**Interviewer 1:** [00:00:00] Okay, so I would start with some background questions.

**Interviewee:** [00:00:06] Okay.

**Interviewer 1:** [00:00:07] What is your current position at your job ?

**Interviewee:** [00:00:09] I'm currently actually changing jobs. So this is my last week as a postdoc at the [university: removed for anonymity]. Okay. And from next week, I'll be joining as Data Scientist at gas company here in [city: removed for anonymity].

**Interviewer 1:** [00:00:27] Okay, so congratulations on your new job.

**Interviewee:** [00:00:30] Thank you.

**Interviewer 1:** [00:00:32] So could you tell us what is your overall work experience? And what is your experience specifically in deep learning/machine learning systems?

**Interviewee:** [00:00:39] Okay, so I'll start with the machine learning which I think it's a little bit easier. I've been doing that since the last, I guess, 4 years or something, since I moved to [city: removed for anonymity] to do my PhD. I mainly worked on machine learning for biomedical applications, so in the context of neurodegenerative diseases, mainly. What we have done, basically, we collected data from patients and then we applied machine learning models to predict how the disease will evolve in time.

**Interviewer 1:** [00:01:20] Okay.

**Interviewee:** [00:01:21] Okay, in terms of how for instance disability increases or decreases in time. And I'm mainly using shallow learning so far, meaning that my experience with deep learning is somehow related to side projects I've done. So, for instance, the most important side project I did was last year, when I was in [city: removed for anonymity] doing an internship there and I was working on developing a deep learning model that predicts change in diabetes.

**Interviewer 1:** [00:02:04] Okay.

**Interviewee:** [00:02:05] From administrative records, such as claims of insurance and stuff like that. And in that case I was using Keras.

**Interviewer 1:** [00:02:16] Okay.

**Interviewee:** [00:02:17] With backend Tensorflow, but basically just Keras. And, yeah, what I've been doing is to, what I did actually is to basically implement the attention mechanism, which is a mechanism that works on recurrent neural networks, such as LSTM and etc., directly from a paper. So I was reading a few papers on the topic and then I implemented back on Keras, because at the time there was no standard implementation for that.

**Interviewer 1:** [00:02:54] Okay.

**Interviewee:** [00:02:56] And, yeah.

**Interviewer 1:** [00:02:58] Okay. So thanks a lot. So you told us that your experience in deep learning is 4 years and your overall work experience? Could you just mention also that?

**Interviewee:** [00:03:08] Yeah, it's 4.5, since I started my PhD.

**Interviewer 1:** [00:03:14] Okay. Also, you mentioned the frameworks that you have been using, which is keras. And as a programming language, I would guess you've been using Python.

**Interviewee:** [00:03:21] Yes.

**Interviewer 1:** [00:03:22] Okay. So, we have like one general question for you, which is what kind of bugs/problems/challenges you have faced.

**Interviewer 2:** [00:03:34] Technical issues.

**Interviewer 1:** [00:03:35] Yes, while doing this. So maybe if there is anything on your mind to answer that it would start from there. And then...

**Interviewee:** [00:03:43] Okay. Well, I think that the main problem when you work with those kind of frameworks is that when you program something in Python you usually just read it from top to bottom and then you will understand the flow of the logic you're implementing, right? It's not always the case when you're using deep learning frameworks, because sometimes what you write is not immediately done. And another big problem working with deep learning is that the dimensions of tensors that combines one to another must always be consistent, but since unless using Pytorch, which I'm not an expert, but I've heard, and unless you're using the brand new Tensorflow, which the new version of tensorflow I think is going to solve this problem. Anyway, the problem is that when you read from top to bottom is not always clear if the dimensions of your tensors would be consistent and you find it, of course, it is not compiled. So you find it at runtime maybe even after hours of computation, for instance.

**Interviewer 1:** [00:05:00] Yes.

**Interviewee:** [00:05:01] Of course. If you inspect the summary, Tensorflow and Keras they do a great job on preventing that, because they can show you a preview of what's going to happen in your neural network when you print it, either using the Keras summary or either using like TensorBoard or tools like that. But still sometimes happens that tensors they don't have a consistent shape, so everything will break. And then you figure out later.

**Interviewer 1:** [00:05:36] So you told us about this tensor dimensions. Do you remember any specific bugs that happened to you? When you had a wrong dimension of anything, like of input, of a layer?

