33 we are addressing a use case of applying incoming cash to issued invoices automatically. That is a topic that hundreds of people are working on in different shared service centers for different customers.

00:05:48 So there is a huge cost efficiency gain to be envisioned for those customers. In contrast, if you were to look at a machine learning algorithm,

00:05:58 let's say to optimize something which is only done by a few specialized people in a Treasury department, which could be a few as 5–10 people,

00:06:09 the cost effect would, of course, be much smaller. Another aspect of desirability, of course, is to think about limitations that you may also face.

00:06:22 These limitations come from three different areas. One is a legal limitation that you could face.

00:06:30 Just to give an example from finance. For example, customers may need to explain the result of the machine learning algorithm to their customers,

00:06:40 or perhaps even the court. Think of deriving a credit limit that a customer would question.

00:06:48 A second constraint could also be ethical reasons. So it's currently discussed in other domains like:

00:06:56 "Is it okay if a machine learning algorithm decides on a matter of life or death?" Of course, these use cases are very rare in application development.

00:07:05 And finally, there's the question of whether the machine learning algorithm will also be accepted by the customer or by the users applying it.

00:07:15 For example, if the machine learning algorithm made irreversible decisions that later on prove to be wrong,

00:07:25 would people accept it? So one way out of that is to think about use cases where you can undo your decisions.

00:07:33 Like in the case of SAP Cash Application, if we assign the cash to the wrong invoice of the very same customer,

00:07:42 we could later on change it if the customer calls Now, let's come to the topic of viability,

00:07:49 so the business side of machine learning use cases. The question is: "Is the customer willing to pay to use a machine learning algorithm?

00:08:00 Is he also willing to accept the TCO coming with First of all, the important thing to understand is that customers may be triggered by the term "machine learning"

00:08:12 but actually they are always looking for better automation of their business. So that's the key driver, and not so much which technical means we have to serve these aims.

00:08:24 So basic questions to consider here are: "Does your machine learning algorithm solve a use case that cannot be solved in an easier way?"

00:08:33 So sometimes classical deterministic rules, multilinear linear regressions from mathematical modules, can provide a similar degree of automation.

00:08:43 On the other side, even if classical means applicable, machine learning has the big advantage that it can adapt to changing data

00:08:53 and it can learn over time. So that may also reduce the TCO of a machine learning algorithm compared to classical applications.

00:09:02 Quite often – and that's my experience I want to share with you – the combination of these different tools is the most successful way to go ahead.

00:09:11 Combining, for example, an SAP Cash Application, classical rules, with deep learning from neural networks.

00:09:20 Okay. So let's think a bit more about what automation of use cases really means for the customer,

00:09:29 and how we address it in a product. Customers, of course, think of their business as being composed of end-to-end processes,

00:09:38 and they want to automate that. So if your machine learning use case would just aim at a very isolated problem,

00:09:46 customers may say: "Okay, it's great to have it, but it's nice as a free-of-charge thing." And it may not even be seen as a real valid product,

00:09:55 and they might not be willing to pay for it. On the other side, without customer demand

00:10:01 you would not be able to have a sales force to bring your product to the market and explain the advantages of it to customers.

00:10:09 A good way out of this dilemma is to bundle machine learning with improvements in the business application to cover the full end-to-end process.

00:10:22 Usually, and that is what we successfully did in the case of SAP Cash Application between finance development and the Leonardo team,

00:10:32 is to have a joint team with experts from both ends – from the machine learning side and the application side.

00:10:38 So let's take a look now at what our end-to-end process looks like. I don't want to go through all of the details here on this slide,

00:10:50 but I also want to highlight that you can see several heads that mark the different places where machine learning is applied.

00:10:59 So the Cash Application part is clearly visible in the middle of the picture, but there are also other steps in the process chain where we are working on applying machine learning.

00:11:11 For example, to automate the execution of income and remittance advice. Or other options are to look into automating dispute cases,

00:11:24 or providing proposals on how to do cash collection. It's a whole chain of processes that we improve with our machine learning application.

00:11:34 And that may be something that you can also apply in your specific domain and think about how this whole bundle can serve the customer

00:11:44 and create the value that the customer is looking for. Yeah, these are my experiences so far from the finance side.

00:11:53 I would like to thank you for joining me in this unit. And I will also highlight that in the next unit

00:12:00 you will learn about machine learning in the area of customer service. I hope you enjoy this course and will enjoy the rest of the courses.

00:12:10 Wishing you a good learning experience.

Week 05 Unit 02

00:00:05 Welcome to Week 5 of the openSAP course Enterprise Deep Learning with TensorFlow. In the previous unit, you learned about Industry Applications of Deep Learning with Oliver.

00:00:17 This week, we will be getting into some of those industries, starting with Deep Learning in Customer Service.

00:00:23 My name is Georg Glantschnig, SVP for SAP Hybris Cloud for Customer and Chief Data Scientist, and I will go over the applications of deep learning in the context of customer service.

00:00:35 We have embedded deep learning models into our sales and service applications to support employees in their daily jobs.

00:00:42 The key findings in this presentation are based on projects we have worked on with 20 co-innovation customers. This unit will cover the business problem

00:00:54 and the current landscape that we aim to address with deep learning. Based on customer examples, we will go through the six key steps of the deep learning workflow,

00:01:04 as outlined on the slide. We will close this unit by connecting the dots and show how deep learning augments service agents

00:01:12 as part of SAP Hybris Service solutions in a real customer scenario. An excellent customer experience is a competitive advantage for any business.

