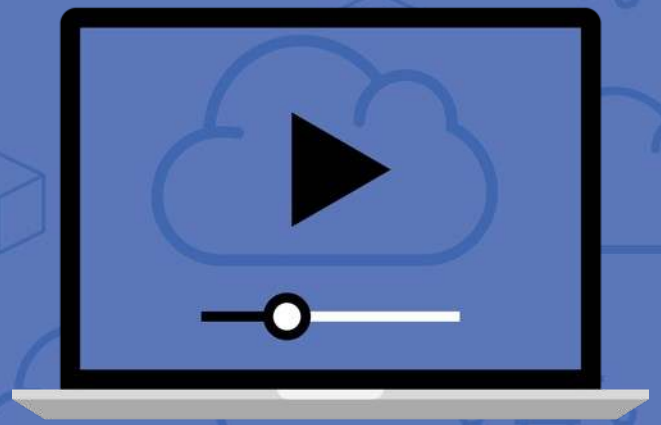
# Processing images with Deep Learning



**Julien Simon, AI Evangelist, EMEA**
**@julsimon**

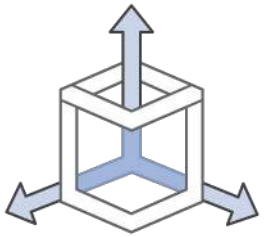
# What to expect

- Amazon Rekognition or Apache MXNet?
- Github projects for image processing with Apache MXNet
- A deeper look at the Convolution operation
- Demos
- Q&A