**Interviewee:** [00:05:47] I don't remember the exact name of the issue, of the exception that is raised by Python. It's probably something very specific like "wrong dimensions" or something like that. I don't remember, but the point is that you're trying to compute the inner product between two tensors with different shapes. So it just breaks, that's it.

**Interviewer 1:** [00:06:12] Anything else?

**Interviewee:** [00:06:14] Well, another problem is, well, it's not strictly related to deep learning, probably small general machine learning is that you have to be careful when you program, because, I mean, most of the time your programming with machine learning is like a small percentage of time you are working on the model. And then the majority or I mean the biggest majority of time you are working on the pipeline. Okay, so from the data to the result and there could be bugs that, for instance, make you leak some of the training set into the test set.

**Interviewer 1:** [00:06:52] Aha.

**Interviewee:** [00:06:53] And there's no way to check that before. So standard programming tools, the standard software tool, they have no way to to see if you're leaking that information. And if you're doing that, you could seriously harm your pipeline, because then you will have over-optimistic results, because you're going to be testing on a training set. So that's another big issue when you program something in machine learning, which I have no idea how to do.

**Interviewer 1:** [00:07:24] So, by leaking the training data into test data, do you mean you do not split it in the right way or that...?

**Interviewee:** [00:07:31] Maybe you split it, but you know, maybe you know splitting is usually easy, because you just split and then you're going to have indexes for the training and the test set. But the point is that what you do before splitting may involve the whole dataset, you may use transformations for instance or whatever on your whole dataset and then even if you split, your leaked information in the test set because you do training set as well and there's no way to... And this is not a bug per se, because the program will run, but it is more like a conceptual bug, because the the results you will have are going to be biased and it isn't gonna be working for real.

**Interviewer 1:** [00:08:21] Okay anything else to add?

**Interviewee:** [00:08:26] I don't know. I don't think so.

**Interviewer 1:** [00:08:31] Okay. So when you were working with this machine learning /deep learning systems, did you use existing datasets or did you collect your own data?

**Interviewee:** [00:08:43] I most of the time use my own data, because in biomedical scenarios it's unlikely that you have a previous data on the same subject, collected the same way, but this is just, I mean a specific case. Because usually when you work, for instance, only images, you use, you start from previously collected data by someone else, previously created image.

**Interviewer 1:** [00:09:10] So could you tell us a bit more about the data collection and some bugs and problems related to that?

**Interviewee:** [00:09:18] So bugs in data collection?

**Interviewer 1:** [00:09:21] Not exactly bugs, but the problems, like if you had enough data or if some of the data content was wrong or you had to change it in some way to be able to train better. Things like that.

**Interviewee:** [00:09:37] I don't know. I don't think I have any relevant issue on that.

**Interviewer 1:** [00:09:43] Okay. So did you you ever had to pre-process the collected data before?

**Interviewee:** [00:09:49] Yes, of course all the time.

**Interviewer 1:** [00:09:51] Yes. Could you mention some pre-processing actions that you have taken?

**Interviewee:** [00:09:55] Okay. So, when speaking of pre-processing there are several things you can do and that of course, it depends according to the type of data you have and this is, for instance, something where the conceptual bug I was mentioning before would take place, because if you process your wholedata set before splitting, and as I mean, there's no way up, I mean so far, there's no way a machine learning tool can identify that before you doing it? And yeah, that's again, it's not going to be a bug because it's all going to work, but still you will have to, you pre-processed your full dataset and that's just wrong, you leaked information again between the training and test set.

**Interviewer 1:** [00:10:46] Okay, and other problems related to training data that you might remember or? You said that you did not have problems with collecting it, but maybe during training...

**Interviewee:** [00:10:59] A problem you may have is that, when you in particular in deep learning you are always optimizing your loss functions with batch gradient descent, which is this algorithm that takes a bit of data and then runs over and over on small batches of your dataset. And, in particular, if you're using GPUs, it is possible that you test it on your machine, then you want to deploy it on the server with the GPUs and etc., then the dimensions of the batches you selected and everything, they just don't fit run there. So it would be nice to know that in advance and then there are some some tricks you can do, but, I mean, they're not very straightforward, because it's hard to know in advance how much RAM your whole network will need. So it may happen that a small portion of data or a smaller version of your network that you are using to develop the network, then is not going to be consistent with what happens when you try to deploy it on a large scale scenario. So yeah, and that will of course raise an issue because you will eat up all the RAM.

**Interviewer 1:** [00:12:27] Okay, so maybe we could talk about the structure of the models you have been using. So you said that you use mostly shallow models that are not very...