00:01:25 This is the very reason many companies have made strategic investments into constantly elevating it for their customer base.

00:01:34 Today, consumers demand new ways to interact, and they are enabled by technologies that are improving at an exponential rate.

00:01:41 When it comes to buying and retail, people want to stay informed of the best deals, and they expect to be given good service regardless of the channel they are using

00:01:50 to purchase and engage their retailer of choice. Customer service is often the tipping point between saving or breaking a customer relationship.

00:02:03 Customers who encounter positive social customer care experiences are three times more likely to recommend a brand.

00:02:11 Because the human brain is 10 times more receptive for bad experiences, 95% of dissatisfied customers will more than likely tell others about their bad experience.

00:02:23 The customer service agent is expected to deliver a consistent excellent customer experience. But this is getting more and more difficult to accomplish

00:02:32 with the ever-increasing access that customers have to engage businesses. Customer agents face the challenges of a growing amount of tickets

00:02:41 and their incentive structure based on resolution time and resolution quality. Tickets either vary in complexity or can be redundant

00:02:51 and would require better tagging and categorization before an agent can even begin to respond.

00:02:58 The question we are trying to answer during this session is: Can deep learning help us to improve the quality of the incoming ticket stream for a service agent?

00:03:11 Now that we have set the stage with the current landscape of customer service, let's do a little exercise.

00:03:18 Bel Air is a fictional airline company. As with any business these days, everyone is expected to be active on social media.

00:03:26 Bel Air is no exception as it uses the usual platforms of Facebook, Instagram, Twitter, LinkedIn and Snapchat.

00:03:35 By looking at historical data, we will train a deep neural network based model from TensorFlow that will predict the service categories for new incoming Twitter messages.

00:03:50 Before we dive in, let's do a quick recap on the different types of machine learning areas. The two most-used types of machine learning algorithms are supervised learning

00:04:00 and unsupervised learning. Supervised learning is where we can use past examples of input-output pairs from experience to train a model.

00:04:11 There are other types such as reinforcement learning and recommender systems, but for this exercise, we will focus on supervised learning

00:04:18 by taking the example of historical customer tickets and applying deep learning to build a neural network model that will predict ticket categories.

00:04:29 If you look at the deep learning workflow, you will find that it is standardized from data collection and exploration,

00:04:37 data preparation including cleansing, de- duplicating, normalizing the data, before we apply feature engineering to create multiple models.

00:04:46 Then select the best-fitting model with the best performance, deploy in production, retrain the model, and do the whole cycle over again,

00:04:54 iteratively until there is diminishing return on value. The focus is to get a pretty good but not yet perfect model deployed

00:05:03 and then it will improve over time. As a first step in data collection and exploration,

00:05:12 we will take a closer look at Bel Air's historical sample data. A large airline receives on average 20k messages a day via their social channels.

00:05:23 For simplicity, we will use a sample data set the size of a day for this exercise. In practice, we would use a data set collected over months or even years.

00:05:35 From a machine learning perspective, the messages are the inputs to the neural network, and the ticket categories correspond to the labels that the network outputs.

00:05:45 We now proceed to analyze the quality of the data. One of the major challenges in applying machine learning in the enterprise

00:05:56 is going from raw imperfect data to clean and useful data for model training. For our exercise, we used a data wrangling tool called Trifacta.

00:06:05 Data cleansing is a very important step and there are different tools out there depending on what input data you have.

00:06:13 In our case, we have simple text, so our main challenges are data samples without labels or duplicates.

00:06:20 On further analyzing our file, we find out there are no duplicate messages. However, there are 0.1%, or roughly 13, data samples without label which will be ignored

00:06:31 for training our model. What about model selection and defining the model architecture?

00:06:42 Traditionally, linear models like support vector machines, together with document feature representations like bag-of-words or "text frequency" weighting,

00:06:51 have been popular for text classification. However, a disadvantage of such methods is that the word ordering is lost.

00:06:59 Moreover, these methods represent each word as an independent discrete feature and fail to capture semantic similarities between words.

00:07:09 Distributed representations, such as Word2Vec that uses a neural network, can capture the semantics of words in a sentence

00:07:19 much more effectively than bag-of-word approaches. For example, the words "love" and "like" would have similar vector representations.

00:07:28 To capture word order, a widely used approach is applying a convolutional neural network. However, such an approach demands storing a stacked representation of all word embeddings as a matrix,

00:07:42 which requires considerable memory when the word vocabulary is large. Specifically, in industry applications, the word vocabulary can be very large,

00:07:52 making the problem challenging. Also, word-based models cannot easily handle "out-of-vocabulary" words and spelling errors.

00:08:01 Instead, in our example, we are going to represent words as a sequence of characters and train a convolutional neural network over these character embeddings.

00:08:12 Such a character-level representation reduces the memory requirements significantly as the number of characters is much smaller than the number of words.

00:08:21 Moreover, this network architecture can easily handle noise and spelling mistakes in the text. We can see on these experimental results that the model achieves superior accuracy

00:08:32 compared to the other approaches across a range of standard benchmark data sets. We will look at the details of the character-level convolutional neural network in the subsequent slides.

00:08:45 What about model design? This is a high-level overview of the workflow of a service ticket classification at Bel Air.

00:08:54 Service tickets – which are essentially text documents – are the input to the model. The character-level CNN we discuss here can process these tickets

00:09:03 and output normalized probability scores for each ticket category. The sum of these probabilities across categories is equal to 1 for each ticket.