# Apache MXNet: Open Source library for Deep Learning

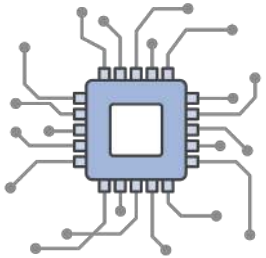
## Programmable

Simple syntax, multiple languages



## Portable

Highly efficient models for mobile and IoT

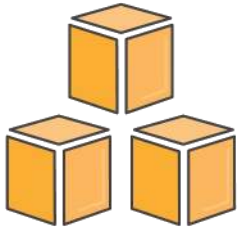


## High Performance

Near linear scaling across hundreds of GPUs

## Most Open

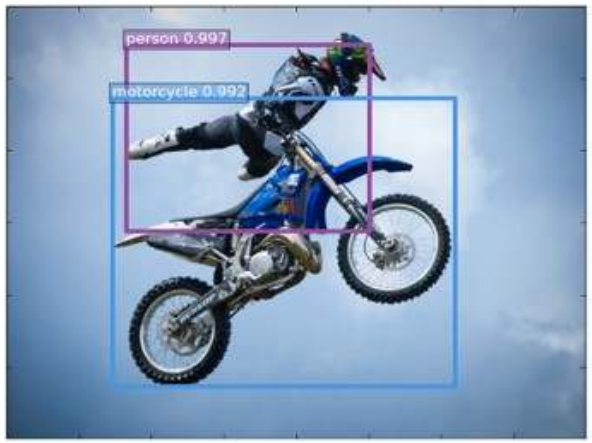
Accepted into the Apache Incubator

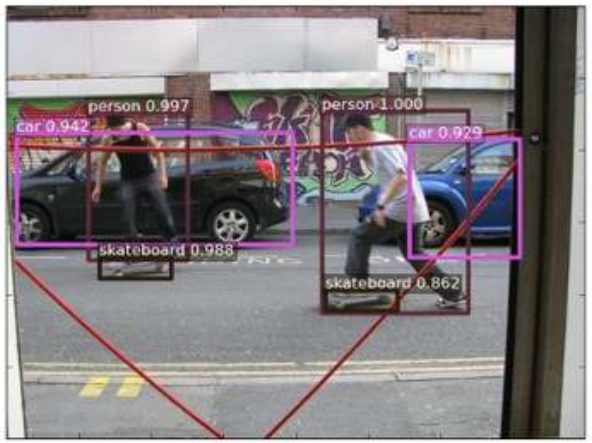


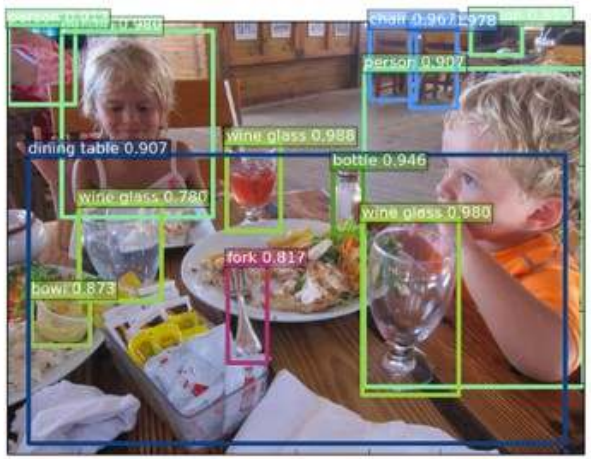
## Best On AWS

Optimized for Deep Learning on AWS

# Object Detection



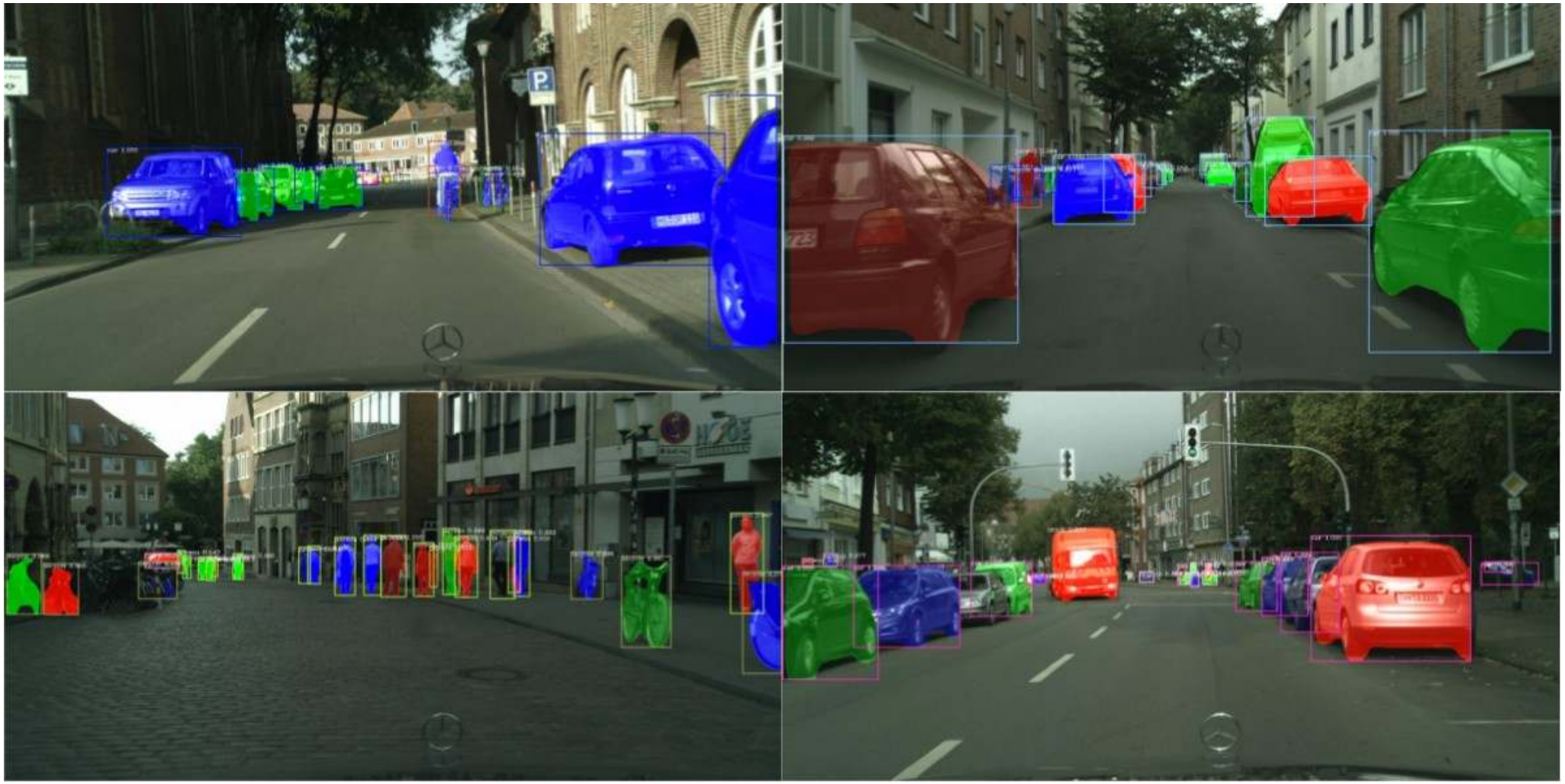






https://github.com/precedenceguo/mx-rcnn

