INNOVATE

ONLINE CONFERENCE

MACHINE LE I

AND AI EDITI

AWS INNOVATE
ONLINE CONFERENCE
MACHINE LE
AND AI EDITI

Deep Dive on Amazon SageMaker

Julien Simon
Global Evangelist, AI & Machine Learning
Amazon Web Services

@julsimon
https://medium.com/@julsimon

Agenda

1. Storage
   - Amazon S3 & Pipe mode
   - Amazon EFS NEW!
   - Amazon FSx for Lustre NEW!
2. Training
   - Distributed Training
   - Managed Spot Training NEW!
3. Model tuning
4. Deployment
   - Model compilation: Amazon SageMaker Neo
   - Cost optimization: Amazon Elastic Inference

Storage

Passing datasets to algorithms

- Amazon SageMaker algorithms accept input data from channels
  - A channel is a named input source defining a dataset
  - At least one, up to twenty: Training, validation, test, etc.
- `Channel` object
  - Name
  - Data source: `S3DataSource` or `FileSystemDataSource`
  - Data format: CSV, RecordIO, etc.
  - Compression type
  - Input mode: `File` or `Pipe` (S3 only)
- The list of channels is passed to `CreateTrainingJob`
- SageMaker Python SDK: `Estimator.fit()` receives a dictionary of
  - `sagemaker.inputs.s3_input` for S3
  - `sagemaker.inputs.FileSystemInput` for EFS/FSx

Storing your dataset in Amazon S3

- Simplest option
  - `sagemaker.session.default_bucket()`, `sagemaker.session.upload_data()`
- [`S3DataSource`]
  - Location: URI
  - Type: Prefix, manifest, augmented manifest
  - Distribution: Fully replicated (training instances receive the full dataset), or sharded (1/nth of the dataset)
- Input mode: `File` or `Pipe`?
  - File: Copy the dataset to each training instance (full or 1/nth)
  - Pipe: Stream directly from S3
    - Training starts faster and runs faster
    - No need to provision lots of storage on training instances
    - Train on arbitrary large datasets, as they don't need to be fully stored or loaded in RAM any longer
- Pipe mode is supported by most built-in algorithms and can be implemented in TensorFlow, Apache MXNet, etc.
  - https://aws.amazon.com/blogs/machine-learning/using-pipe-input-mode-for-amazon-sagemaker-algorithms/
  - https://aws.amazon.com/blogs/machine-learning/accelerate-model-training-using-faster-pipe-mode-on-amazon-sagemaker/

Demo:

A quick look at Pipe Mode with TensorFlow

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker-python-sdk/tensorflow_script_mode_pipe_mode

Storing your dataset in Amazon EFS

https://aws.amazon.com/blogs/machine-learning/speed-up-training-on-amazon-sagemaker-using-amazon-efs-or-amazon-fsx-for-lustre-file-systems/

- If your organization is sharing data over NFS, this is a good option
  - Shared datasets, notebooks, etc.
  - Train directly from EFS, no data movement required

- Training instances must run in a VPC, and open port 2049 (NFS)

- FileSystemDataSource
  - Filesystem id: provided by EFS
  - Type: 'EFS'
  - Directory path
  - Access type: Read-only or read-write

```
estimator = TensorFlow(
          entry_point='tensorflow_mnist/mnist.py',
          role='SageMakerRole',
          train_instance_count=1,
          train_instance_type='ml.c4.xlarge',
          subnets=['subnet-1', 'subnet-2'],
          security_group_ids=['sg-1'])

file_system_input = FileSystemInput(
          file_system_id='fs-1',
file_system_type='EFS',
          directory_path='/tensorflow',
file_system_access_mode='ro')

estimator.fit(inputs=file_system_input)
```

