Floor 28, Tel Aviv, July 7th, 2019

Optimize your Machine Learning workloads on AWS

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon

Agenda

- Infrastructure

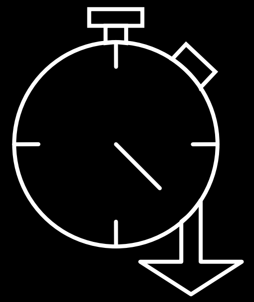
- Training
  - Easy tips to speed up the training process
  - Automatic model tuning

- Inference
  - Model compilation: Amazon SageMaker Neo
  - Cost optimization: Amazon Elastic Inference

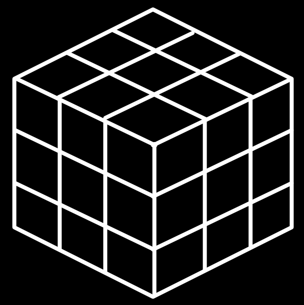
Optimizing infrastructure

Amazon EC2 P3dn

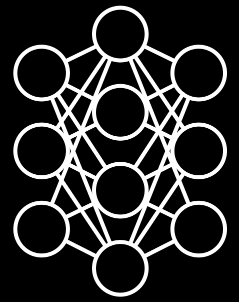
https://aws.amazon.com/blogs/aws/new-ec2-p3dn-gpu-instances-with-100-gbps-networking-local-nvme-storage-for-faster-machine-learning-p3-price-reduction/



Reduce machine learning training time



Better GPU utilization



Support larger, more complex models

KEY FEATURES

100Gbps of networking bandwidth

8 NVIDIA Tesla V100 GPUs

32GB of memory per GPU (2x more)

96 Intel Skylake vCPUs (50% more than P3) with AVX-512

Amazon EC2 C5n

https://aws.amazon.com/blogs/aws/new-c5n-instances-with-100-gbps-networking/

Intel Xeon Platinum 8000

Up to 3.5GHz single core speed

Up to 100Gbit networking

Based on Nitro hypervisor for bare metal-like performance

Instance Name	vCPUs	RAM	EBS Bandwidth	Network Bandwidth
c5n.large	2	5.25 GiB	Up to 3.5 Gbps	Up to 25 Gbps
c5n.xlarge	4	10.5 GiB	Up to 3.5 Gbps	Up to 25 Gbps
c5n.2xlarge	8	21 GiB	Up to 3.5 Gbps	Up to 25 Gbps
c5n.4xlarge	16	42 GiB	3.5 Gbps	Up to 25 Gbps
c5n.9xlarge	36	96 GiB	7 Gbps	50 Gbps
c5n.18xlarge	72	192 GiB	14 Gbps	100 Gbps

Tips to speed up the training process

Tips to speed up training

- Scale out with distributed training
- Pick the best format for your dataset
  - Use protobuf instead of CSV or JSON
  - Pack samples into record-based files
    - TFRecord (Tensorflow) or RecordIO (MXNet)
    - Splitting in 100MB files looks like the sweet spot
  - Amazon SageMaker: protobuf-encoded RecordIO
- Use Pipe Mode for large datasets
  - Stream directly from Amazon S3, don’t copy
  - Train on arbitrary large datasets
- Monitor CPU/GPU usage in Amazon CloudWatch
  - Use the largest batch size that makes sense for your dataset
  - Multiply batch size by the number of GPUs on the instance

Automatic Model Tuning

Examples of hyperparameters

XGBoost

Tree depth
Max leaf nodes
Gamma
Eta
Lambda
Alpha
…

Neural Networks

Number of layers
Hidden layer width
Learning rate
Embedding dimensions
Dropout
…

Tactics to find the optimal set of hyperparameters

Finding the optimal set of hyper parameters

1. Manual Search ("I know what I'm doing")
2. Grid Search ("X marks the spot")
   Typically training hundreds of models
   Slow and expensive
3. Random Search (Bengio 2012) Spray and p
   Works better and faster than Grid Search
   But… but… but… it's random!
4. Hyperparameter Optimization: use Machin
   Training fewer models
   Gaussian Process Regression and Bayesian Optimization