https://github.com/zhreshold/mxnet-yolo

# Object Segmentation

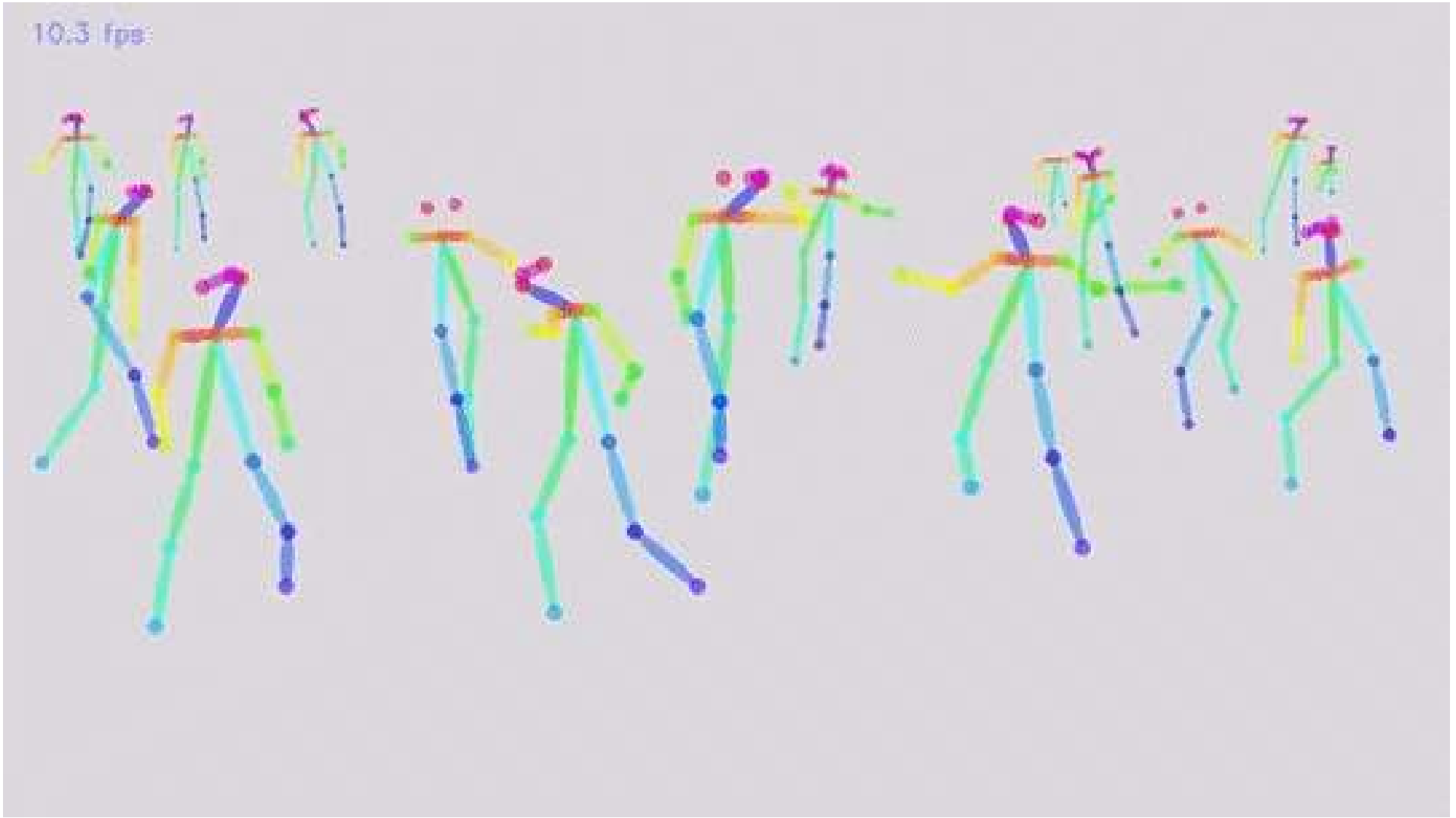


https://github.com/TuSimple/mx-maskrcnn

# Text Detection and Recognition

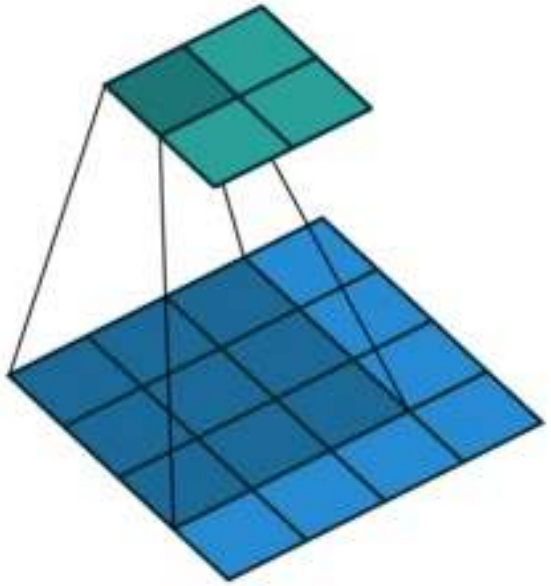

https://github.com/Bartzi/stn-ocr

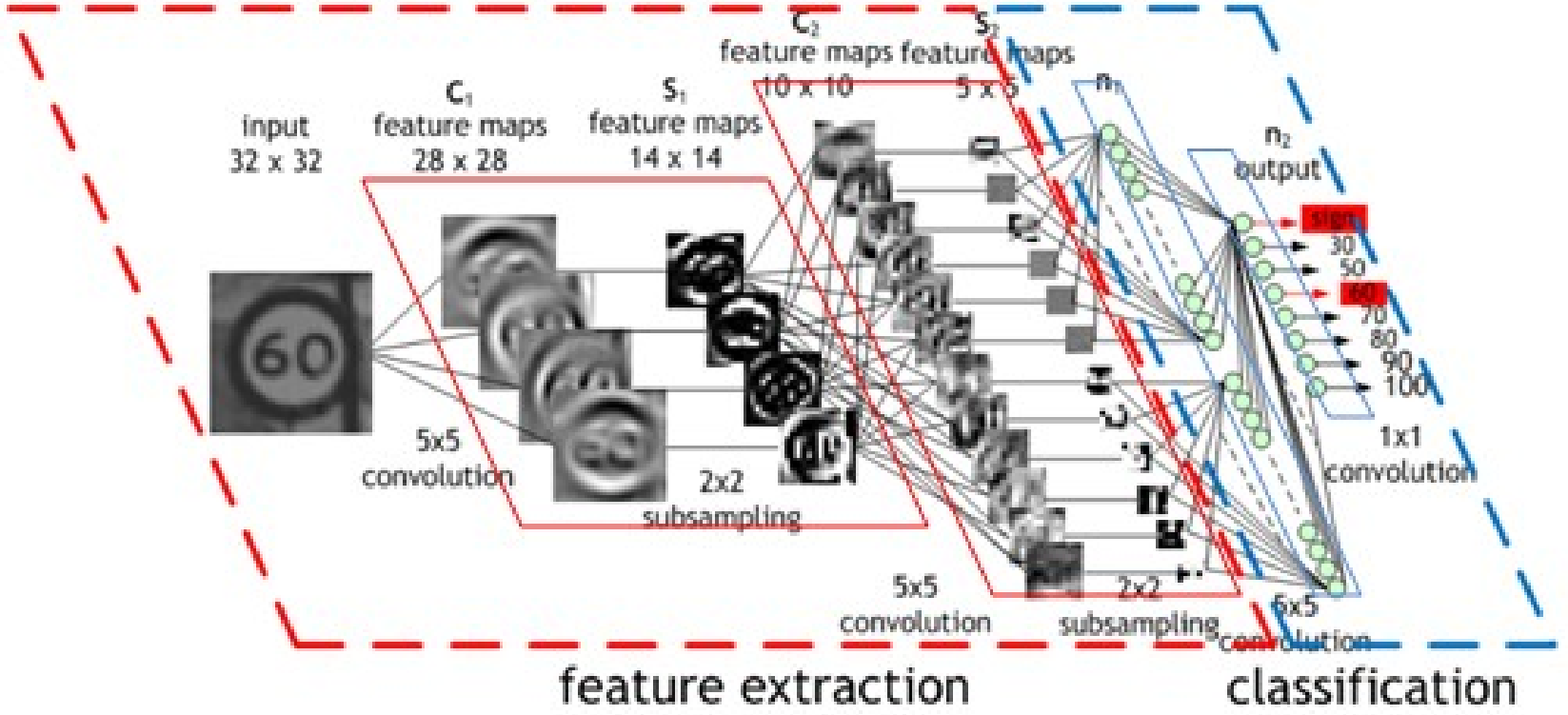
# Real-Time Pose Estimation




https://github.com/dragonfly90/mxnet_Realtime_Multi-Person_Pose_Estimation

# Convolutional Neural Networks





# Amazon SageMaker

## Build

Pre-built notebook instances

## Train

Highly-optimized machine learning algorithms

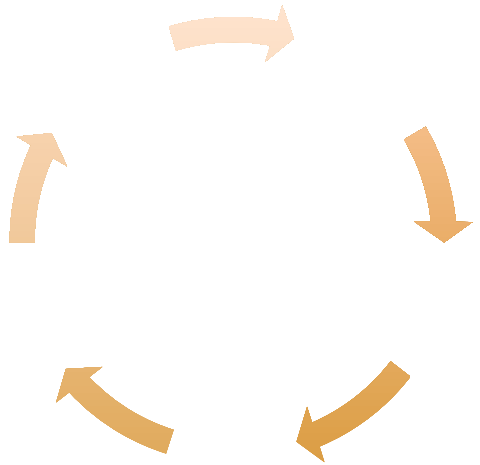
One-click training for ML, DL, and custom algorithms

