**Name:** Interview-09-20220802

**Created On:** 8/9/2022 4:49:34 PM

**Created By:** WJ

**Modified On:** 8/12/2022 5:12:18 PM

**Modified By:** WJ

**Size:** 13 KB

Speaker 1:

Okay. First, I'd like to ask some questions about your background. The background questions can be answered briefly so that we can save time for the other questions. First, could you briefly describe your role in your team?

Speaker 2:

Okay. Yeah, well, related to machine learning, I have two relationships I guess. One is I'm a data scientist at a UAV company, which is a data collection company with drones, that uses drones to collect data. Basically, my job is to use this data, mostly aerial images, and get insights from it. Maybe classification tasks, object detection tasks, and so on. Maybe use things like HuggingFace to upload our models and use the inference API to just use them after we have trained them. Then I'm a master's student in the National University of Columbia. I'm currently working on deep learning for medical imaging. To keep it short, basically that's my background right now and my relationship to machine learning.1

Speaker 1:

Okay. Can you tell me a recent time when you used a portray model from an external model hub?

Speaker 2:

Why do I use an external model?

Speaker 1:

Yeah.

Speaker 2:

Basically, because it's faster to get something going. Right now, maybe models are very robust and very easy to maybe fine tune to your problem. Instead of building the wheel again, you just use a pre-trained model that maybe is good for your problem. Then, you just use it. It's maybe easier to just do that and use these model registries just to make things faster. Maybe sometimes you just don't need something very custom made for your problem. It's just faster and easier, I guess.2

Speaker 1:

Have you used any model as a backbone?

Speaker 2:

Yeah. For my research, I usually use EfficientNet as a backbone for my problems. Since the tasks that I do at my job are not that hard, I usually just pick a decent enough model in, for example, HuggingFace for image classification or for object detection. I just use those and then fine tune it.3

Speaker 1:

Okay. Then we'll move on to the second part, which is about the model selection. Here, we are trying to understand the process that engineers follow as they decide which Pre-trained Neural Network to reuse in their projects. Can you think about the last time why you chose a pre-trained model from a model hub, how did you choose it? Can you summarize your decision making process?

Speaker 2:

Okay, I mean, it depends on the problem. For example, my research, I of course read papers and I look up what's the thing that everyone's kind of moving towards and which backbones are they using? Usually, I just follow that kind of, I don't know how to say trend, I guess. For the other part, I basically just look for models that are built for the problem that I'm trying to solve. If I want to do image classification, I look up probably the best performing image classifications models on ImageNet, or something like that, or CIFAR, or whatever in a hub. Then, I used them.4

Speaker 1:

When you say it's the best model, do you mean they have the best performance in terms of the accuracy? Or you can see different metrics?

Speaker 2:

Usually, it's accuracy. Maybe you can look up different metrics. For example, for the medical part, you'll look up at the KAPPA metric, but usually in classification, it's just maybe accuracy, maybe the number of parameters. If it has enough accuracy and not many parameters, then it's probably better because it's faster to fine tune. It's not a very complex process that I follow, but I think it works. It works fine right now.

Speaker 1:

Where do you usually select the pre-trained models? There are many models from the model hubs and there are also many models from the GitHub projects. Which one do you choose?

Speaker 2:

Well, before, I used to look at GitHub. Well usually in the READMEs they have the instructions to get their model going. But right now, I think it's easier to just use a model hub. For example, well, I used [inaudible 00:06:17], but I think HuggingFace is very, very simple. I think the ease of use is very important if you're trying to use a pre-trained model because you probably just want to get something going fast.

I think hugging face is really good. I think their documentation is really good. Sometimes in other hubs, you're kind of guessing how to do some stuff. One of the main maybe disadvantages of the hubs that I've seen are maybe they don't have enough documentation for data pre-processing. For example, they usually have examples of how you should compile the model and how you should train it. But then you're kind of lost on the part that's before that. Okay, you use this certain data and this certain data has this size or whatever.