**Interviewee:** [00:12:38] Yes.

**Interviewer 1:** [00:12:39] But did you ever face like problems related to wrong model structure, where you had too many layers or too few layers? Where you had to change anything about model structure to make your training work at all or work better?

**Interviewee:** [00:12:52] Well, you know the structure of your network is your biggest degree of freedom. You can do whatever you want. Just so far, there are no mathematical guarantees of any of that. So, I don't think that, I mean, it's very hard to, using Keras, for instance, it is very hard to find a bug in this case, because Keras helps you a lot in building one layer after another, just by mentioning them, the whole network. So I don't think that unless you want to build your own layer and there is this place where you need to specify all the dimensions of the tensors and etc, and thats related to the first source of possible bug I was mentioning. So, unless you're doing that, which it's not the case unless you have very specific task, nowadays Keras has most of them. I am always talking about Keras, because it's the one I've used the most and I am not really an expert on PyTorch. But yeah, it's usually all the layers that are popular they are already implemented there so. I don't think that's a possible source for bugs. And anyway, you always have TensorBoard, which helps you a lot in looking at your network, layer after layer. So.

**Interviewer 1:** [00:14:22] Yeah, okay. Thank you. So about hyperparameter tuning, any problems about that.

**Interviewee:** [00:14:31] Well, yeah, the main problem with that is that it's computationally very intensive, because often you have no idea how to define the some dimension. So you would like to tune it, but to tune it you need to run cross-validation and if you want to run cross-validation, it means that you have to run multiple fits and if one fit takes, I don't know, one week, then you would have multiple weeks. You will need multiple weeks to tune just one input parameter, and then maybe you have 20 of them. So yeah the main issue there again, it's not a bug per se, it is more like technological gap we have to fill, is that the most of the hyperparameters by themselves they have just, they don't have meanings by themselves. So you don't have a way to redefine it apriori. Okay? You just have to test different values and there are tools which helps you doing that, in particular also, the brand new version of Keras has some automagic to do that. There are heuristics and yeah, usually people try not to tune them by cross-validation, unless it's needed, because sometimes training times can be very high.

**Interviewer 1:** [00:15:55] But do you maybe remember an instance when you changed some hyperparameters, increased/decreased it and you got an improvement in your accuracy or improvement in your performance?

**Interviewee:** [00:16:08] Yes. It happens. Of course, for instance when you have to choose, I don't know, the first thing that pops in my mind is the Dropout parameter, that's very important. Because, since Dropout allows you to exploit the whole ensemble structure of your network. It helps you a lot in generalizing and regularizing your network. So yeah, that's an important thing. And another important one is the hyperparameters related to early stopping, which is the strategy to stops training whenever it is useless going any further, and yet the number of steps you allow the network to not get better helps you when you are stuck, for instance, in a local minima of your loss function. And yeah, those parameters are important and they can dramatically affect how good your performance is.

**Interviewer 1:** [00:17:14] Okay. So another question I have is about the loss functions. Do you use a predefined loss functions? Did you ever have a custom-written one, any problems that you remember about these loss functions?

**Interviewee:** [00:17:27] I usually, no, I've never properly written a custom loss function. I always use the previously implemented ones. So I don't recall any problem with that. One minor problem probably is that when you approach at first developing neural networks with Keras, it's probably from the documentation not very clear how the dimensions of your output should be, how your output should be shaped. So, for instance, when you have categorical classes, then you may have like a one auto-encoding or any different dummy encoding of your class, of your output. Because that output is the same you going to see at the output of your network. So the output of the two things should match and from the documentation it was not very clear on how to shape that. But once you find it once, it's basically the same for everything. So.

**Interviewer 1:** [00:18:39] Okay, so you mentioned that you sometimes have problems with GPU and that models can take a long time to train. So where do you train your models and do remember any problems related to hardware, other than memory issues?

**Interviewee:** [00:18:58] Okay. So yeah, according to the GPU specifics you are using, of course the training time will change.

**Interviewer 1:** [00:19:06] Yeah.

**Interviewee:** [00:19:08] And yeah, usually we train our network on the server we have the [university: removed for anonimity].