00:09:14 These categories for Bel Air are "Compliment", "Request", and "Complaint". The category with the highest score is the predicted class.

00:09:23 In our example, the first ticket is classified as "Compliment", the second as "Request", and the third as "Complaint".

00:09:35 Let's look at the model architecture in more detail. The model starts by encoding each character in the input data as an N-dimensional vector.

00:09:44 These vectors are used to encode input sequences into a matrix by concatenating the character embeddings.

00:09:52 In the matrix, each column corresponds to the embedding vector of a particular character. For example, the first column in the "Excellent" matrix on the top row represents the embedding vector for "E",

00:10:04 the second column for "X", and so on. The encoded input sentences are passed through several layers of convolutions to extract features.

00:10:18 The features extracted by the convolutional layers are fed to subsequent fully connected layers in the network,

00:10:25 all the way to the output layer with softmax activation. The number of neurons in the output layer depends on how many categories we have.

00:10:34 The output layer generates the probability for each class. Why did we choose a character-level model instead of a word-level model?

00:10:43 In addition to the good performance, the character-level model has the following compelling advantages.

00:10:49 In industry applications, we often have the requirement that the model must be able to continuously learn over time.

00:10:56 This also means that it has to deal with new words that appear in the vocabulary. The set of possible characters, however, is fixed.

00:11:04 Representing the input as a sequence of characters sidesteps the need for updating a new vocabulary every time when a new word appears.

00:11:14 Real-world data sets can be noisy and have a large vocabulary. Character-level models can naturally handle spelling errors in the input.

00:11:22 They also have a much smaller memory footprint as they only need to store one embedding per character instead of one embedding per word.

00:11:34 Once we have designed the model, the next part is training and testing the model using the prepared data set.

00:11:41 For this, we split the data into a training and test The idea is to evaluate the model performance on unseen data.

00:11:50 Here, we are using 80% of the data to train the model, and then use the 20% of the data to let the model predict the output and compare it to the actual label.

00:12:05 Building predictive applications in the real world is an iterative process, and the metrics that you choose to evaluate your machine learning algorithms are very important.

00:12:15 The choice of metrics influences how the performance of machine learning algorithms is measured and compared.

00:12:21 The estimated performance of a model tells how well it performs on unseen data. Since our Bel Air example is a classification type problem,

00:12:31 we will look at some common evaluation metrics for classification tasks. Let's start with accuracy – the most intuitive one.

00:12:41 It is simply the ratio of correctly predicted observations to the total number of observations. In our case, we have a test accuracy of 86.7%.

00:12:52 This is a very good result since it will take away about 5–6 hours of a workday for an agent just trying to understand the context and route tickets to the right department.

00:13:05 Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

00:13:14 Let's look at "Request" with a value of 0.9. This means, out of all the tickets for which the model predicted the category "Request",

00:13:24 90% are actually in this category. Recall is the ratio of correctly predicted positive observations

00:13:33 to all observations in the actual category. Again, let's stay with our category "Request" and a value of 0.86.

00:13:43 This means that out of all the tickets truly being a "Request", the model predicted 86% correct. F1 score is the harmonic mean of precision and recall which combines precision and recall

00:13:57 into a single number. What other insights do we get?

00:14:03 In this example, our model does a fairly good job of predicting "Requests" and "Compliments", but it's not that great yet for predicting "Complaints".

00:14:13 From a business perspective, we may want to try tweaking our model in the next iteration to catch more of these complaints and turn these unhappy customers into satisfied customers.

00:14:28 Now that we have trained and evaluated the model, it's time to take it for a test drive. In this example, I am going to use an internal API testing tool

00:14:38 to call the model which is already deployed as an but you can very well be using Postman or any other open source tools to test your model API.

00:14:48 The key aspect here is to test the API and analyze performance and get a sense before deploying it as part of an application process.

00:14:58 I enter a sample text into the text field... and then the tool displays a couple of helpful information items about your model,

00:15:09 like language detected, the label predicted, the confidence score, and many others. So let's conclude: API and model work as expected.

00:15:27 Let's put all these concepts we have just learned together. We have built a deep learning model, very like the example you just saw,

00:15:34 which is deployed and integrated to our SAP Hybris Cloud for Service application. It's time to see it in action.

00:15:43 We will see two different roles interacting with our application. Jack, a traveler with Bel Air,

00:15:49 and Jenny, a service agent at Bel Air using our SAP Hybris Service solution and our trained deep learning model.

00:16:00 Jack just took a flight with Bel Air, but due to the short connection, his luggage didn't arrive at the destination.

00:16:06 So he goes to the Twitter app on his phone and searches for Bel Air's Twitter page. Jack then tweets to Bel Air's customer service about his missing bags,

00:16:14 hoping to find some information on how to track his bags. Once the tweet is posted, it gets converted to a service ticket in SAP Hybris Cloud for Service

00:16:25 and shows up as a notification on the agent's home page. Jenny, the agent, sees the new tweet posted from Jack about his missing luggage

00:16:33 and clicks on the notification link that takes her to the ticket details view. As SAP Hybris Cloud for Service is activated with deep learning for Service Ticket Intelligence,

00:16:43 the incoming message goes through the deployed neural network model to surface key words giving it a mathematical representation

00:16:51 and ultimately predicting the service category. On clicking on the prediction icon,

00:16:56 the agent can see the confidence level of the prediction – in this case, 95% – and keywords that contributed to the prediction.

