From Black Box to a Known Quantity: 
How to Build Predictable, Reliable 
ML-based Services

{salim,villavieja}@google.com 
SREcon 2019
We are not machine learning hackers/ninjas.
We are not machine learning scientists.

We are experienced SREs and we have collected production insights through a large number of interviews (~40) from teams using ML in production at Google over the last 15 years.
From Black Box to a Known Quantity: How to Build Predictable, Reliable ML-based Services

{salim,villavieja}@google.com
SREcon 2019
ML Best practices & monitoring

{salim,villavieja}@google.com
SREcon 2019
Our goals

ML best practices

&

monitoring
Find the mistake about ML

- ML is easy
- ML is new
- ML is a black box, no need to know more
- Train “one and done”
- You rarely rollback

- ML monitoring is like any other monitoring
- More data is better
- Learn all the patterns
- Transparent to user
- Compatibility is a no-op
One Very Important ML model
At Google
Youtube ML for video recommendations

- Continuously training & Fast deployment
- Keep high accuracy
- World Wide input data
- Revenue facing
- More video time +
- More ad clicks
- Special events one day
- User can easily detect not accurate models
But it's not that easy in production

- Continuously training models
- Deploying every N hours / days / weeks
- Filtering spam/bad videos
- Monitor view time...
- Multiple devices
- Guarantee freshness
Theoretical Machine Learning Pipeline

Training Offline: (effort spent 10%)

Production data → Transform → Training (Compute) → Validation → Trained model

Serving Online: (effort spent 90%)

User-facing request → Transform → Serving → Prediction Classification → Serving model
Theoretical Machine Learning Pipeline

Training Online: (effort spent: 10%)

Serving Online: (effort spent: 90%)

DEPRECATED!

NOT RELIABLE

Production data

Transform

Training (Compute)

Validation

Trained model

Deployment

Serving model

User-facing request

Transform

Prediction Classification
BEST PRACTICES: Why are they important

Part 1: TRAINING & DATA QUALITY

Part 2: HARDWARE RESOURCES (GPU/TPU)

Part 3: QUALIFICATION

Part 4: BACKWARDS COMPATIBILITY/CONF. MANAGEMENT

- RELIABLE
- FAST
- PROD READY
- EASY
(re) Training

(not prototyping)
Training

Integral part of the release process

Not coding, debugging, testing
Input data coming to the training pipeline can't be stopped

Production changes fast:

**Model loss** increases with time at a constant rate.
Production changes fast:

**Model loss** increases with time at a constant rate.
Training

Production changes fast:

**Model loss** increases with time at a constant rate.
Training

Production changes fast:

**Model loss** increases with time at a constant rate.
Training

Production changes fast:

**Model loss** increases with time at a constant rate.
Training

Production changes fast:

**Model loss** increases with time at a constant rate.
Model Age for a popular Google Service

➔ Over 3 weeks, model never older than $X$
➔ Updates continuously
Model Age for a popular Google Service

Non-stop training
Models might need to evolve fast

➔ Over 3 weeks, model never older than $X$
➔ Updates continuously
Training: Filtering is key

- **Good:** Train your model with *all* data, from oldest to newest
- **Bad:** We can't *ALWAYS* train on all production data. (Youtube 1.2 TB ML model)
- Production data has *tons of duplicate information* and needs to be filtered.
- **Filtering:** collapse duplicate values, to construct the model efficiently.
- **Data Imputation:** replacing missing data with substituted values

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>carlos, male, 41, Spanish, 6.2, SRE, <strong>NULL</strong>, 80%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>salim, male, 44, American, 5.8, SRE, +4, 90%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>maria, female, 0, Norway, 6.0, SWE, +25, 60%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>fep, <em>agender</em>, Spanish, 6.0, SWE, +5, 75%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>maria, female, 0, Norway, 6.0, SWE, +25, 60%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Training: Data size