Amazon FSx for Lustre

Fully managed Lustre file system for compute-intensive workloads

https://aws.amazon.com/fsx/lustre/

Massively scalable performance

Seamless access to your data repositories

Simple and fully managed

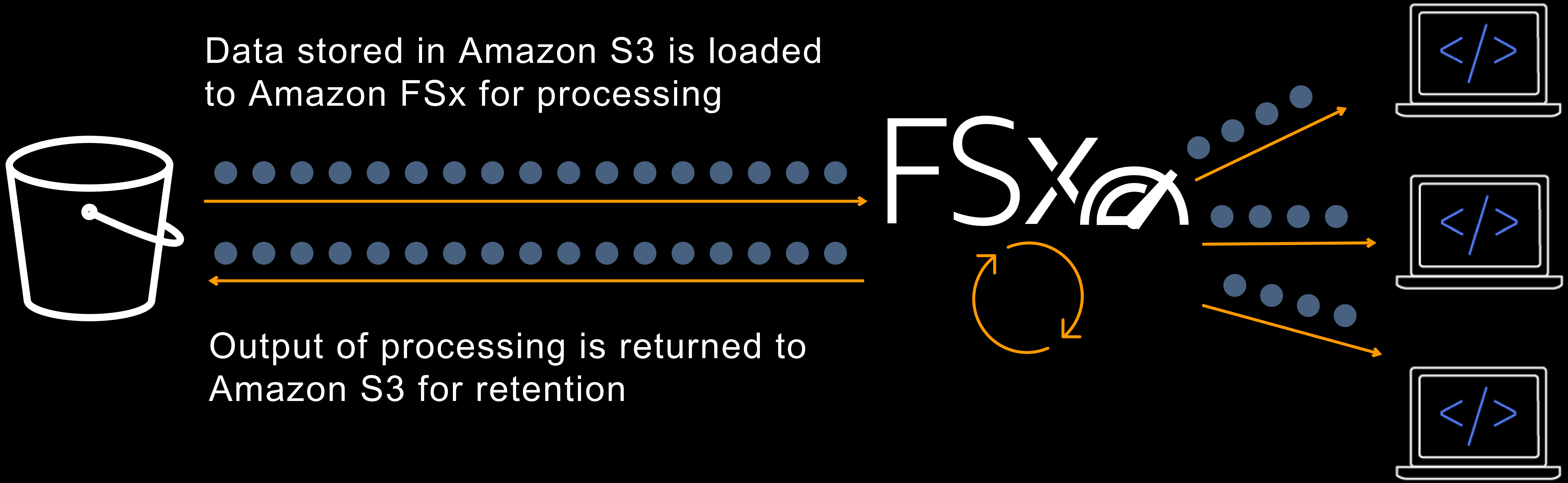
Native file system interface

Cost-optimized for compute-intensive workloads

Secure and compliant

Seamless integration with Amazon S3

Link your Amazon S3 dataset to your Amazon FSx for Lustre file system, then…



When your workload finishes, simply delete your file system

Storing your dataset in Amazon FSx for Lustre

https://aws.amazon.com/blogs/machine-learning/speed-up-training-on-amazon-sagemaker-using-amazon-efs-or-amazon-fsx-for-lustre-file-systems/

- Best option for high-performance, low-latency training
- Create an FSx file system, link it to your S3 bucket, train
- Delete the file system when you're done

- Training instances must run in a VPC, and open port 998 (Lustre)

- FileSystemDataSource
  - Filesystem id: provided by FSx
  - Type: 'FSxLustre'
  - Directory path
  - Access type: Read-only or read-write

```
estimator = TensorFlow(
          entry_point='tensorflow_mnist/mnist.py',
          role='SageMakerRole',
          train_instance_count=1,
          train_instance_type='ml.c4.xlarge',
          subnets=['subnet-1', 'subnet-2'],
          security_group_ids=['sg-1'])

file_system_input = FileSystemInput(
          file_system_id='fs-2',
file_system_type='FSxLustre',
          directory_path='/fsx/tensorflow',
file_system_access_mode='ro')

estimator.fit(inputs=file_system_input)
```

Training

Distributed Training

- Natively available for most built-in algorithms
- Natively available for TensorFlow, Apache MXNet, etc.
- You need to implement it yourself if you use a custom container

- Zoom on TensorFlow: Two modes available
  - Parameter Server
    - Asynchronous gradient averaging and weight distribution
    - All instances talk to each other: Networking can become a bottleneck and slow down training
  - Horovod
    - Based on Ring-AllReduce algorithm
    - More efficient communication helps scale near-linearly to 256 GPUs
    - https://github.com/aws-samples/sagemaker-horovod-distributed-training

Demo:

A quick look at TensorFlow with Horovod

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/sagemaker-python-sdk/tensorflow_script_mode_horovod

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

Managed Spot Training

https://aws.amazon.com/blogs/aws/managed-spot-training-save-up-to-90-on-your-amazon-sagemaker-training-jobs/