Easier training with hyperparameter optimization

## Deploy

Fully-managed hosting at scale

Deployment without engineering effort



# Demos

https://github.com/juliensimon/dlnotebooks
https://github.com/guyernest/TensorFlowTutorials

1) Classifying MNIST with a CNN model (Keras)
2) Classifying images with pre-trained CNN models (MXNet)
3) Fine-tuning a pre-trained CNN model (Keras)
4) Generating new MNIST samples with a GAN (MXNet)

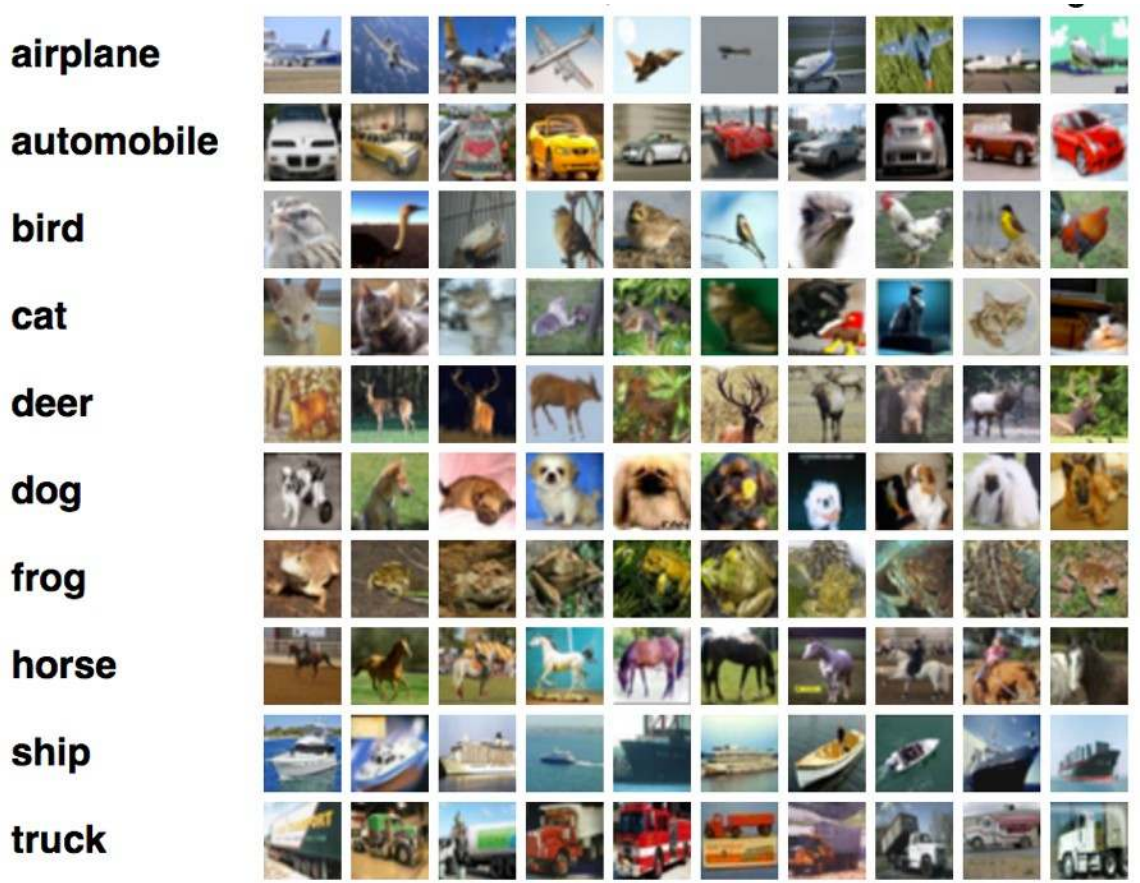
# Demo #2 – Using a pre-trained model

```
*** VGG16

[(0.46811387, 'n04296562 stage'), (0.24333163,
'n03272010 electric guitar'), (0.045918692, 'n02231487
walking stick, walkingstick, stick insect'),
(0.03316205, 'n04286575 spotlight, spot'),
(0.021694135, 'n03691459 loudspeaker, speaker, speaker
unit, loudspeaker system, speaker system')]

*** ResNet-152

[(0.8726753, 'n04296562 stage'), (0.046159592,
'n03272010 electric guitar'), (0.041658506, 'n03759954
microphone, mike'), (0.018624334, 'n04286575 spotlight,
spot'), (0.0058045341, 'n02676566 acoustic guitar')]

*** Inception v3

[(0.44991142, 'n04296562 stage'), (0.43065304,
'n03272010 electric guitar'), (0.067580454, 'n04456115
torch'), (0.012423956, 'n02676566 acoustic guitar'),
(0.0093934005, 'n03250847 drumstick')]
```

