Putting Deep Learning Models in Production

Sahil Dua
Let’s imagine!
But ...
whoami

➔ Software Developer @ Booking.com
➔ Previously - Deep Learning Infrastructure
➔ Open Source Contributor (Git, Pandas, Kinto, go-github, etc.)
➔ Tech Speaker
Agenda

➔ Deep Learning at Booking.com
➔ Life-cycle of a model
➔ Training Models
➔ Serving Predictions
Deep Learning at Booking.com
Scale highlights.

- 1,500,000+ room nights booked every 24 hours
- 1.4 million+ active properties in 220+ countries
Deep Learning

➔ Image understanding
➔ Translations
➔ Ads bidding
➔ ...

@sahildua2305
Booking.com
Image Tagging
Image Tagging

<table>
<thead>
<tr>
<th>Classes</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>oceanfront</td>
<td>0.79</td>
</tr>
<tr>
<td>nature</td>
<td>0.79</td>
</tr>
<tr>
<td>beach house</td>
<td>0.62</td>
</tr>
<tr>
<td>building</td>
<td>0.62</td>
</tr>
<tr>
<td>penthouse</td>
<td>0.61</td>
</tr>
<tr>
<td>apartment</td>
<td>0.61</td>
</tr>
<tr>
<td>housing</td>
<td>0.61</td>
</tr>
</tbody>
</table>
### Image Tagging

<table>
<thead>
<tr>
<th>Classes</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>oceanfront</td>
<td>0.79</td>
</tr>
<tr>
<td>nature</td>
<td>0.79</td>
</tr>
<tr>
<td>beach house</td>
<td>0.62</td>
</tr>
<tr>
<td>building</td>
<td>0.62</td>
</tr>
<tr>
<td>penthouse</td>
<td>0.61</td>
</tr>
<tr>
<td>apartment</td>
<td>0.61</td>
</tr>
<tr>
<td>housing</td>
<td>0.61</td>
</tr>
</tbody>
</table>

- **Sea view:** 6.38
- **Balcony/Terrace:** 4.82
- **Photo of the whole room:** 4.21
- **Bed:** 3.47
- **Decorative details:** 3.15
- **Seating area:** 2.70

*Image courtesy of Booking.com*
Image Tagging

Using the image tag information in the right context
Swimming pool, Breakfast Buffet, etc.
Lifecycle of a model
Lifecycle of a model

1. Data Analysis
2. Train
3. Deploy
Training a Model - on laptop
Training a Model - on laptop

@sahildua2305
Machine Learning workload

➔ Computationally intensive workload

➔ Often not highly parallelizable algorithms

➔ 10 to 100 GBs of data
Why Kubernetes (k8s)?

➔ Isolation
➔ Elasticity
➔ Flexibility
Why k8s – GPUs?

➔ In alpha since 1.3

➔ Speed up 20X-50X

resources:
  limits:
    alpha.kubernetes.io/nvidia-gpu: 1
Training with k8s

➔ Base images with ML frameworks
  ◆ TensorFlow, Torch, VowpalWabbit, etc.

➔ Training code is installed at start time

➔ Data access - Hadoop (or PVs)
Startup

Training pod

Code

.. start.sh
train.py
evaluate.py

git
Startup

Training pod

```
start.sh
train.py
evaluate.py
```

Data
Streaming logs back

Training pod

.. start.sh
train.py
evaluate.py

Logs

PV
Exports the model

Training pod

.. start.sh
train.py
evaluate.py

PV

model

.. start.sh
train.py
evaluate.py
Serving predictions
Serving Predictions

Client

Model

Input
Features

Prediction
Serving Predictions

Model 1

Model X

Client

Client

Input Features

Prediction

Prediction

@sahildua2305
Serving Predictions

Client

Input Features
Prediction

Model 1

Model X
Serving Predictions

- Stateless app with common code
- Containerized
- No model in image
- REST API for predictions
Serving Predictions

Client → Input Features → App → Prediction → Database
Serving Predictions

➔ Get trained model from Hadoop
➔ Load model in memory
➔ Warm it up
➔ Expose HTTP API
➔ Respond to the probes
Serving Predictions

Input
Features

Prediction
Serving Predictions

Client

Input Features → Prediction

Features

Prediction

@sahildua2305
Deploying a new model

➔ Create new Deployment
➔ Create new HTTP Route
➔ Wait for liveness/readiness probe
Performance

\[
\text{PredictionTime} = \text{RequestOverhead} + N \times \text{ComputationTime}
\]

\(N\) is the number of instances to predict on
Optimizing for Latency

➔ Do not predict if you can precompute
➔ Reduce Request Overhead
➔ Predict for one instance
➔ Quantization (float 32 => fixed 8)
➔ TensorFlow specific: freeze network & optimize for inference
Optimizing for Throughput

➔ Do not predict if you can precompute
➔ Batch requests
➔ Parallelize requests
Summary

➔ Training models in pods
➔ Serving models
➔ Optimizing serving for latency/throughput
Next steps

➔ Tooling to control hundred deployments
➔ Autoscale prediction service
➔ Hyper parameter tuning for training
Want to get in touch?

LinkedIn / Twitter / GitHub
@sahildua2305

Website
www.sahildua.com
THANK YOU

@sahildua2305