00:17:04 Satisfied by the prediction, the agent clicks on the thumbs-up sign, thus explicitly giving feedback to the model

00:17:12 as it saves him approximately 30 seconds in reading the Twitter message and manually assigning the service category.

00:17:20 Thank you for joining me in this unit. I hope that you found these concepts valuable

00:17:25 and that it's helped you get a better understanding of deep learning in Customer Service. You will learn more about other industries in the next unit

00:17:33 as it covers Machine Learning in Banking. That unit will be led by Dr. Eckner of JP Morgan Chase.

00:17:40 Enjoy the rest of the course.

Week 05 Unit 03

00:00:05 Hello, welcome. My name is Sinziana Eckner, and today I will talk about Machine Learning in Banking.

00:00:12 So, first a little bit about myself and my role. I am a data scientist at JPMorgan Chase, on the Digital Intelligence team,

00:00:22 and I lead the recommendation system effort. And a little bit about the digital intelligence:

00:00:28 We are a team of data scientists and our scope is personalizing Chase's digital domains, which include Chase's Web sites and mobile apps.

00:00:39 So our goal is personalizing the digital channels with machine learning, via machine learning models. And one thing that I would like to point out is that our area is within the consumer banking space,

00:00:59 consumer and retail banking. JPMorgan Chase is a large financial institution that is composed of several banks,

00:01:07 including the investment bank, the commercial bank, the private bank, and the consumer bank. Our focus is on consumer applications, so we do not have any financial-facing applications,

00:01:23 but really consumer-facing, like, retail banking and, for example, business banking, mortgages, and everything in the consumer space.

00:01:37 So a little bit about myself and my background: I've been on the Digital Intelligence team for about a year now.

00:01:45 I was in a previous team – also at JPMorgan Chase – for two years prior to this. I was doing modeling there for business banking applications,

00:01:55 and before that, I completed a PhD in mathematics at the Courant Institute at New York University. My research there was in probability theory, and I loved Courant – I had a lot of fun there,

00:02:05 and I have a lot of fun in my current role at Chase. I'm also a lead organizer for the NYC Women in Machine Learning and Data Science meetup group,

00:02:16 and our goal there is to form a community of women interested in machine learning topics in New York City. And I'm very proud of the community that we've built.

00:02:25 So now let me get into a little bit about the work of Digital Intelligence at Chase. So I'll describe a few digitally focused use cases.

00:02:44 One use case is experience personalization – so, everything that has to do with the customer experience. And one example of that is location-based background image.

00:02:56 Other applications are essentially how the site changes for you, how the tiles rearrange themselves, how the content rearranges itself – everything related to that.

00:03:11 Another use case is ad targeting on Chase-owned channels. Ad targeting is powered by our machine learning models

00:03:20 with the goal of personalizing the experience and providing the customer with the most relevant offer for them at a given time.

00:03:28 And another use case is paid media targeting, which is targeting on Web sites external to Chase.

00:03:37 And, similarly, we use data that our team processes and our models for paid media targeting. So, now I'll describe a few other applications which are not in the digital space,

00:03:54 but are really in the consumer banking space. So a few applications that our team has worked on are fraud modeling.

00:04:05 We've tested some models for fraud and found that our machine learning models can bring really good value there, both in terms of accuracy and in terms of speed of execution,

00:04:19 which is crucial to fraud. Another example is credit scoring.

00:04:25 And yet another example is image processing – we had a project of distilling insights from creatives with artificial intelligence

00:04:37 which was focused around which aspects of creatives appeal more to a consumer and which don't. And these insights are being used in the design process,

00:04:53 so improving the design process to better appeal to the consumer. So now I will address a question that I get asked frequently, which is:

00:05:03 Do you use machine learning in banking? And the answer is yes, otherwise I wouldn't be talking here, giving this course.

00:05:11 But there is a common misconception that in banking we rely on very simple models, simple tools like Excel, and we don't really use the power of machine learning.

00:05:21 And there are a few reasons for this. The industry has changed, so it is true that in the past, we relied on single-node processing,

00:05:29 and the tools that we used were much less complex. But now, we do all our computations in Spark and Hadoop, which enables distributed computation,

00:05:42 which internally enables fast prototyping and lets us take advantage of large-scale machine learning using Big Data.

00:05:52 And another aspect to this problem is: How is machine learning in FinTech different than the rest of the field?

00:06:03 It is different, actually, from two points of view: We are under a lot of regulatory scrutiny and compliance, and we have high exposure to risk.

00:06:15 And because of these two aspects, we tend to prefer simple models over complex models. And we actually have an internal team dedicated to reviewing our models and making sure that

00:06:30 our models are interpretable and can be described to regulators. But, essentially, where the differentiation comes in is: high risk versus low risk.

00:06:49 So for areas of low risk, such as marketing, we have much more freedom and much more extensive use of machine learning,

00:06:56 because the risk is low and if a machine learning model can provide improved accuracy, then the low-risk exposure enables us to use that model.

00:07:07 Versus areas of high risk, like capital forecasting. In areas like capital forecasting, our models need to be simple and interpretable

00:07:17 because we really need to isolate the effects. For example, if we use unemployment in a capital forecasting model,

00:07:25 we need to really understand what the impact of such a macroeconomic variable is on the forecast. So depending on the risk exposure,

00:07:36 we differentiate between the simple models and using more complex applications. So now I'll discuss my area, the area that I focus on, which is recommendation systems,

00:07:51 and I will talk about one of our use cases, which is recommendations for Ultimate Rewards. So a little bit about the problem:

00:08:00 Ultimate Rewards is Chase's system of loyalty, where consumers can translate their spend on their cards into points,

00:08:11 which they can then redeem on the Web site against a variety of options, like gift cards, you can redeem for cash, travel, experiences, and much more.