- Validation data is not the same as trained data
  - Trust that your high-accuracy model is correct with data not used during training.
    - 80-20/70-30 might vary depending on the model
    - Randomly selected set from the trained data
- Do not confuse with qualification (to be seen later)

No "one size fits all"

Image source: https://vimeo.com/122534562
License: https://creativecommons.org/licenses/by/3.0/
Training at Scale

- Very large data sets.
- How many models are continuously training (batch)?
  - Different regions? Different time zones?
  - Available compute resources might be an issue.
- Snapshot your model:
  - Warm start on training
  - Avoid losing time if scheduled out

Image source: [https://vimeo.com/122534562](https://vimeo.com/122534562)
License: [https://creativecommons.org/licenses/by/3.0/](https://creativecommons.org/licenses/by/3.0/)
### Summary: Data Quality on Training

#### CORRECT

- **Data imputation and data validation so that your models never receive unexpected inputs.**

#### COMPLETE

- **Missing inputs previously used**

#### SNAPSHOTs

- Train over previous models (resuming and rollback)

#### BIAS

- Monitor amount of data from different sources. Features skews (train features diff from inference features)

#### ANOMALIES

- Can't train with all SuperBowl day/New Years

#### DATA RATIOS

- Continent X pipeline stopped and the youtube recommendation models stops taking into account those videos.

#### AUTOMATION

- Be ready to add fields on old data.
- Be ready to fix your data (spam data in trained models)
Before Machine Learning Pipeline

Training (time spent 10%)

Production data → Transform → Training (Compute) → Validation → Trained model

Offline
After Machine Learning Pipeline

Training

Production data → Data Imputation → Filtered data → Data Quality

Training (Compute) → Validation → Trained model

Offline
Why hardware resources are important

- Two different & disjoint environments
  - Training
  - Serving/Inference
- Cost of Training resources grows at a higher rate than Production resources
Qualification
Model Qualification

- Models are qualified with a separate input data
  - How is this data chosen? (previous or same prod day)
- Models are tested with the same production binary.
- Or we have an A/B testing scenario
  - Same production code/release
  - Dynamically decide % predictions to each model
Canary is a must

Even with a portion of global input, canary behaves like prod
Model Qualification

The model is signed post qualification.

- Some providers allow to register models for versioning
- Signature specifies type of model, input/output data

Only allow *signed* models in production.
Backwards Compatibility
Backwards Compatibility

- **Input data changes:**
  - New fields, new values, null/empty values not contemplated.

- **API changes:**
  - Tensorflow API changes frequently, Incompatible model
  - "The model was completely valid and healthy as configured, it was simply not configured for the type of traffic it would receive"

- **Fallback mechanisms, when rollback not an option:**
  - Most teams do not have a non-ML fallback mechanisms
  - What happens if you run out of quota/capacity.

- **Old models need to be deprecated, they might not be reusable**
  - New inputs (labels) deployed
  - Signing the models helps on this. However, we're not able to ask the model compatibility?
Three important (config) questions

- Do code and models go together? Are they deployed in the same package?
- How do we verify model and code compatibility?
  - *Always* push through canary
- What API version is this model for?
What's an ML pipeline like?
Monitor training phase as well as live serving
- Monitor Prediction latency
  - Consider TF/RPC overhead
- Monitor Model aging
  - Accuracy loss through model aging
What goes wrong when you don’t have alerting?

How can you identify this change in behavior?

Alerting must be domain-specific
Training

Production changes fast:

**Model loss** increases with time at a constant rate.
Conclusions

Train continuously
Add filtering
Data imputation
Stamp new models...
... Deprecate old models
Use domain-specific alerting
The Future of ML in Production

- Open source available training data sets
  - Already anonymized + No need to delete user data
- Models as a Service
  - Credit card/Image recognition/Text to Speech as unique APIs
With thanks to

very many of our colleagues across Alphabet: DeepMind, Google, YouTube

And to our colleagues at
Clarifai
ThoughtWorks
That’s all.

Questions? Comments?

{salim,villavieja}@google.com