**Interviewer 1:** [00:19:16] Okay.

**Interviewee:** [00:19:17] Which is mounting two GPUs. And I don't know apart from high computational time and RAM that finishes, I'm not sure we had any other issues. Also because you know, all the neural network libraries are developed... Oh, yeah, maybe this is a problem. You are sticked basically to work with NVidia GPUs, because none of the deep learning frameworks for non-NVidia GPUs, so developed instead of being in Cuda, they are developed on OpenCL, are not as good as in terms of performance, as the one developed in Cuda. So one problem the open-source community now has is to fill that gap and to implement libraries in OpenCL that can work on any GPU. Because, so far, Nvidia, basically they develop the best libraries. So Tensorflow itself is using as a back-end those QDNN(?) libraries. So one problem that you may have is that you need to download the Cuda deep learning libraries. So you need to have an account on CUDA and everything.

**Interviewer 1:** [00:20:40] Okay, that's interesting.

**Interviewer 2:** [00:20:42] So you mentioned that your server has multiple GPUs.

**Interviewee:** [00:20:49] Yeah.

**Interviewer 2:** [00:20:49] When you run your code on multiple GPUs, did you face any concurrency problem?

**Interviewee:** [00:20:55] No, because I usually use one of the best Keras feature, which is a function that uses multiple GPUs and basically the algorithm that implements as far as I got, it is like that. It splits your batch in two, you assume that your GPUs are equal, of course if you have one that is faster than the other you may have problems, but just problems in terms of computational time. So it splits the competitions among the GPUs and any documentation it says that up to 8 GPUs, which is the I think the highest number you have on AWS server, if I remember correctly. The speed-up is almost linear, because it splits the training batch in multiple GPUs, computes on each GPU the small variations of gradient and then aggregates everything back on the full gradient. So yeah, you can exploit multiple GPUs in this way and I think in deep learning so far, it is the most common way to exploit multiple GPUs, but it's completely transparent from an implementation point of view, because you implement all of your logic and everything and then basically you just wrap everything around this "use multiple GPUs" function that does all the splitting and combining approach, so.

**Interviewer 2:** [00:22:32] So, you don't write all of this code yourself.

**Interviewee:** [00:22:35] No. Nowadays, it is hard, I mean it would be weird now to develop something in Cuda or to properly handle with GPUs, unless you are an NVidia software engineer. If you're a Data Scientist or deep learning/machine learning practitioner, this is usually very easy to just call high-level libraries that exploit the low-level libraries.

**Interviewer 2:** [00:23:03] Thank you.

**Interviewer 1:** [00:23:06] So you mentioned this RAM problems and I wanted to ask if you do any steps to kind of anticipate it? Do you use any practices to manage memory as best as possible, so that you don't run into this troubles?

**Interviewee:** [00:23:22] Okay, so not to running into that troubles at all is hard. What I usually do is to launch beforehand. There is a little NVidia script that keeps the performance of your GPU always on your screen and if you try to, since it is almost impossible to foresee, I mean not impossible of course, but very very hard to foresee the amount of RAM taken by your neural network, I usually start with like very small batches of training data. Like for instance one sample or few samples and that is easy to calculate how big will it be, right? Because if it's an image, you can just, since it is all 32 bits, you can just count, but you will not have the right number of parameters of your network beforehand. So what I usually do is that, I launch the network with this small batch and see how much RAM is occupied and then just from that, I don't just train the network with that. I just launch it to see how much memory will it take and then I kill it and increase it, if I see that I have a little bit of space, that's what I do. Of course, it's suboptimal to be, it's not very a way to deal with that. Yeah, that's what I do.

**Interviewer 1:** [00:24:51] So you mentioned that you have been using machine/deep learning for biomedical problems, let's say.

**Interviewee:** [00:25:00] Yes.

**Interviewer 1:** [00:25:00] So maybe you could tell us a bit more about, like you have to collect data and you have to make predictions based on this data. So how do you select relevant features? Do you remember cases when you included something, but it wasn't useful or you had to add some things for your accuracy to improve? Can you give us more details on these?

**Interviewee:** [00:25:19] Yeah, so basically the things like that, you will have your input features and then you would have more data that you have on your patients. For instance, if it's a patients-based dataset. Okay, and it's likely that you will you use different sources of data, such as the age, sex or whatever, just to identify groups of similar patients. Okay, just to perform some patients' matching. And one problem that that you may have is that sometimes you need to switch between Python and R, because some routines that are very common are implemented in R libraries and other modules are implemented in, of course also in R, but I find it more easy to work with python. So yeah one problem you may have is this, that you will use two different languages to accomplish the same project and the way you collect data is usually related to the doctors, so it's usually up to them.

**Interviewer 1:** [00:26:30] Okay.

**Interviewee:** [00:26:31] Yeah, and that's a big issue because, they are not always a tech-savvy, so they might give you data in weird formats or with different conventions and etc. And, for instance, something that is done in [continent: removed for anonimity] or at least in [country: removed for anonimity] is to use the comma as a separator of decimal. Okay, and sometimes doctors do that without even knowing, because Excel is doing that. So yeah, you will struggle a bit to get the features from those weird formats and then you need to select the relevant ones and selecting the relevant features again is a problem of designing the right pipeline to do it. Because, it's very very, because feature selections per se, for instance, it's not a pre-processing step. So you cannot run feature selection on everything and then on the same data run your prediction model. Because you already giving to you to the model the right feature for that specific instance of training set. So this is a problem related to the information leaking between training and test set, I was mentioning before. There is no way to have, up to now, there's no way to have some machine learning framework that helps you in finding these conceptual bugs. And to perform feature selection, yeah, it depends very much on the model you're using and the data you have, if you have categorical data, if you have continuous data, so if you have categorical data, for instance, you may want to use decision trees and sample of decision trees because they handle well with categorical data. And not deep learning, because deep learning is more, I would say, useful if you have perceptual data, like images or textual data, like that. But if you have categorical features that you know are useful you just can use decision trees and in that case the important measure of the feature will be different than for instance when you use sparse linear regression, which would be another case of doing some embedding feature selection inside the fitting model and as will be the case for instance for linear continuous gaussian data.

**Interviewer 1:** [00:28:54] So, can you give us like an example of something like you used some feature and it was not useful or you had to provide a different categorical analysis of that feature. Anything like that?

**Interviewee:** [00:29:07] Well, I think that usually when when you design a machine learning pipeline and you have features that are not important, you should not take them out. Because you should let the model to assign a low weight on those features and just discard it. This is what happens when you use the sparsity enforcing methods. They just will discard unrelevant features. Of course, if you have very few important features and you are submerged with a lot of noisy useless features, then you should put a very much effort in enforcing sparsity and the problem should work anyway, because deleting features apriori, it is somehow fishy.

**Interviewer 1:** [00:29:59] I see. So likem I guess that's what you've done, you have kept the features, you got low weights for them. And after you removed them did you see improvement in accuracy or rather in performance?

**Interviewee:** [00:30:11] Yeah, you don't really see, I mean, the the model can handle it by itself. So you will see the difference between doing, I don't know for instance linear regression which takes all the features and doing sparse linear regression which suppresses useless features. So you would see the difference in accuracy and I mean the difference, it can be huge. Because many confounding factors must be, can be filtered.

I see.

**Interviewer 1:** [00:30:42] So I also wonder how you evaluate these models, cause what you said is that you are trying to predict the history of the disease, which is not like a yes/no or it's not even like categorical.

**Interviewee:** [00:30:55] In my case, I always had like a transition in the disease course, disease type. So I had a binary label which was indicating onr type of disease or another and this little label was given by doctors not looking at the data I using to predict, but looking at MRIs or other stuff. This is another important bit, because if the relationship, you must always know if the relationship between input and output, if the label is given by someone who looked at the same data you are looking in your machine learning model or it comes from an external source, that is another important bit that I don't know maybe might let you to draw to the wrong conclusions. But yeah, unless you have something that is quantitative or at least measurable, you can not evaluate your model.

**Interviewer 1:** [00:31:59] Okay, so, I think one last question is whether do you remember any very nasty bugs that took a lot of your time and, yeah, caused a lot of problems for you?

**Interviewee:** [00:32:13] So let me think, a very nasty one.

**Interviewer 1:** [00:32:18] Or mildly nasty one?

**Interviewee:** [00:32:23] I think the the nastiest I can remember is yeah. Okay, so there was this problem when I was writing the attention layer in Keras, I had this problem in figuring out why, I mean looking at the paper it looked like the implementation was right, but then I had. I don't even remember how I solved eventually. Because I was trying so many solutions. I think it was always related to the size of the of the tensors by the end of the day. I think, I was using a transposed version of the tensor instead of the normal one, and it took a lot of time to understand that. Because yeah, you have to follow many many others things.

**Interviewer 1:** [00:33:25] Okay. So do you have anything else to add?

**Interviewee:** [00:33:29] No, I don't know, I don't think so.

**Interviewer 1:** [00:33:32] Okay, so we thank you a lot for your time.

**Interviewer 2:** [00:33:36] Thank you.

**Interviewer 1:** [00:33:37] Bye. Have a nice day.

**Interviewee:** [00:33:39] Bye, bye.