00:08:24 And the goal of the business is engaging the customer and enticing the customer to redeem their points. And we achieve this goal with recommendations via ad targeting.

00:08:39 So we do ad targeting and we try to match each consumer with the redemption option that they are most likely to be interested in.

00:08:49 And we combine the goals of showcasing the best redemption option while, at the same time, optimizing for cost.

00:09:01 So those two aspects are included in the optimization: finding the redemption option that is most likely to appeal to you while, at the same time,

00:09:11 keeping in mind the cost of the redemption option to the firm. The last thing that I will discuss is one algorithm which is used for recommender systems,

00:09:22 and this is probably the most fun part of the talk because here I'm going to get into some of the details of the algorithm and the intuition behind it.

00:09:31 Recommender systems have been popularized in industry in recent years, especially with the Netflix challenge.

00:09:41 The Netflix challenge really raised the bar on recommendations, and the research community produced really a lot of great research in that respect.

00:09:52 And recommender systems are used in a variety of businesses. For example, Amazon does product recommendations,

00:10:01 Netflix movie recommendations as I mentioned, Spotify music recommendations... So they are used for a variety of different types of products.

00:10:14 So let me discuss the problem setup. The problem setup is you start with a collection of users, which can be in the millions,

00:10:23 and a collection of items. And when I say "items", items can be anything from products, as in the case of Amazon,

00:10:30 movies, gift cards, as in the case of UR... so just a collection of things that you want to show to the consumer.

00:10:42 So these are the two dimensions of the model. And we form a ratings matrix which encompasses the user item interaction.

00:10:58 So the rating matrix contains information about how each consumer rates a particular item. And the rating can be explicit – in the case of Netflix, users are asked to explicitly rate movies

00:11:14 on a scale from 1 to 5. Or it can be an implicit rating – in the example of a news article, the time spent by a consumer

00:11:26 reading a particular Web page is an example of an implicit rating. So we form this user item rating matrix.

00:11:37 And one thing to point out is that this matrix is sparse, because users will not have interacted with all the items,

00:11:46 but actually, users generally interact with a very small subset of items. And the number of items can be in the hundreds, thousands, or even tens of thousands.

00:11:58 In the case of movies, there's tens of thousands of options, and one particular customer would have likely seen maybe a hundred movies.

00:12:06 So we set up this sparse rating matrix. And the goal of the algorithm is to complete the matrix

00:12:14 and predict a user's rating of all the possible items. So how is the matrix completion achieved?

00:12:26 What the algorithm does is a matrix factorization, so collaborative filtering. The algorithm that I'm focusing on does a matrix factorization.

00:12:35 So the sparse matrix is factorized into two matrices of smaller dimension: a user matrix and an item matrix.

00:12:46 And these matrices are then multiplied to reproduce the original rating matrix. And once the rating matrix is completed for a particular user,

00:13:04 all ratings of items are ranked in order of importance, and the first-ranked item, or the first three or the first five, depending on the application,

00:13:19 are shown to the consumer. So that is the general idea of the algorithm.

00:13:25 And what the algorithm really does under the hood is leverage user and item correlation. So, in a sense, it takes advantage of this "users like me" concept

00:13:36 where similar users are clustered together and the users are recommended items which users like them have liked in the past.

00:13:51 And this algorithm is similar to neighborhood-based collaborative filtering, except that here, the filtering is done on users and items at the same time.

00:14:04 And another algorithm that it is similar with is principal component analysis. So it relies on this idea of the dimensionality of reduction

00:14:15 and finding the latent dimensions in the data by exploring the correlation. One other aspect that I would like to point out is that this method is an unsupervised learning technique,

00:14:28 so there are no class labels. But the problem can be posed as a multiclass classification where you can think of each item

00:14:38 being considered as a class. For example, a gift card for Starbucks you can model whether a consumer is interested in that or not.

00:14:50 But the challenge in treating the problem as a multiclass classification problem is that if you have 10,000 items, then you have to build 10,000 models,

00:14:58 whereas collaborative filtering essentially performs the prediction in one shot. So coming back to our Ultimate Rewards example,

00:15:12 as I mentioned, our loss function incorporates both engagement and cost, and the optimization is done incorporating these two dimensions.

00:15:25 So thank you, I hope you enjoyed your course, I hope I gave you an idea of the machine learning techniques and use cases that we have in banking.

00:15:36 And I hope you had a great time – thank you.

Week 05 Unit 04

00:00:06 Hello everybody, welcome to Week 5 of the openSAP Deep Learning course. Today, we're going to be talking about Medical Image Segmentation with Fully Convolutional Networks.

00:00:17 My name's Gabriel, I'm a PhD student at Stanford University, and for my research, I do a lot of work with deep learning and medical imaging,

00:00:26 and so that's why I'm here to explain medical image segmentation today. So in particular what we're going to be talking about today is...

00:00:34 we're going to be discussing what exactly medical image segmentation is and what some other applications of deep learning and medical imaging are.

00:00:41 Then we're going to talk about fully convolutional networks, which is a popular architecture that people use specifically for medical image segmentation.

00:00:50 And then I'm going to talk about some of the specifics of how we actually use fully convolutional networks in the medical image segmentation case.

00:00:57 And then there are some IPython Notebook examples that I've made that will show you how to train a fully convolutional network to perform brain tumor segmentation.