But then when you're trying to adapt it to your problem, they don't have many documentation and you just have to kind of guess or try to make it work. That's one of the disadvantages. Also, it's very easy to use as it is, but if you want to customize it, then you're kind of, again, in a bit of a problem. For example, in the HuggingFace models, they are made in PyTorch usually. They say, "Yeah, you can use normal PyTorch to do your stuff," but then it's wrapped in many obstructions that going back, it's hard.5,6

Speaker 1:

Yeah.

Speaker 2:

So yeah.

Speaker 1:

Okay. The next question is when selecting the model, do you care more about the performance than the architecture?

Speaker 2:

Probably yes, I think.

Speaker 1:

Okay.

Speaker 2:

Yeah, I think, I don't know, there's this trend that I don't really like in ML right now that you just use whatever is best now and you don't really care... I mean, unless you're doing research on building new models or something, you really don't care how they are built. For example, this trend of transformers and this image generation models is crazy. I don't know. I feel like it's not the right way to do it. They're just putting more parameters, more computation, and they just say like, "Okay, it's working better. It works better, so whatever." I don't think it's right, but I think how it is right now, you just choose whatever works best at the moment, I think.7

Speaker 1:

Yeah. Do you think the pre-trained models, which is in the model registries, can accurately describe their behavior?

Speaker 2:

Sorry. Can you repeat that?

Speaker 1:

Do you think the pre-trained models available in the model registries can accurately describe their behavior?

Speaker 2:

No. Well, no, you mean interpretability?

Speaker 1:

Including all their claims in the model cards or the documentations, including all the metrics.

Speaker 2:

Oh, okay, okay, okay. Well, I think they don't really describe how it works on the insight. Usually, model hubs just reference the paper where the model was published and it is like, "Okay, you can read how it works, the instructions of how to use them are this." Usually, if you're not doing research, you just use it. Maybe the part of reading the paper and knowing how did they do it? It's not very relevant I guess.8

Speaker 1:

Okay. To what extent do you think the discrepancies of the performance metrics can affect your decision making? Here, for the discrepancies, I mean the actual performances can sometimes be different from the camp performances in the models.

Speaker 2:

You mean the difference between the performance that they report and the actual performance that you are getting?

Speaker 1:

Yeah, to what extent do you think that would affect your decision?

Speaker 2:

I mean, it's very significant, because at least for me, I try different models and then it really doesn't matter... At first, you choose the one that is reporting the best metric, but then you try to like, "Okay, I'm going to try to do my problem. Then I'll see if it's working well with my problem or not." Then you probably start trying different models that maybe are not reporting the best accuracy, but for your problem, they work better. I guess the final decision is of course, after you try it in your problem, not only what they report, but the initial choice, I guess it's based on the report accuracy metric that they-9

Speaker 1:

Here, you say you should try the model first, do you mean you should fine-tune the model and see the performance on your datasets?

Speaker 2:

Yeah.

Speaker 1:

Okay.

Speaker 2:

I feel like that's... Yeah.

Speaker 1:

Okay. Next question is to what extent do you think the robustness of the models can affect your decision?

Speaker 2:

The robustness? Let me think. Robustness. I mean usually, when a model is more robust, I mean it should probably work better with your data or any different dataset. But usually, robustness is not very like... I don't know, maybe you know more than me, but it's not reported. In the model hub, they don't say like, "Okay, this model is robust and it works well with different datasets." They usually just report another metric. I actually don't know how to measure robustness other than maybe training the model with different datasets and seeing like, okay, it works well with different types of data. But I don't know, do you know how you measure that? That would be maybe a good addition to a model hub like, okay, you worked on different types of data. That's probably even more important than the accuracy.10

Speaker 1:

I was thinking about how to measure this robustness thing for model hubs. The model hub can report the robustness.

Speaker 2:

Yeah.

Speaker 1:

The next question is to what extent do you think the explainability of the models can affect your decision?

Speaker 2:

The interpret of-

Speaker 1:

Yeah.