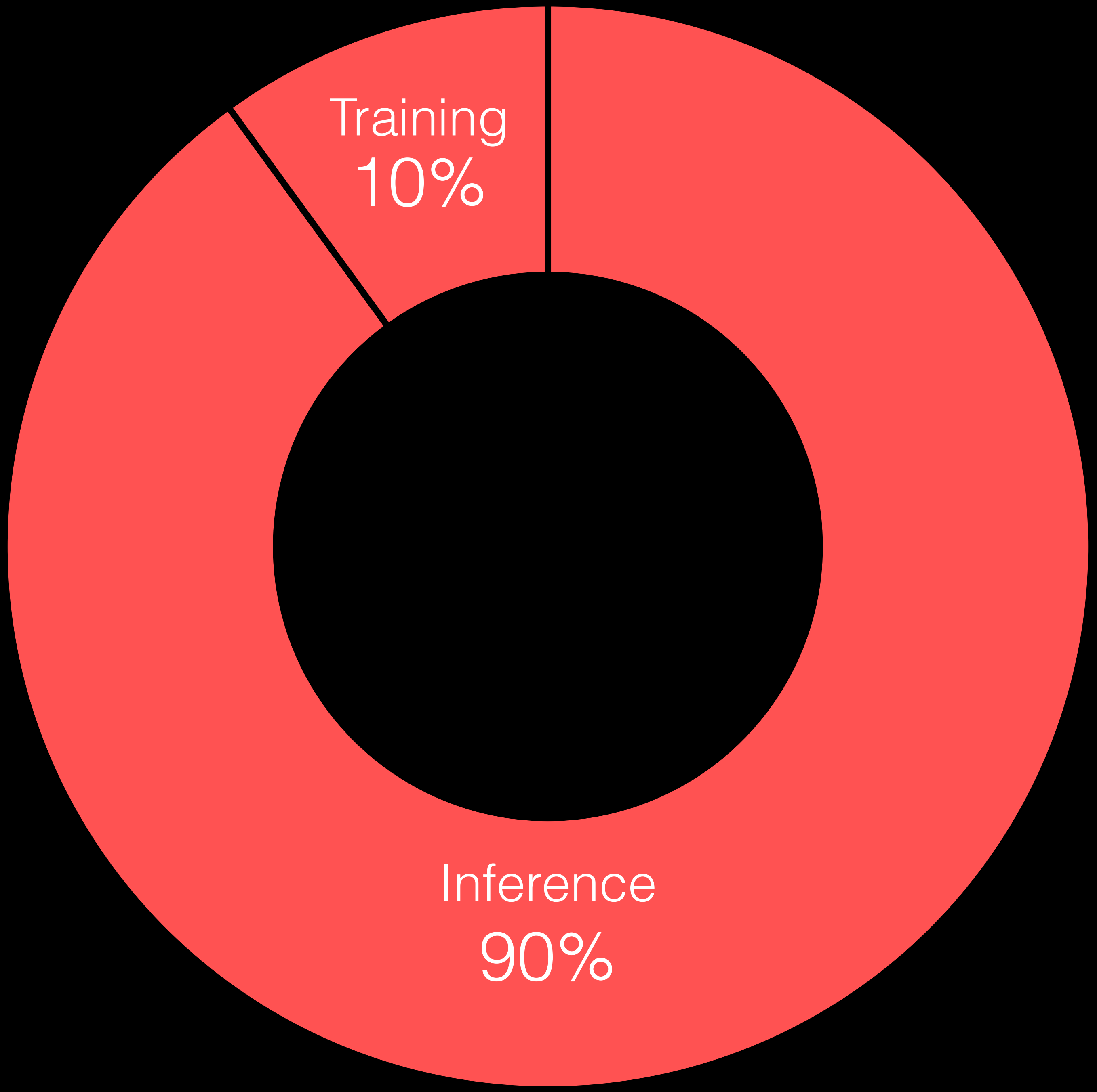
Demo:

HPO with Keras

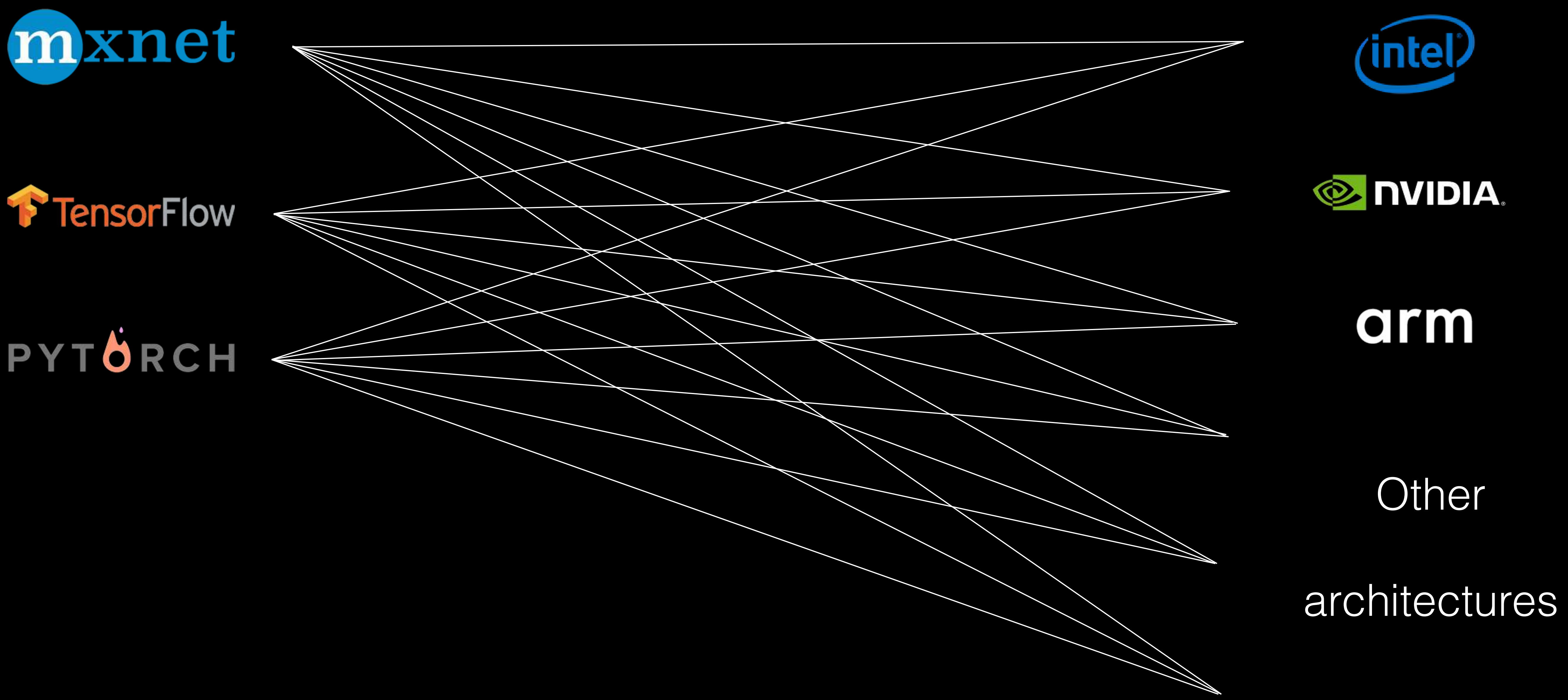
https://gitlab.com/juliensimon/dlnotebooks/tree/master/keras/04-fashion-mnist-sagemaker-advanced

Model compilation with Amazon SageMaker Neo

Predictions drive complexity and cost in production



Model optimization is extremely complex



Amazon SageMaker Neo

Train once, run anywhere with 2x the performance

Get accuracy and performance

Automatic optimization

Broad framework support

Broad hardware support

KEY FEATURES

Integrated with Amazon EC2 and Amazon SageMaker

Free of charge!