00:01:08 Okay, medical image segmentation – so what are the goals of medical imaging? In general, we want to see inside somebody's body,

00:01:19 and not just because we're curious, but usually because there's something going wrong. Either the person has a disease, or they have some pain,

00:01:27 or certain body parts are not functioning correctly. And so the question is: How can we find the things that work and how can we find the things that don't work,

00:01:34 such as which parts of the body are diseased? And so with medical imaging, we get a lot of medical image data,

00:01:41 which is the acquired images from the medical imaging devices. And so most of the time, these devices... we can't directly image the body – we have to use

00:01:52 various techniques, such as X-rays or MRIs, which stands for magnetic resonance imaging. And so this returns three-dimensional or four- dimensional images,

00:02:04 because we image the body in 3D, which leads to three-dimensional pixels. So not like a 2D image that you normally see on the computer, but really like a 3D volume.

00:02:16 So these images have a width, a depth, a height, and depending on whether there's multiple modalities, they can have multiple channels as well.

00:02:24 Here I have an example. So here's an example of how people typically view medical images.

00:02:32 Because they're 3D, we can't just view them like a 2D image on the screen. We have to figure out a different way to do that.

00:02:39 And so typically what's done is we look at the image through various cross-sectional planes. And these cross-sectional planes are called the axial plane, the coronal plane, and the sagittal plane.

00:02:50 And then we can also take a 3D view of the image, as shown here. So the axial plane is like a top view – you look at the body from the...

00:02:58 actually, it's a bottom view – you look at the body from the bottom and you look upwards. The sagittal view looks from the side.

00:03:05 And the coronal view is a view from the front. And so this way, if we look at a cross section through all these planes,

00:03:10 we can get a good idea of what's going on in the body. But as you can see here, since medical images don't directly acquire the structures inside the body,

00:03:21 the interpretation can become difficult and requires very specialized knowledge a lot of the time. And so this is where the interest in deep learning comes in because it can potentially help us to

00:03:32 make it easier to analyze medical images. Medical image segmentation with fully convolutional networks – so what exactly is

00:03:42 the goal of medical image segmentation? Well, usually we have some image – either 3D or 2D – and we're interested in finding particular structures.

00:03:52 So, we may be interested in: Is there a tumor? Where are the blood vessels? Is there something abnormal in this particular part of the body?

00:04:01 And so this is different from image classification, for example, where we would have an image and we want to make a particular classification.

00:04:09 So, for example: Is there an aneurysm or is there no aneurysm? Is the patient healthy or not healthy? Whereas with image segmentation,

00:04:17 we're really interested in identifying exactly where in the image the structure of interest is. And so here I have an example with trying to find blood vessels.

00:04:26 And so the way we perform image segmentation in this case is that the output of our network, or at least the output that we're interested in, is another image,

00:04:35 except here, every pixel is labeled 1 if it contains a structure of interest, and 0 if it doesn't. So, for example, here a pixel value of 1 indicates that there is a blood vessel there in the picture,

00:04:48 and a pixel value of 0 indicates that there is no blood vessel. And so this way, we get an image with two regions,

00:04:55 one region containing the structure of interest, which is why this process is called "image segmentation" – because we're segmenting the image.

00:05:02 And so, how do we actually do image segmentation with deep learning? Well, as I said at the beginning, people typically use fully convolutional networks for this.

00:05:12 And so, what are fully convolutional networks? Well, as the name implies, fully convolutional networks are neural networks

00:05:20 that primarily use convolutional layers. In fact, most of the time, a fully convolutional network only uses convolutional layers.

00:05:28 And a particular characteristic of fully convolutional networks is that the input is an image, and the output is also an image.

00:05:35 So this is different from regular tasks, for example, like classification where the output of the network is just a vector of the particular classes.

00:05:45 Here, the output is really an image itself, like I showed in the previous slide, with labeled pixels. So let's go into that in a bit more detail.

00:05:55 So how do we do supervised learning for image segmentation? Well, just like with regular supervised learning,

00:06:02 we need a data set of input examples and ground truths. And so, since our output is a segmented image,

00:06:12 now the labeled images, or the ground-truth images, also need to be segmentations. So we need to get ground-truth images

00:06:20 where the structures of interest have been correctly labeled at the pixel level. So really, every pixel needs to be labeled.

00:06:28 Then, once we have those, we can simply perform supervised learning as you would with a regular network, except now we just feed in the example images and the ground-truth labeled images.

00:06:40 And so, for the model we use a fully convolutional network, as shown here. You can see the various convolutional layers, which eventually results in the segmented image.

00:06:50 And then we also use specific loss functions, just to take into account that the output is now an image instead of a vector of numbers.

00:07:00 So that's fully convolutional networks for medical image segmentation in a nutshell. Of course, to understand these things, it's best to actually work with them and get an idea that way.

00:07:10 So for that reason, I set up an example problem with a fully convolutional network on the BraTS data set, which is for Multimodal Brain Tumor Image Segmentation,

00:07:22 which is a popular sort of competition in medical imaging and should highlight some of the specifics of what it takes to get deep learning to work with medical imaging.

00:07:33 So I'll go over that now. So I have two Notebooks:

00:07:38 one to do the data processing, and one to train the fully convolutional network. In particular, with medical imaging... to work with medical images, you need to understand

00:07:50 the different file formats that they come in, because they come in a variety of file formats. And in particular, the BraTS data is stored as DICOM images,

00:08:01 which is just a very popular medical image data format, and so, we'll need to use special Python libraries to access them.