https://medium.com/@julsimon/an-introduction-to-the-mxnet-api-part-5-9e78534096db

# Demo #3 – Image classification: fine-tuning a model

- CIFAR-10 data set
  - 60,000 images in 10 classes
  - 32x32 color images
- Initial training
  - Resnet-50 CNN
  - 200 epochs
  - 82.12% validation
- Cars vs. horses
  - 88.8% validation accuracy




https://medium.com/@julsimon/keras-shoot-out-part-3-fine-tuning-7d1548c51a41

# Demo #3 – Image classification: fine-tuning a model

- Freezing all layers but the last one
- Fine-tuning on « cars vs. horses » for 10 epochs
- 2 minutes on 1 GPU
- 98.8% validation accuracy

```
Epoch 10/10
10000/10000 [==============================] - 12s
loss: 1.6989 - acc: 0.9994 - val_loss: 1.7490 - val_acc: 0.9880

2000/2000 [==============================] - 2s
[1.7490020694732666, 0.98799999999999999]
```

# Resources

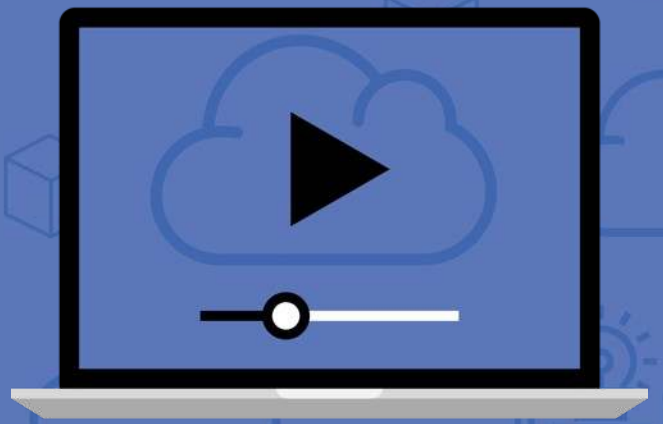
https://aws.amazon.com/machine-learning
https://aws.amazon.com/blogs/ai

https://mxnet.incubator.apache.org
https://github.com/apache/incubator-mxnet
https://github.com/gluon-api

https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

https://medium.com/@julsimon

# Thank you!



**Julien Simon, AI Evangelist, EMEA**
**@julsimon**