- Save up to 90% on training costs
- Fully managed: Obtain spot instances, start training, handle interruptions
- Implement checkpointing to resume interrupted jobs
  - Available in built-in algorithms for computer vision
  - Default behavior in TensorFlow
  - If checkpointing is not implemented,
    the training job is restarted from scratch
  - You get billed for data download only once
- `CreateTrainingJob`
  - `EnableManagedSpotTraining = true`
    `MaxWaitTimeInSeconds`
    `= MaxRuntimeInSeconds`
      + time waiting for spot instances

```
estimator = TensorFlow(
        entry_point='script.py',
        role='SageMakerRole',
        train_instance_count=1,
        train_instance_type='ml.p3.2xlarge',
        train_use_spot_instances=true,
        train_max_run=3600,
        train_max_wait=7200)

estimator.fit(inputs)
```

Demo:

Fashion-MNIST classification with Keras/TensorFlow

+ Script Mode
+ Managed Spot Training
+ Elastic Inference

https://aws.amazon.com/blogs/machine-learning/train-and-deploy-keras-models-with-tensorflow-and-apache-mxnet-on-amazon-sagemaker/

https://gitlab.com/juliensimon/dlnotebooks/tree/master/keras/05-keras-blog-post

Tips to speed up training

- Scale out with distributed training

- Pick the best format for your dataset
  - Use protobuf instead of CSV or JSON
    - https://github.com/aws/sagemaker-python-sdk/blob/master/src/sagemaker/amazon/common.py
  - Pack samples into record-based files
    - TFRecord (TensorFlow) or RecordIO (MXNet)
    - Splitting in 100MB files looks like the sweet spot
  - Protobuf-encoded + RecordIO ❤

- Amazon S3: Use Pipe mode for large datasets NEW!

- Monitor CPU/GPU usage and network throughput in Amazon CloudWatch

Model tuning

The never-ending quest for hyperparameters

XGBoost

Tree depth
Max leaf nodes
Gamma
Eta
Lambda
Alpha
…

Neural networks

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

Finding the optimal set of hyperparameters

https://aws.amazon.com/blogs/machine-learning/amazon-sagemaker-automatic-model-tuning-now-supports-random-search-and-hyperparameter-scaling/

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**: "Spray and pray"
   - Works better and faster than Grid Search
   - But… but… but… it's random!

4. **Hyperparameter optimization** (HPO): Use machine learning
   - Requires fewer training jobs
   - Gaussian Process Regression and Bayesian Optimization

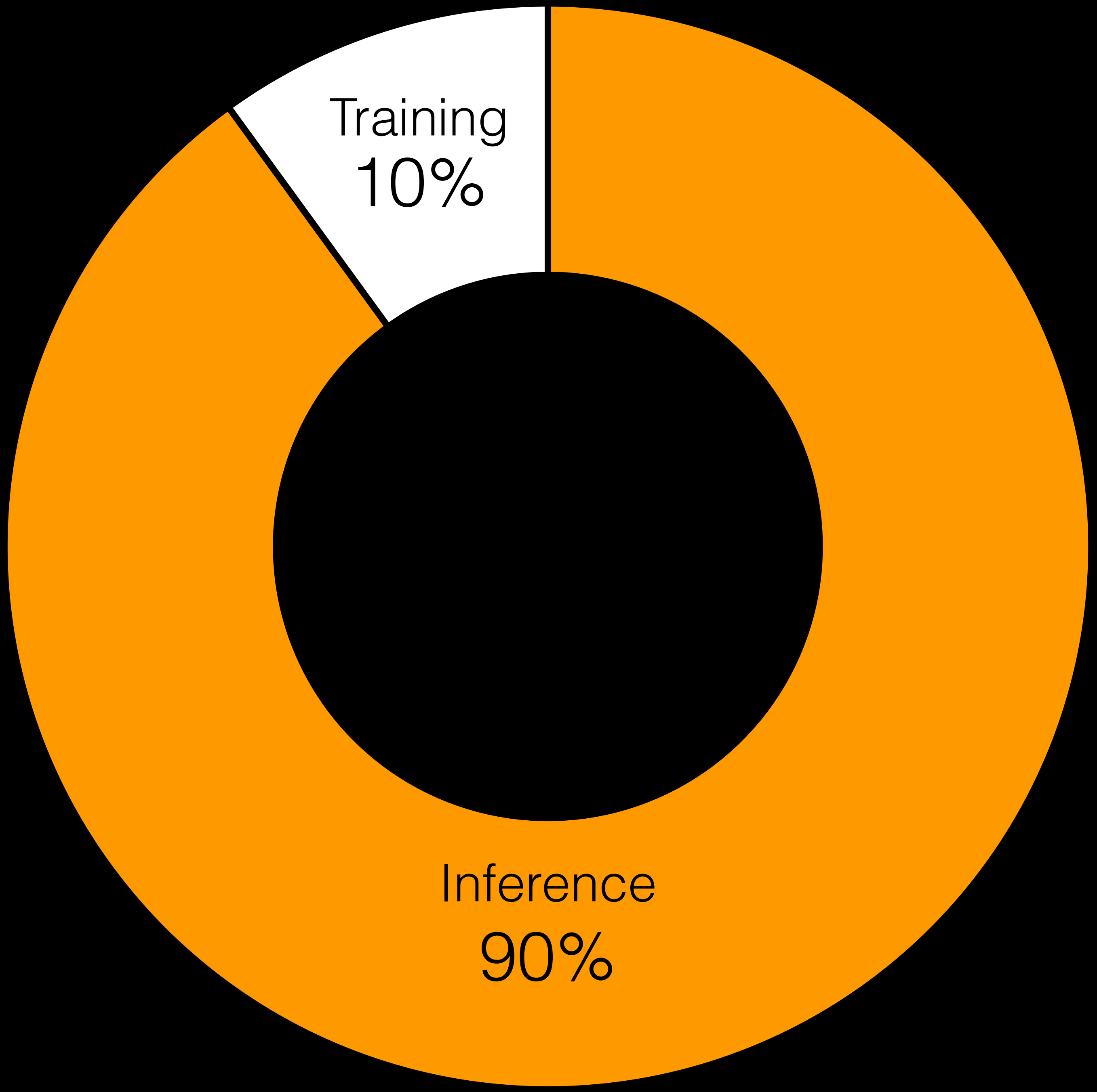
Demo:

HPO with Keras

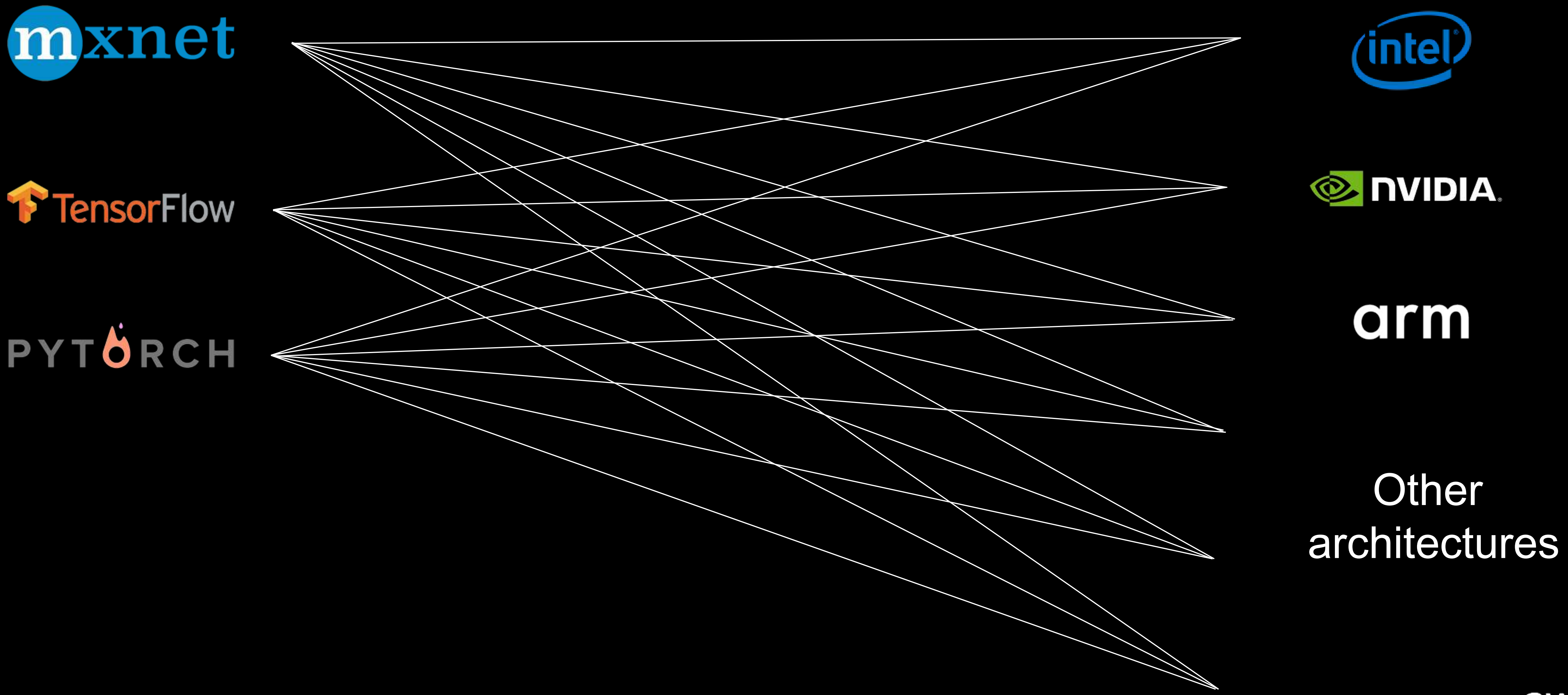
https://gitlab.com/juliensimon/dlnotebooks/tree/master/keras/04-fashion-mnist-sagemaker-advanced