Speaker 2:

... the application?

Speaker 1:

Yeah.

Speaker 2:

Okay. The thing is you don't see that when you choose the model. You just know that after you've worked with the model and then you say, "Okay, it works well. Now I'm going to maybe use, I don't know, class activation modules, something to explain what the network is seeing." But then it only happens after you do this explainability analysis or something. I don't know, at the start, it may not be very important, because the model hubs don't report explainability kind of data, but [inaudible 00:15:40]-

Speaker 1:

If the model hub can report this kind of metrics, it could be helpful?

Speaker 2:

Of course, yeah.

Speaker 1:

Okay.

Speaker 2:

It would be very helpful, I think, especially for example, in the medical side of things, for example, when I'm working on... We have this problem that, okay for example, the model works well classifying cancer, but we don't know what's the reasoning behind the network. Then usually, doctors or pathologists are very, I don't know how to say it, but they don't trust these kinds of systems. You kind of have to work on the explainability for them to actually trust it and maybe for it to transition into production, into real life, so they can them to help them decide and diagnose and all that.11

Speaker 1:

Okay. Next question is how frequently do you re-train the models?

Speaker 2:

Do I re-train models?

Speaker 1:

Yeah.

Speaker 2:

Well frequently, I guess, if I'm trying to... Like I told you at the beginning, when I'm trying a model from a model hub, I usually train it on my data and see how it works. I guess if that counts as training, I guess.

Speaker 1:

I mean, do you re-train the model using their training data?

Speaker 2:

No. No, no, no.

Speaker 1:

Okay. Okay.

Speaker 2:

No, because I mean, there's this problem too, that, well for example, in HuggingFace, most of the models are transformers, right?

Speaker 1:

Yeah.

Speaker 2:

It's very expensive to re-train something like that. For example, for me, I don't have the money or the computational power to re-train that. I just have to kind of trust that they did it well and just use them. Because it's impossible for me to re-train a network that big, it's not a possibility.

Speaker 1:12

Okay. Do you think the lack of trainability or fine-tune ability is a problem while using the pre-trained models?

Speaker 2:

The lack of what, sorry?

Speaker 1:

Trainability or fine-tunability.

Speaker 2:

Yeah, yeah, yeah. Especially the trainability. Right now, with this trend of very big models, for students, maybe... Unless you're in a very, very good university that has something like a deal with Amazon or something, you literally can't re-train a network like that. That's a problem, because you just have to trust everything that they've done. I mean, you kind of do already, because usually the pre-trained models are from big companies or universities. You're kind of like, "Okay, they probably did it right." But of course, re-training it is impossible. Fine-tuning them, it's okay. Usually, with some transformers that I've done things, actually fine tuning them is very kind of simple because they don't need much data or training times. You can just like, "Okay. It works."13

Speaker 1:

Okay. Okay. The last question for this part is what other challenges do you face when selecting a pre-trained model from a model registry?

Speaker 2:

Challenges?

Speaker 1:

Yeah.

Speaker 2:

The data processing, I guess. Sometimes, they have this pre-made function that helps you pre-process the data. You just use that and then, okay, it's ready for the network. But then maybe you don't know what they're doing with the data. Sometimes, especially with sensitive data, medical data, it's... Maybe some progressing that they do is not relevant or is bad for the problem. The training of course, with what we mentioned before, you can't re-train a very big network at all. What else? Maybe the flexibility, they make it so easy that it's very inflexible. It's very easy to do, but if you want to change something, it's very hard. I think that those are the most... I think that's the main challenges.14

Speaker 1:

Okay. Then, we'll move on to next set of questions, which is about the deep learning software attributes. We would like to learn about what sort of information is useful to engineers who reuse the pre-train models from the model registries? Here, I'd to show you some... Here are the three traditional attributes, which is defined by NPM, which is used for the JavaScript packages. For quality, popularity, maintenance, can you take a look at the definition here? The first paragraph here and let me know when you are ready?