00:08:11 In particular, here we're going to use SimpleITK, which stands for the Simple Imaging Toolkit, which is very popular in medical imaging.

00:08:21 So in the first cell, we just import the packages we need, like SimpleITK, some operating system packages – NumPy, matplotlib... fairly standard stuff.

00:08:34 With some code to just make the plotting a bit prettier. Okay, so now we can start reading the data and getting the file paths.

00:08:45 So once you've downloaded the BraTS data, you can store it in this data path, and then, what we're going to do in particular is... BraTS is multimodal,

00:08:58 which means that it has images of each patient which have been imaged at different modalities, which means that different settings for the MRIs were used.

00:09:10 So, for example, you can use the magnets in various ways to highlight different structures in the body. And so, for example, here we're only going to use one modality, just to keep things simple,

00:09:28 which is FLAIR MRI, which stands for fluid- attenuated inversion recovery. This means that we use the magnetic pulses of the MRI in a way that will

00:09:39 decrease the signal from fluids in the body, which are usually fairly bright in MRI, and this will allow us to see the surrounding structures better.

00:09:48 Okay, so in particular, now we just need to get the file paths of all the FLAIR MRI DICOMS and load them and convert them to NumPy.

00:09:57 So that's what we do here – we just search the BraTS data folder, look for all the FLAIR input images and the ground truths, and store them in a list.

00:10:08 Okay, so before we go into a whole bunch of data processing, it's usually a good idea to inspect the data and see what we're working with, just so that we know it's correct.

00:10:17 And so that's what we do here. And as I mentioned earlier, when we look at medical images,

00:10:21 we tend to look at a cross section rather than the whole thing at once, as otherwise it becomes difficult to view.

00:10:27 So here we look at a particular slice for one of the images, and we can plot both the cross section of the image and also the ground-truth segmentation,

00:10:39 since that's an image as well. So you see here that we have the cross section of what is obviously a brain

00:10:47 with a brighter structure here, which in this case, is the tumor that we're trying to segment. And so here, we see the segmentation, and we can see the tumor is a bright spot,

00:11:00 but in particular, different parts of the tumor have different levels of severity, or different classifications, and so, we're not just trying to identify the tumor,

00:11:09 but really we want to know for each part what the category is. So we're doing multiclass segmentation here.

00:11:19 Okay, so now that we know what the images look like, that they're correct, and what we can expect from them,

00:11:26 we can now start processing the data into a data set for training and testing. In particular, what we're going to do is loop over every 3D image

00:11:35 and just extract 2D cross sections from it, so that we can use those as input for a 2D convolutional network.

00:11:45 Another thing to be aware of with medical imaging, if we go back, is that in the segmented image, most of the pixels are 0.

00:11:54 This means that the segmentations are sparse, so there's not a lot of signal, or a lot of ground-truth pixels for the network to learn from,

00:12:02 and so, if we're not careful, the network will just learn to predict 0 everywhere. So for this reason, what's typically done with medical imaging is that we split the data set into two parts:

00:12:12 One part, which is called the negative set, is where we just take arbitrary slices and we don't care about how many ground-truth pixels they have,

00:12:22 so many of those will just be sparse, or even 0, and then a positive set, where we make sure that most of the segmentations in the positive set,

00:12:31 or the ground-truth labels in the positive set, have many labeled pixels, and then during training, we sample from both of these sets to make sure that we are

00:12:42 always feeding the neural network at least some images that have a lot of ground truth, and this will help to stop it from predicting just 0.

00:12:51 Okay, so to store the data, we use HDF5. HDF5 is a data format that's pretty useful for deep learning.

00:12:58 It lets us work with the data without having to load it all into memory, which is convenient because here the data set gets up to a few gigabytes if we load it all.

00:13:09 And then the final part, as you probably already know, for neural networks, to train them we have to normalize the input images,

00:13:15 and then we also crop them to a size of 128x128 pixels, just to make learning a little bit faster. So that's what we do here.

00:13:24 We define some functions just to make things easier, so we first have our process image segmentation function to process a single slice,

00:13:33 and so this just takes the slice and crops it, And then we have a split positive/negative, which looks at the segmented image.

00:13:44 The segmented image sees which ones have a small ground-truth value and which ones have a large ground-truth value,

00:13:50 and then splits that into the positive and negative sets. And then we have a function to do the entire data set collection.

00:13:59 So we use HDF5, we create our data sets, then we use SimpleITK to load each image, and then we process them to get the slices and then store them in the data set.

00:14:11 Yeah, so that's the data processing, so now let's get to the exciting part, which is training the network.

00:14:18 So here I have the notebook for training the fully convolutional network. In particular, we're going to use TensorFlow to build the network and optimize its weights.

00:14:27 So again, we just import the packages we need, like h5py to access the data, TensorFlow for the training. We're going to need to specify the data path,

00:14:36 where we get our data from, because we need that for training. And so, yeah, here you can see that we can use h5py to open the HDF5 files,

00:14:44 and then once they're open, they are mostly, like, a NumPy file. So we can do things like look for the maximum of the training array

00:14:52 to see what number of classes there are. And we can also look at the shapes.

00:14:58 And so here you can see that we ended up with 14,622 training images and 1,474 test images. Right, so that's a decent amount.

00:15:14 Okay, and so now, for building the network, what I like to do with TensorFlow is, before going into it, I like to make some convenience functions, just to make building the network a bit easier.

