CS 329S: ML in Industry

Pete Warden, petewarden@google.com

Who am I?

Tech Lead on TensorFlow Lite Micro.

Founding member of the TensorFlow team, helped create TensorFlow Lite.

Previously CTO of Jetpac, a deep learning startup acquired by Google in 2014.

Why am I here?

Chip invited me!

I’ve worked with hundreds of teams building real-world products around ML.

I hope I can share my perspective on what works, what doesn’t, and what’s different from academia. My goal is to prepare you for some of the challenges I wasn’t expecting when I first started to try to ship ML products.

Please jump in and ask questions at any point, I hope this can be interactive.

What will I be covering?

- Case studies
- Common challenges
- TinyML
- Server and client-side differences
- Federated learning

Case Studies

Plant Village

Agricultural non-profit sponsored by UPenn

Easier to talk about than most commercial projects, since it’s more open, but faced a lot of the same challenges.

https://grow.google/intl/europe/story/transforming-farmers%E2%80%99-lives-with-just-a-mobile-phone

PlantVillage - Goals

Build an Android app that helps developing-world farmers identify and treat diseases on cassava plants, an important food crop.

Had to run locally on a phone, since most locations didn’t have reliable cell coverage.

Plant Village - Initial approach

Trained botanists travelled to farms in Kenya and took hundreds of thousands of photographs of cassava leaves, and labeled them with any disease diagnosis they spotted.

Plant Village - Initial results

Training an ML model on the images produced impressive (9x% accuracy) from the evaluation loop on the reserved test set.

Using MobileNet v1 meant it was fairly easy to get running on a phone, with low latency (less than one second per inference).

This is the point many research projects would declare victory, publish a paper, and move on. However ...

Plant Village - Field testing

Next, they gave the resulting application to farmers to try out. Lots of problems emerged!

- Different image framing. Farmers aim the camera very differently from botanists.
- Training set prevalence of diseases didn't match real-world prevalence. The outputs of the model weren't reliable probabilities.
- Confidence was hurt by bad results on "out-of-domain" images. "My hand has leaf rot?"

Plant Village - Different image framing

To handle the more varied framings, switched to a model that used bounding boxes and object localization to frame disease areas, versus whole-image classification.

Plant Village - Misleading probabilities

Calibrating the results by priors, in a Bayesian way:

- What’s the overall rarity of the disease?
- What area is it most common in?
- What time of year does it occur?

Plant Village - Out-of-domain images

Added an initial “is a leaf?” model that runs before the disease classifier.

Very common pattern, especially in applications like receipt OKR, or even credit card number identification.

Image Search Quality

Started with a question:

“Why are luxury car image searches producing poor user satisfaction scores?”

Image Search Quality - Debugging

Analyzing the logs revealed that “Jaguar” was particularly affected by this issue.

Visualizing the training data set in an embedding space showed that lots of photos of jungle cats (the other “jaguars”) were labelled as cars!

Looking at the tool used to tag images showed that the UI for “Jaguar (car)” versus “Jaguar (cat)” was easy to mistake, so labelers had been mislabeling.

Image Search Quality - Solution

Fixed the user interface for the labeling tool, and re-ran all “jaguar”-labeled images through it.

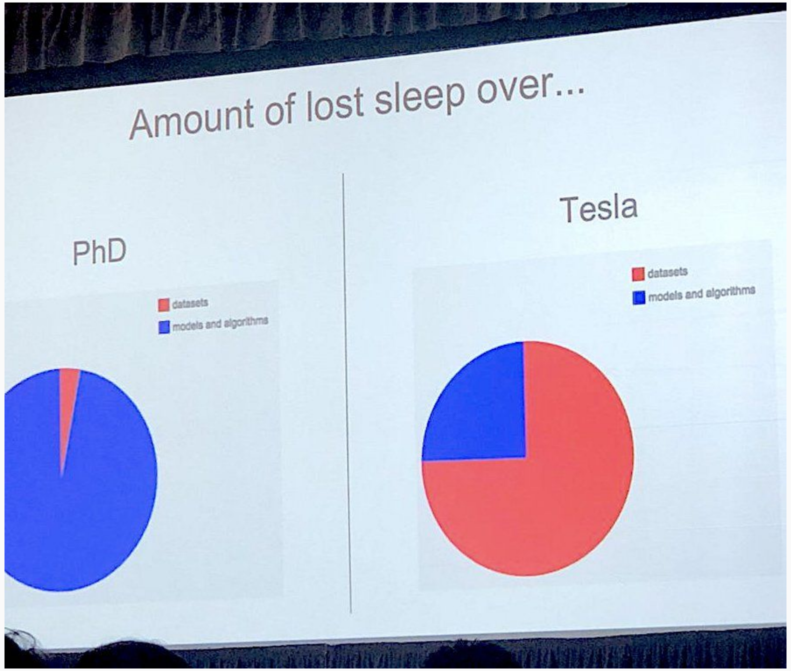
Common Challenges

Training Data beats Model Architecture

Andrej Karpathy:

(Blue is models and algorithms)

(Red is datasets)



Training Data beats Model Architecture

Few people will teach you how to improve your dataset, since academia doesn't have many resources to gather meaningful data.

Time spent on data gathering and labeling tools will be far more likely to help the success of your product than iterating on model architectures.

Pick as "standard" a model architecture as possible early on, and stick with it.

https://petewarden.com/2018/05/28/why-you-need-to-improve-your-training-data-and-how-to-do-it/

What do you do with your model’s results?

The outputs of your deep learning model need to trigger some kind of action or decision, in order to be useful.

The user’s needs are almost certainly not captured by your loss function and training evaluation set accuracy.

Figuring out how to make your ML product useful requires traditional application development skills.

TinyML

TinyML

I specialize in running machine learning models on embedded systems that might only have hundreds or even just tens of kilobytes of memory.

There are hundreds of billions of devices like this out there. ML can give them voice interfaces, spot sensor anomalies, even run image recognition. I’m very excited about a future of trillions of these devices, running off batteries or energy-harvesting for years!

See tinymlbook.com for more details (email me for a copy)

Server and Client-side Differences

Google Translate Camera

This is an application where you point your phone at a menu or sign in a foreign language, and it gives a translation.

There was a server-side version, where you took a photo, waited a few seconds for the server to respond, and then received the translation.

Word Lens was a startup that did something similar, but on the device itself.

Google Translate Camera

The Word Lens model was significantly less accurate than the server-side equivalent, at least according to training evaluation metrics.

Users rated the Word Lens results as more accurate and satisfying though! How come?

The very low latency of the on-device results gave feedback to users so they aligned the framing until everything “popped” into place. Waiting several seconds to get a poor translation was a lot worse.

The Good News

Machine learning is very similar on the server and client sides. Everything you're learning in the cloud will translate well to other deployment areas.

You might expect you'll have far more resources available in the cloud, but for most applications the economics mean that you only have a very small amount of compute available for each action. Servers cost companies money, but client-side compute doesn't.

Federated Learning

What is Federated Learning?

It’s a way of improving models using client-side data, and then sharing updates in a provably-private way, that doesn’t allow any central access to the client-side data.

It’s used in products like Google Keyboard to spot up-and-coming new words for suggestions, for example.

Federated Learning Challenges

It’s hard to explain to gatekeepers (like medical administrators) the privacy model.

Running training on-device is a lot less common than inference, and the software stack is a lot less mature.

Getting labels to go with the user data is hard, and verifying label quality is even harder.

FL is an advanced topic, and I recommend it be used as an incremental addition to a project.

Thanks!
Questions?

petewarden@google.com

@petewarden