Speaker 2:

Okay.

Speaker 1:

Okay. Are you ready?

Speaker 2:

Yeah, yeah.

Speaker 1:

Okay. For these three attributes, what do you think would best help your team or you select a pre-trained model from the model registries?

Speaker 2:

Personally, I will... Probably the most important ones are quality and popularity.

Speaker 1:

Can you give a order of these three?

Speaker 2:

Order? Okay.

Speaker 1:

Yeah.

Speaker 2:

Okay. For me personally, it will be popularity first, quality second, and maintenance third. I feel like popularity is a good way of knowing where I guess the trend is going, what are people using? I think it's good. Usually, when a package is popular, it usually comes with the others too. It's very well documented usually and they do maintenance regularly. I think that and then I think that's it.15

Speaker 1:

Okay. Then I will show you three different specific attributes, which we proposed in this project. I will ask you several questions about each of them later. First, I'd to ask define the provenance here, which we defined as a measure of model lineage or traceability. Some examples are like whether it's a model documentation contains links to the paper, whether it contains the dataset, architecture, something like that. Can you think about a time when you met the provenance problem before when using the pre-trained models?

Speaker 2:

A problem?

Speaker 1:

Yeah.

Speaker 2:

Not really. Not really. I think the provenance attribute is usually common right now. For example, in HuggingFace or the other hubs, usually, that's what they usually always have. They usually have the link to a paper, they of course know what's the account, or the company that uploaded the model. They usually report which dataset it was trained on. The architecture, sometimes they show a little part, okay, this is the same as this, but with this tweak or this change and it works better.

Usually, the architectures are getting very obscure, I guess. You don't know what's going on in there. That may be a problem if you're really into deep diving into your model architecture. I think maybe the architecture is a little bit of a problem because they don't really disclose it totally. They just show little parts.16

Speaker 1:

What do you think would be useful to know beforehand? Or what do you think the model hubs or model registries can provide in order to solve these problems?

Speaker 2:

That's a hard question. I don't know. I don't know what they could do to make it... I mean, usually, for example, the architecture is... Okay, let me check my index, one sec. Usually, they just... Like this Microsoft one. They don't really say that much. How to site it and that's it. Okay. I think the architecture, usually the architecture, when they have it, it's just a screenshot from the paper. Okay, we changed this and they don't explain it.

It works better and you should use it. I mean, that's fine, but if you are doing something very important, I guess, where you have to know how the network is inside, then... They probably should maybe dumb down the architecture a little bit, like a summary of how it works. It would be pretty good I think, maybe not deep diving into everything, but just showing a small summary of how things work. Like, "Okay, we use this convolution layers, and then we do this, and then done."17

Speaker 1:

Do you mean if they visualize the neural network, it could be helpful?

Speaker 2:

Yeah. The thing is the models are so big now that doing the visualization of the network is going to be too big. Maybe something that shortens the architecture, but has the... You can get the idea of how it works, that'd be cool.

Speaker 1:

Okay. Okay. Then, we'll talk about the reproducibility. Here, we define it as the ability of different practitioner to produce the same accuracy and training or evaluation time from a pre-trained model, as defined in the papers source code or the original group. Can you think of a time when you met any reproducibility problems while using the pre-trained models before?

Speaker 2:

Yeah. Yeah, that one, yeah. One time, I was building this baseline for my master's thesis. They had, literally, how you say it there, they had all the steps, like, "Okay, for you to reproduce our results, you need to do this, do this, this, this." It was in GitHub and they had every source code, every notebook, the data was probably the same that I'm using. It was very straightforward, I guess. But then when I got into actually reproducing it, I had a lot of problems because the code was very complex and then it didn't work.

Then, I had to look into someone else's code, which is always a pain, and change it, and fix stuff. Then after all those problems, I managed to make it work and then I didn't get the same accuracy. I even used the same seed and everything, but I didn't get the same. Yeah, it's a problem. I don't know, maybe because of the sarcastic nature of networks, I don't know. But this is very hard. Usually, I don't know about you, but I usually never get the same result. Something pretty close, but I never get the same result.18

Speaker 1:

What do you think can be useful to know beforehand to improve the reproducibility of models?

Speaker 2:

Well, the hyper parameters are very important. The hardware may be a problem, maybe. Sometimes when you're trying to reproduce something, they say, "Okay, we trained this in 16 GPUs." Then you have one and it's going to take you 50 times longer. They can see where there's experience. I don't know how to make it better actually. I mean, usually, they tell you the seeds that they used to make it reproducible. In PyTorch, they usually do this deterministic flag. It's more reproducible too. But then at the same time, you don't get it. You don't get the same results, so I don't know. I actually don't know what you could do better. Maybe packaging the whole thing into a... No, but that's... I don't know.

Speaker 1:

You mean like a Docker image, which can give you the whole environment?

Speaker 2:

Yeah, I guess, but then maybe a Docker image. Maybe uploading it to one of the hubs and wrap it into all the stuff that they have. You just do the whole thing in one line, but I don't know if it's useful.19

Speaker 1:

Okay. All right. Last one is portability. Here, we define it as with which an engineer can take a pre-trained model and reuse it in other environment or software project. Can you think about time when you had portability problems before?

Speaker 2:

It's usually hardware. I think in that side, what we were talking about earlier about HuggingFace, like, "Okay, we trained this transformer in a TBU pot that costs $1 million per day to run." I mean, you just can't do that. I feel like that's one of the main issues. The fine-tuning could be more documented I guess. Usually, the models, like in HuggingFace, they tell you how to do it with the same data that they used, but then maybe some documentation of, "Okay. How can you do it with your data?"

It would be pretty cool because you could save a lot of time. The energy consumption. It's very related to the hardware because it's like a TPU pod, okay, then you just can't run it, it's too expensive. They report training times and stuff with the top hardware than when you're going to do it, it's going to take you 50 times longer. That's also a thing like, "Okay, you can fine-tune it, but then it's going to take you three days."20

Speaker 1:

What do you think would be useful to know beforehand in order to solve the portability problems?

Speaker 2:

Maybe performance or time per part or something, depending on the hardware that you use, that will be maybe useful. Well, the framework that they use, it's usually fine, they usually report it, like, "Okay, we did it in PyTorch, 10 seconds," whatever.

Speaker 1:

Yeah.

Speaker 2:

The fine-tuning, more documentation about fine-tuning because usually, people... Well, I don't know, but usually, you want to use whatever pre-training network that you choose on your data. You usually don't want to use it on the same data that they use. Maybe more documentation there. I think this part is kind of okay. The main problem of these three that I see is the reproducibility, but the portability is usually okay I guess.21

Speaker 1:

Okay. Well, I think we are running out of time, but I have five more questions. If you have to leave, you can let me know.

Speaker 2:

No, it's okay. Don't worry. Don't worry.

Speaker 1:

Okay. Okay. Thank you very much. The last question for this part is except for these three attributes, do you think there are any other attributes that could be helpful for the pre-trained model we're using?

Speaker 2:

That's a good question. I don't know, maybe, but I don't know what. Let me check here.

Speaker 1:

Maybe you can think about what we have discussed for the previous questions and maybe there can be something helpful.

Speaker 2:

Yeah, I'm thinking about it, but it's very hard. Usually, I mean there hasn't been another attribute ever. Maybe. I don't know, relevance to some tasks, but usually they have that. In HuggingFace, for example, they say, "Okay, this model is for image classification, this one's for object detection, this one's for text generation." Those tags, those flags, or whatever, are useful. Because maybe if you're maybe starting into the deep learning thing, you just look through that. I want to identify something, let's look at the image classification models, and then use one of those.

Those types are useful. The data that they were trained on is useful too. Well, something that's very nice, I think of HuggingFace especially, is the inference API that they have on every model, where you can upload any image or whatever or text or anything and the model loads and does its thing. I think that's pretty cool. It could actually work. It could actually be a good... a very naive, but good method to maybe choose a model, just play with it a little bit and it's, okay, it kind of works well with what I kind of want to do. That's cool. I don't know what else, maybe examples.

Speaker 1:

Do you mean the demos?

Speaker 2:

Demos maybe, yeah.

Speaker 1:

Okay.

Speaker 2:

I can work.

Speaker 1:

Okay.

Speaker 2:

I don't know what else, honestly.22

Speaker 1:

Okay. Yeah. The last set of questions is about the pre-trained model trustworthiness. We are trying to understand how pre-trained model shortcomings affect engineers ability to rely on and reuse them. Which aspects of the pre-trained model do you assume are trustworthy?

Speaker 2:

Like what we said earlier, maybe usually trust top universities and big tech companies. It's maybe not the best metric to trust a model, but you usually think, "Okay, if Google did it, then it's probably well done. They probably didn't fake their results or report something that's not real." That's one of the ways that makes you trust in a model. Also, the popularity is also a good metric to trust something. Then of course, the paper, I guess the paper is very important too. If you actually have to check it out and read it, then you can kind of get an idea if it's a good model or not.23

Speaker 1:

Yeah. Have you found any discrepancies between the [inaudible 00:41:12] pre-trained models and the downloaded version in terms of accuracy, latency, and architecture?

Speaker 2:

That's a good question. The thing is, as I told you, usually at least us, we don't... we train it on the same data. We don't know if there's a discrepancy between what they board and what you see. Usually, you just trust the metrics. Then as I told you, when you try on your problem, then you see if it's useful or not for your problem. It's hard to see a discrepancy when, for example, the model was trained on 14 million images. Yeah.24

Speaker 1:

Yeah, okay. If the discrepancies exist, to what extent do you think they are acceptable?

Speaker 2:

The discrepancies usually refer to the discrepancies on the reported performance, right?

Speaker 1:

Yeah.

Speaker 2:

What else do you think?

Speaker 1:

For example, let's say if the discrepancy is less than a certain percentage of accuracy, what percentage do you think is acceptable?

Speaker 2:

I feel if it's close enough to the accuracy that they report, it's fine. Maybe like a 5% difference.

Speaker 1:

Okay.

Speaker 2:

I guess it's okay. But as I told you, it's hard to actually check that. Right now, you kind of have to trust it and just do your stuff and then see if it works or not. But checking if what they report is actually true, it's kind of hard, especially with the bigger and bigger models that are coming now.25

Speaker 1:

Okay. The last question here is do you think the discrepancies have significant impacts?

Speaker 2:

Yeah, they could. It also depends on your problem, I guess, because sometimes you... It depends on the problem. Sometimes, an 80% accuracy is okay, sometimes 95% accuracy is not okay. It really depends on your problem, but of course, you kind of have to try to... I mean, what they report should be what the reality is, but it's of course a problem, you usually expected to behave how they shared it. Depending on the problem, it can be very problematic, I guess. Sometimes not, but on some problems, yeah, for sure.26

Speaker 1:

Okay. Okay. All right. That's all for my questions. Thank you very much. I'll stop recording.

**Annotations**

1 Role:

1. A data scientist at a UAV company. Classification, object detection.

2. Master student working on DL for medical imaging.

2 Reuse scenario:

- Because it's faster to get sth going.

- Maybe models are very robust and easy to fine tune.

- sometimes you just don't need sth very custom made for your proble.

3 Backbone:

- Usually use EfficientNet as a backbone

- My job are not that hard. I just pick a decent enough model in Hf for image classification or object detection. I just use those and then fine tune it.

4 Decision making:

- depends on the problem

- paper/what's the thing everyone's kind of moving towards and which backbones are they using.

- I basically just look for models that are built for the peoblem that I'm trying to solve.

- It's not a very complex process that I follow

5 Where:

- I used to look at GitHub .

- instructions in READMEs.

- The ease of use is very important.

- Why HF: sometimes in other hubs, you are kind of guessing how to do some suff. E.g. No enough documentatio nfor data pre-processing.

6 Model hub disadvantages:

- When trying to adapt to your problem, they don't have many documentation and you just have to kind of guess or try to mkae it work.

- Hard to customize.

7 Performance or architecture

- Performance > architecture

- You don't care how they are built.

- I don't like it.

8 Accurate behavior:

- don't really describe how it works

- Usually if you are not doing research, you just use it.

-

9 Discrepancy:

- very significant

- The final decision is of course, after you try it in your problem

10 Robustness:

- More robustness -> probably work better with your data or any different dataset

- Usually robustness is not reported.

- I don't know how to measure robustness. That would be maybe a good addition to a model hub.

11 Explainability:

- You don't see that when you choose the model. You just know that after you've worked with the model.

- It only happens after you do this explainability analysis or sth.

- If the model hub report explainability kind of data, it would be very helpful.

- especially for medical side of things. The model works well classifying cancer, but we don't know what's the reasoning behind the network. Doctors or pathologists don't trust. It's hard to explain.

- This kind of measurement can help them decide and diagnose and all that.

12 Retrain:

- No. It's very expensive to retrain transformer models.

- I just have to trust that they did it well and just use them. It's not a possibility for me to re-train them.

Fine-tune:

- Frequently.

13 Trainability/fine-tunability

- Yes, especially trainability.

- You literally can't retrain a network.

- Fine-tuning them is very kind of simple because you don't need much data or training times.

14 Other challenges:

- Data (pre-)processing.

- They make it so easy that it's very inflexible. It's very hard to change anything.

15 Traditional attributes:

- Quality and Popularity: Most important ones

- Popularity > quality > maintenance

- popularity: a good way of knowing the trend.

- It usually comes with the others too. It's very well documented and they do maintenance regularly.

16 Provenance issue:

- usually common

- usually the architectures are gettin gvery obscure. You don't know what's going on there. They may be a problem if you are really into deep diving into your model architecture.

- They don't really disclose the architecture totally. They just show little parts.

17 Provenance help:

- Usually the architecture. It's just a screenshot from the paper. They changed this and they don't explain it.

- Maybe not diving into everything. But just showing a small summary of how things work.

- Like "we use this convolution layers, and then we do this, and then done."

- Something that shortens the architecture but has the visualization... would be cool.

18 Reproducibility issue:

- They had all the steps there, every source code, every notebook, the data. Very straightforward. When I reproduced it, I got a lot of problems because the code was very complex and it didn't work

- Look into someone else's code is always a pain and change it and fix stuff. After I made it work, I didn't get the same accuracy with the same seed and everything.

- Maybe because of the sarcastic nature of networks. Something pretty close, but never the same result.

19 Reproducibility help:

- Hyperparameter

- Hardware, e.g. GPUs

- Usually they tell you the seeds that they used to make it reproducible

- In Pytorch, they usually do the deterministic flag -> more reproducible.

- Maybe a docker image. Maybe upload it to one of the hubs and wrap it into all the sutff. You just do the whole thing in one line, but I don't know if it's useful.

20 Portability issue:

- Hardware

- Fine-tuning could be more documented. How can you do it with your data.

- Energy consumption: very related to the HW

21 Portability help:

- Performance or time per part or something, depending on the HW you use.

- More documentation about fine-tuning

- usually you don't want to use it on the same data that they use.

- The portabiltiy is usually okay

22 Other attribuse:

- The tasks/categories are helpful

- The data that they were trianed on is useful too.

- The inference API that they have on every model

-

23 Trustworthiness:

- Trust top universities and big tech companies

- Popularity is a good metric to trust something

- Then the paper is very important too.

24 Discrepancies?

- No. Usually you just trust the metrics.

25 Acceptable discrepancy:

- maybe like a 5% difference

- It's hard to check that.

26 Significant impact:

- depends on your problem, it can be very problematic. But sometimes not.