00:15:25 Just because working with TensorFlow directly is a bit low level. So since we're going to train a convolutional network with many convolutional layers,

00:15:34 it's handy to define one function to build a convolutional layer, and then another function to apply many convolutional layers in sequence.

00:15:43 In particular, we'll also be using the leaky rectified linear unit, just because it tends to form a bit better than the regular rectified linear unit and still is quite fast to evaluate.

00:15:55 Finally, for training, we need to sample batches, as you probably know, since we'll be using stochastic gradient descent,

00:16:02 so we need a function which now randomly selects positive and negative examples from our data set. So that's all that we've done here.

00:16:09 The leaky ReLU function, our two-dimensional convolutional function where we can specify the input, the dimensions of the filter, the number filters, the strides, the weight initialization and

00:16:20 the activation function, just calling TensorFlow, and then the convBlock function, which will just call our convolutional function a number of times,

00:16:29 and return the final output tensor. Okay, so now the network parameters.

00:16:36 So this is always the difficult part to get right with deep learning: How many layers? How many filters? What filter size? What learning rate?

00:16:47 And the answer is that with all these parameters, you simply try a lot of things and see what works, or you have to look in the literature and see what other people have tried, to get an idea.

00:16:57 Here we'll keep things simple, so I did a bit of experimentation to find something that works for you guys,

00:17:02 but feel free to experiment with other things. So we're going to use a network with five layers, each layer has a filtered image of 7x7,

00:17:11 and we'll train for 10,000 iterations with a learning rate of 1e-3, which tends to be a good learning rate for fully convolutional networks.

00:17:21 Additionally, since – as I mentioned before – all the labeled images are somewhat sparse, and to deal with this, we made a positive and a negative set...

00:17:31 however, even with doing that, the network can still learn to output 0 or miss many of the labeled pixels. And so, what we want to do is add weights to our loss function,

00:17:43 which will amplify the loss on the rare pixels. Because this will ensure that the network tends to focus on those pixels instead of just outputting 0.

00:17:51 And so, that's what we define here as well. So you can see that the first label, which is just the empty space, we give a weight of 1,

00:17:59 but then the tumor labels, we increase the weights by quite a bit, so things like 60.0 and 20.0, which I found through experimentation worked well,

00:18:10 just to make sure that the network picks those Okay, and so now we can finally go into constructing the TensorFlow graph.

00:18:18 So for input, we need to have placeholders, and the input image, of course, has a width and a height, and a number of channels,

00:18:27 so our x placeholder needs to be a four- dimensional placeholder, because we also need an extra dimension for the batch size.

00:18:34 And, similarly, for the labeled segmentation, we just need three dimensions, because it's just an image where each pixel value denotes the label.

00:18:44 We then convert the input segmentation into a one-hot vector, because that's what's needed for learning.

00:18:53 And then we set up the convolutional part of our network, so we call our convBlock with the number of layers that we want,

00:18:59 and then we use a final convolutional layer, just to make sure that we output an image with the correct number of classes.

00:19:05 So in this case, we want to output an image with a number of channels equal to the number of classes. So it's like we're performing multiclass classification on each pixel.

00:19:20 We then take a softmax to convert the output into the predicted probability map. And then we set up our weighted loss function, which is just the cross entropy function

00:19:31 multiplied by the class weights, and then we take the average. Okay, so for now, we're ready to start optimizing and training our network.

00:19:39 So we build the AdamOptimizer using TensorFlow, and then we start to train it and we initialize everything.

00:19:46 Adam tends to work well, so I highly recommend trying with that, but feel free to experiment with other things.

00:19:52 Okay. So here's a section that trains the network.

00:19:56 We simply loop over all the training iterations, we get batches of data, and we run the training operation. Really straightforward,

00:20:04 and then every now and then we evaluate on the test set, just to see how things are performing. So here is our loss curve, and we can see that it looks quite erratic,

00:20:15 mostly because we're using stochastic gradient descent. In this case, it looks like the loss is kind of going down,

00:20:22 but it's unclear whether it's really converging. So here, it would definitely be recommended to train for more iterations,

00:20:28 just to get a better idea of what's going on. Okay, but now we can still take a look at what our network is predicting, to see what it does.

00:20:37 So here, we look at the test set and evaluate the network predictions on the test set. And in particular, we take the index of the maximum predicted probability at each pixel,

00:20:48 and this gives us the output segmentation, so that's what we've plotted here. So in general, here the first image is the input image, the second image is the ground-truth segmentation,

00:20:57 and the third image is what the network predicted. So, in general, we can see that the network isn't doing a bad job.

00:21:03 Especially surprising considering that we have such a simple network. Just that it tends to get the structure of the tumor correct, the general categories.

00:21:16 However, it does seem to miss a few parts of the tumor during its classification, or it labels a few erroneous pixels.

00:21:24 Such as here, where it's picking up bright spots in the brain and labeling those as tumor. However, on the whole, not bad, especially considering where we started from,

00:21:36 but definitely some room for refinement. That being said, we can see how image segmentation works now with fully convolutional networks,

00:21:46 and it's a fairly complex task. And so getting these things right requires quite a bit of experimentation with the network architecture,

00:21:55 and so I'd highly encourage you to experiment with things like more layers, more filters. Maybe even adding more paths to the convolutional network,

00:22:05 maybe if you want to look at some papers and see what they're doing, you can get some ideas from that as well.

00:22:12 Yep, but any case, that's how fully convolutional networks work for medical image segmentation, and I hope you enjoyed the lecture.

00:22:18 Thank you.

www.sap.com