Open-source runtime and compiler; 1/10th the size of original frameworks github.com/neo-ai

Compiling ResNet-50 for the Raspberry Pi

Configure the compilation job

```
{
  "RoleArn":$ROLE_ARN,
  "InputConfig": {
   "S3Uri":"s3://jsimon-neo/model.tar.gz",
   "DataInputConfig": "{\"data\": [1, 3, 224, 224]}",
   "Framework": "MXNET"
  },
 "OutputConfig": {
  "S3OutputLocation": "s3://jsimon-neo/",
  "TargetDevice": "rasp3b"
 },
 "StoppingCondition": {
 "MaxRuntimeInSeconds": 300
 }
}
```

Compile the model

```
$ aws sagemaker create-compilation-job
--cli-input-json file://config.json
--compilation-job-name resnet50-mxnet-pi

$ aws s3 cp s3://jsimon-neo/model-
rasp3b.tar.gz .

$ gtar tfz model-rasp3b.tar.gz
compiled.params
compiled_model.json
compiled.so
```

Predict with the compiled model

```
from dlr import DLRModel
model = DLRModel('resnet50', input_shape,
output_shape, device)
out = model.run(input_data)
```

Demo:

Neo on SageMaker

Cost optimization with Amazon Elastic Inference

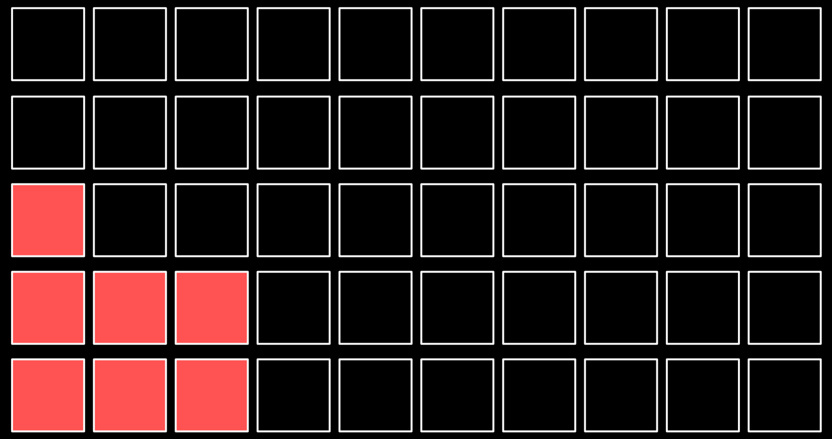
Right-sizing your inference infrastructure

- Statistical ML models, small DL models, and of course dev/test infrastructure: CPU instances (C5) deliver the best cost/performance ratio

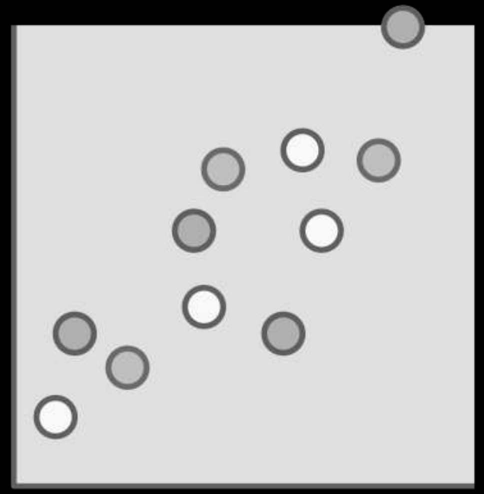
- Very large DL models
  - GPU instances (P2 or P3) should work best, especially if you need high throughput
  - If not, C5n could be a reasonable alternative

- But what about everything in between?
  - Mid-sized models
  - NLP models
  - Low throughput, low latency workloads
  - « Too slow on CPU, too expensive on GPU » ?

Are you making the most of your GPU infrastructure?



Low utilization and high costs



One size does not fit all

Amazon Elastic Inference

Reduce deep learning inference costs up to 75%

Lower inference costs

Match capacity to demand

Available between 1 to 32 TFLOPS per accelerator

KEY FEATURES

Integrated with Amazon EC2 and Amazon SageMaker

Support for TensorFlow and Apache MXNet

Single and mixed-precision operations

Demo:

Elastic Inference on Amazon SageMaker

Getting started

http://aws.amazon.com/free

https://ml.aws
https://aws.amazon.com/sagemaker

https://github.com/aws/sagemaker-python-sdk
https://github.com/awslabs/amazon-sagemaker-examples

https://medium.com/@julsimon
https://gitlab.com/juliensimon/dlnotebooks

Merci!

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon