Chapter 15. The Future of Pipelines and Next Steps

In the past 14 chapters, we have captured the current state of machine learning pipelines and given our recommendations on how to build them. Machine learning pipelines are a relatively new concept, and there's much more to come in this space. In this chapter, we will discuss a few things that we feel are important but don't fit well with current pipelines, and we also consider future steps for ML pipelines.

Model Experiment Tracking

Throughout this book, we have assumed that you've already experimented and the model architecture is basically settled. However, we would like to share some thoughts on how to track experiments and make experimentation a smooth process. Your experimental process may include exploring potential model architectures, hyperparameters, and feature sets. But whatever you explore, the key point we would like to make is that your experimental process should fit closely with your production process.

Whether you optimize your models manually or you tune the models automatically, capturing and sharing the results of the optimization process is essential. Team members can quickly evaluate the progress of the model updates. At the same time, the author of the models can receive automated records of the performed experiments. Good experiment tracking helps data science teams become more efficient.

Experiment tracking also adds to the audit trail of the model and may be a safeguard against potential litigations. If a data science team is facing the question of whether an edge case was considered while training a model, experiment tracking can assist in tracing the model parameters and iterations.

Tools for experiment tracking include Weights and Biases and Sacred. Figure 15-1 shows an example of Weights and Biases in action, with the loss for each model training run plotted against the training epoch. Many different visualizations are possible, and we can store all the hyperparameters for each model run.

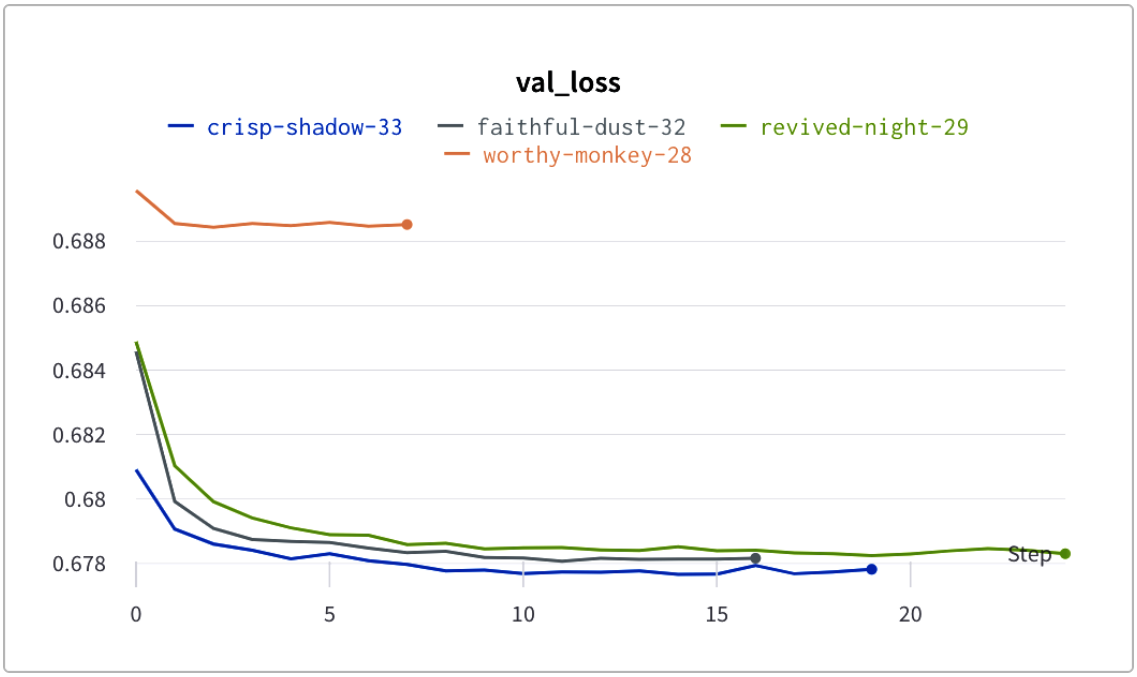

Figure 15-1. Experiment tracking in Weights and Biases

In the future, we expect to see the experiment and the production process become more tightly linked so that a data scientist can smoothly switch from trying out a new model architecture to adding it to their pipeline.

Thoughts on Model Release Management

In software engineering, there are well established procedures for versioning code and managing releases. Large changes that may be backward-incompatible get a major version change (from 0.x to 1.0, for example). Smaller feature additions get a minor version change (1.0 to 1.1). But what does this mean in the machine learning world? From one ML model to the next, the input format of the data may be the same and the output format of the predictions remains the same, so there is no breaking change. The pipeline still runs; no errors are thrown. But the perfor-

mance of the new model may be completely different from the one that came before. Standardization of machine learning pipelines requires model versioning practices.

We suggest the following strategy for model release management:

- If the input data is changed, the model version gets a minor change.
- If the hyperparameters are changed, the model version gets a major change. This includes the number of layers in a network or the number of nodes in a layer.
- If the model architecture is completely changed (e.g., from a recurrent neural network [RNN] to a Transformer architecture), this becomes an entirely new pipeline.

The model validation step controls whether the release happens by validating that the new model's performance is an improvement on the previous model's performance. At the time of writing, only a single metric is used in this step by a TFX pipeline. We expect that the validation step will become more sophisticated in the future to include other factors such as inference time or accuracy on different slices of the data.

Future Pipeline Capabilities

In this book, we've captured the state of machine learning pipelines at the time of writing. But what will machine learning pipelines look like in the future? Some of the capabilities that we'd like to see include:

- Privacy and fairness becoming first-class citizens: at the time of writing, the assumption is that the pipeline does not include privacy-preserving ML. Analysis for fairness is included, but the ModelValidator step can only use overall metrics.
- Incorporation of FL, as we discussed in Chapter 14. If data preprocessing and model training happen on a large number of individual devices, a machine learning pipeline would need to look very different from the one we've described in this book.
- The ability to measure the carbon emissions of our pipelines. As models become larger, their energy usage becomes significant. Although

this is often more relevant during the experimentation process (especially searching for model architectures), it would be very useful to integrate emissions tracking into pipelines.
- Ingestion of data streams: in this book, we have only considered pipelines that are trained on data batches. But with more sophisticated data pipelines, machine learning pipelines should be able to consume data streams.

Future tools may further abstract some of the processes in this book, and we expect that future pipelines will be even smoother to use and more automated.

We also predict that future pipelines will need to tackle some of the other types of machine learning problems. We have only discussed supervised learning, and almost exclusively classification problems. It makes sense to start with supervised classification problems because these are some of the easiest to understand and build into pipelines. Regression problems and other types of supervised learning such as image captioning or text generation will be easy to substitute into most components of the pipeline we describe in this book. But reinforcement learning problems and unsupervised problems may not fit so well. These are still rare in production systems, but we anticipate that they will become more common in the future. The data ingestion, validation, and feature engineering components of our pipeline should still work with these problems, but the training, evaluation, and validation parts will need significant changes. The feedback loops will also look very different.

TFX with Other Machine Learning Frameworks

The future of machine learning pipelines will also likely include openness regarding underlying machine learning frameworks, so that a data scientist doesn't need to choose between building their model in TensorFlow, PyTorch, scikit-learn, or any other future framework.

It is great to see that TFX is moving toward removing pure TensorFlow dependency. As we discussed in Chapter 4, some TFX components can be used with other ML frameworks. Other components are going through a transition to allow the integration with other ML frameworks. For example, the `Trainer` component now provides an executor that allows training models independently from TensorFlow. We hope that we will see more generic components that integrate frameworks like PyTorch or scikit-learn easily.

Testing Machine Learning Models

An emerging topic in machine learning engineering is the testing of machine learning models. Here, we don't mean model validation, as we discussed in Chapter 7, but rather a test of the model inference. These tests can be unit tests for the model or complete end-to-end tests of the model's interactions with an app.

As well as testing that the system runs end to end, other tests may center around:

- Inference time
- Memory consumption
- Battery consumption on mobile devices
- The trade-off between model size and accuracy

We are looking forward to seeing best practices from software engineering merge with data science practices, and model testing will be part of this.

CI/CD Systems for Machine Learning

With machine learning pipelines becoming more streamlined in the coming months, we will see machine learning pipelines moving toward more complete CI/CD workflows. As data scientists and machine learning engineers, we can learn from software engineering workflows. For example,

we are looking forward to better integrations of data versioning in ML pipelines or best practices to facilitate deployment rollbacks of machine learning models.

Machine Learning Engineering Community

As the field of machine learning engineering is forming, the community around the topic will be vital. We are looking forward to sharing best practices, custom components, workflows, use cases, and pipeline setups with the machine learning community. We hope this publication is a small contribution to the emerging field. Similar to DevOps in software engineering, we hope to see more data scientists and software engineers becoming interested in the discipline of machine learning engineering.

Summary

This book contains our recommendations for how to turn your machine learning model into a smooth pipeline. Figure 15-2 shows all the steps that we believe are necessary and the tools that we think are best at the time of writing. We encourage you to stay curious about this topic, to follow new developments, and to contribute to the various open source efforts around machine learning pipelines. This is an area of extremely active development, with new solutions being released frequently.

Figure 15-2. Machine learning pipeline architecture

Figure 15-2 has three extremely important features: it is *automated*, *scalable*, and *reproducible*. Because it is automated, it frees up data scientists from maintaining models and gives them time to experiment with new ones. Because it is scalable, it can expand to deal with large quantities of data. And because it is reproducible, once you have set it up on your infrastructure for one project, it will be easy to build a second one. These are all essential for a successful machine learning pipeline.