Software Engineering Assignment

Johan Edstedt

August 2023

1 Introduction to My Topic of Research

My research (so far) involves estimating a 3D world from 2D camera images. I focus mainly on what is called the matching problem, i.e. matching pixels in images that correspond to the same 3D point in the world. This is an important task because it turns out that if we are able to say which pixels match, we can basically derive all the 3D we need from there. For example, we can get the depth, the relative position of the cameras, surface normals, and much more just from these matches. In particular, my research involves two extremes, so-called dense matching (finding every match), and keypoint detection and description (matching sparse sets).

2 Two Principles from Robert's Lectures Related to My Research

Verification vs Validation: Most of my work involves training machine learning models. I found the division between Verification: "Checking of whether we are building the product right" and Validation: "Checking of whether we are building the right product" to be particularly relevant. For example, verification would involve something like the following: Make sure that the input images that come into the model are standardized, or that the network is able to fit to a single example. Validation on the other hand would be on complete checks on the system, such as evaluating the model on a validation set after training. In my own experience, the slide stating that ML testing is more about validation than verification is only partially correct. In the early stages of a project I spend most of the time on verification, while in later stages (where all components are verified) we can compare approaches on a higher level. However, it may be the case in industry that the weight is primarily on validation.

Data Validation: Validating that the data that goes into the model is actually useful is important. In my own research, it's been especially useful when combining multiple datasources into a joint dataset. My own approach has been to manually visualize examples, and validate performance by ablation over the data.

3 Two Principles from Guest Lecture that Relates to My Research

I'll discuss the topic of boundary value testing from the lecture of Felix Dobslaw. In particular, I will discuss Boundary Value Testing in the context of 3D computer vision, and program derivatives.

Boundary Value Testing: I found the example given regarding the decision boundaries for loan approval enlightening and thought about how this could be applied to my own topic. One immediate challenge is how to transfer the concept of a binary boundary to regression targets that are continuous. However, in matching there are actually some binary decisions. For example, the decision of whether to trust a correspondence or not. It would be interesting to investigate more closely what causes some matches near the decision boundary to be classified as reliable while others end up as untrustworthy, and in particular what types of counterfactuals (changes in features) are required.

Program Derivatives: Another interesting topic brought up in the presentation was that of extending the notion of derivatives to programs that are traditionally thought of as non-differentiable. In particular, the challenges surrounding the choice of distance function when doing numerical differentiation.

My work is related to robust statistics, where outputs are often computed from non-differentiable optimization processes, such as Random Sample Consensus (RANSAC). Recent advances have enabled researchers to "bypass" this non-differentiability by either approximating the gradients [2] or by the implicit function theorem [3].

4 Topics from the List

4.1 Sustainability

My understanding of sustainability is basically that something is sustainable if we can keep doing it for a very long time without serious consequences to our environment (sometimes sustainability includes more than this, but I'll mostly focus on environmental impact). In the following, I'll look mostly at the carbon impact of deep learning models, and what Software Engineering practices could help.

In the seminal paper of Anthony et al. [1], a framework for computing the carbon footprint of deep learning models was proposed. They discuss multiple ways of reducing the environmental cost of training models, including scheduling training to times of the day when renewable energy sources are more abundant, as well as running more efficient implementations.

Here too I think Software Engineering can play a major role. For example, while the paper emphasizes the training cost of a model, there is often much experimentation that consumes orders of magnitude more resources than the final experiment. This experimentation is often inefficient, with ideas being tried out multiple times. I believe the DRY principle would be useful for practitioners when it comes to experimentation. Make sure that what you're testing is not a variant of what you have already tested previously and dismissed, as you will end up repeating more, leading to larger amounts of memory used.

Furthermore, proper software engineering helps prevent technical debt, which also matters for machine learning [5]. In the context of sustainability, this could mean less error-prone and more power-efficient code. Both these have the effect of producing models that consume less electricity, and in consequence, have a smaller carbon footprint.

4.2 Continuous Deployment

Continuous deployment (CD), in the context of machine learning, means frequent updates to trained models used for inference. A notable example would be Tesla, which does frequent software updates to its deployed FSD systems. I could technically possibly mean other things as well, but I'll focus on the inference type of deployment.

I chose the recent study by Paleyes et al. [4], for an up-to-date review of the challenges involved in deploying machine learning models. The authors there argue that CD is challenging when deploying machine learning models, as the changes made can affect three axis. Code, model, and data. They

argue that while CD is necessary for keeping a model up to date, it may also lead to harm, as changes in model behaviour may not be picked up by regular software testing. This can cause issues with models not being backwards compatible [6].

In my own experience, I have had to deal with these issues in my own research code, both while developing, and after open-sourcing code publicly. For example, I've had issues with backwards compatibility when updating the API of a model. One thing I've found useful is to have a small set of "sanity-check" example images on which I evaluate the model after any changes. If these are ok I usually let through any pull requests or changes. Of course, there may be edge cases where the model performance is affected, that can not be easily detected. This does seem like an inevitable problem with CD, rapid development can never be fully tested.

5 Future Trends and Directions

One aspect I've seen a lot of progress on recently is the standardisation of deployment of deep learning models. One big issue with `PyTorch` has previously been on deploying to other devices, e.g., phones. Standardized formats such as ONNX will continue to increase the interoperability between frameworks, which I think is an important future direction.

There are also efforts to unify the core neural network operations, for example mojo. These languages aim to abstract hardware details, to enable writing machine learning code that works on all devices.

In the context of ML engineering I think standardisation will remove a lot of the deployment burden. Rather, I think even more focus will be put on the curation and preprocessing of good data for the models.

Another trend I am observing is the increase of "foundation" models, i.e., very large models pretrained on large datasets, which you fine-tune or "prompt". Dealing with these types of models, which may be outside the control of the company, seems like it will continue to increase in the coming years. A common counterargument to the need of ML engineering in the future is that with these models, there will simply be an API that does not require any ML engineering. While this argument may hold some water, I think it overestimates the capabilities of foundation models. While this may be true in 20 years, current and near-future models are not general enough to be able to handle arbitrary tasks. This means that they will have to be adapted, and monitored continuously to work, which means ML engineers will be required.

In conclusion think the importance of ML engineering is likely to increase in the coming years. While unifying frameworks can reduce the need for ML engineers to handle deployment, the total amount of work surrounding the development of new ML models is more likely to increase than decrease, and software engineering principles will continue to be relevant.

References

[1] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems, July 2020. arXiv:2007.03051.

[2] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6684–6692, 2017.

[3] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004, 2021.

[4] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys*, 55(6):1–29, 2022.

[5] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.

[6] Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. An empirical analysis of backward compatibility in machine learning systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3272–3280, 2020.