Deployment

Predictions drive complexity and cost in production



Model optimization is extremely complex



Amazon SageMaker Neo

https://aws.amazon.com/blogs/aws/amazon-sagemaker-neo-train-your-machine-learning-models-once-run-them-anywhere/

mxnet
TensorFlow
PYTORCH

intel
NVIDIA
arm

Get accuracy and performance

Automatic optimization

Broad framework support

Broad hardware support

KEY FEATURES

Integrated with Amazon EC2 and Amazon SageMaker

Open-source runtime and compiler; 1/10th the size of original frameworks

https://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
```

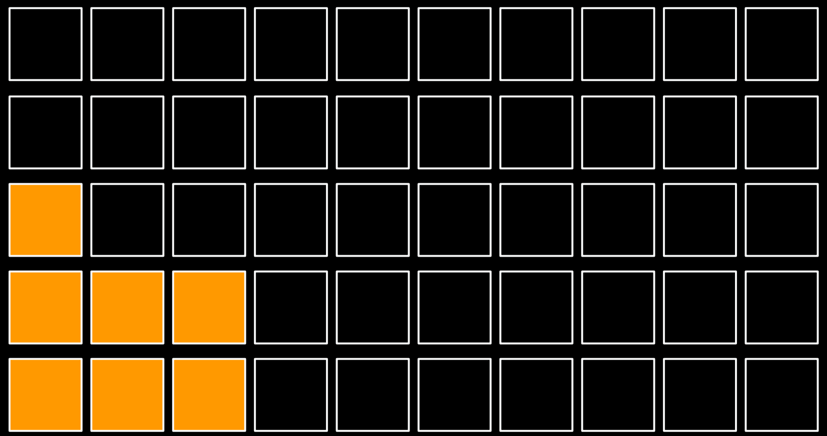
Predict with the compiled model

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

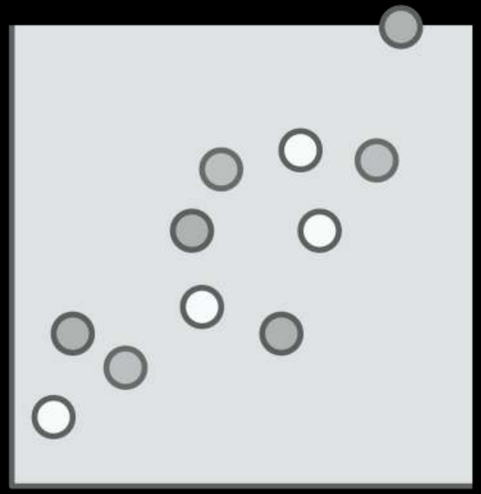
Right-sizing your inference infrastructure

- Statistical ML models, small DL models, dev/test
  - 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, not cost-effective 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

https://aws.amazon.com/blogs/aws/amazon-elastic-inference-gpu-powered-deep-learning-inference-acceleration/

Reduce GPU inference costs up to 75%

Match capacity to demand

Available between 1 and 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:

Fashion-MNIST classification with Keras/TensorFlow

+ Script Mode
+ Managed Spot Training
+ Elastic Inference

https://aws.amazon.com/blogs/machine-learning/train-and-deploy-keras-models-with-tensorflow-and-apache-mxnet-on-amazon-sagemaker/

https://gitlab.com/juliensimon/dlnotebooks/tree/master/keras/05-keras-blog-post

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/aws/sagemaker-spark

https://github.com/awslabs/amazon-sagemaker-examples

https://gitlab.com/juliensimon/dlnotebooks

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning
Amazon Web Services

@julsimon
https://medium.com/@julsimon