Microsoft

AI Immersion Workshop

May 9, 2017 | W Seattle Hotel, Seattle, WA

Microsoft

De-mystifying Deep Learning

Anusua Trivedi
Data Scientist
Email: antriv@microsoft.com
Twitter: @anurive

Talk Outline

- Deep Learning (DL)
- Deep Neural Networks (DNN)
- Types of DNNs
- DL Frameworks
- Use Cases

Traditional ML Vs DL

Traditional ML requires manual feature extraction/engineering

Deep learning can automatically learn features in data

Feature extraction for unstructured data is very difficult

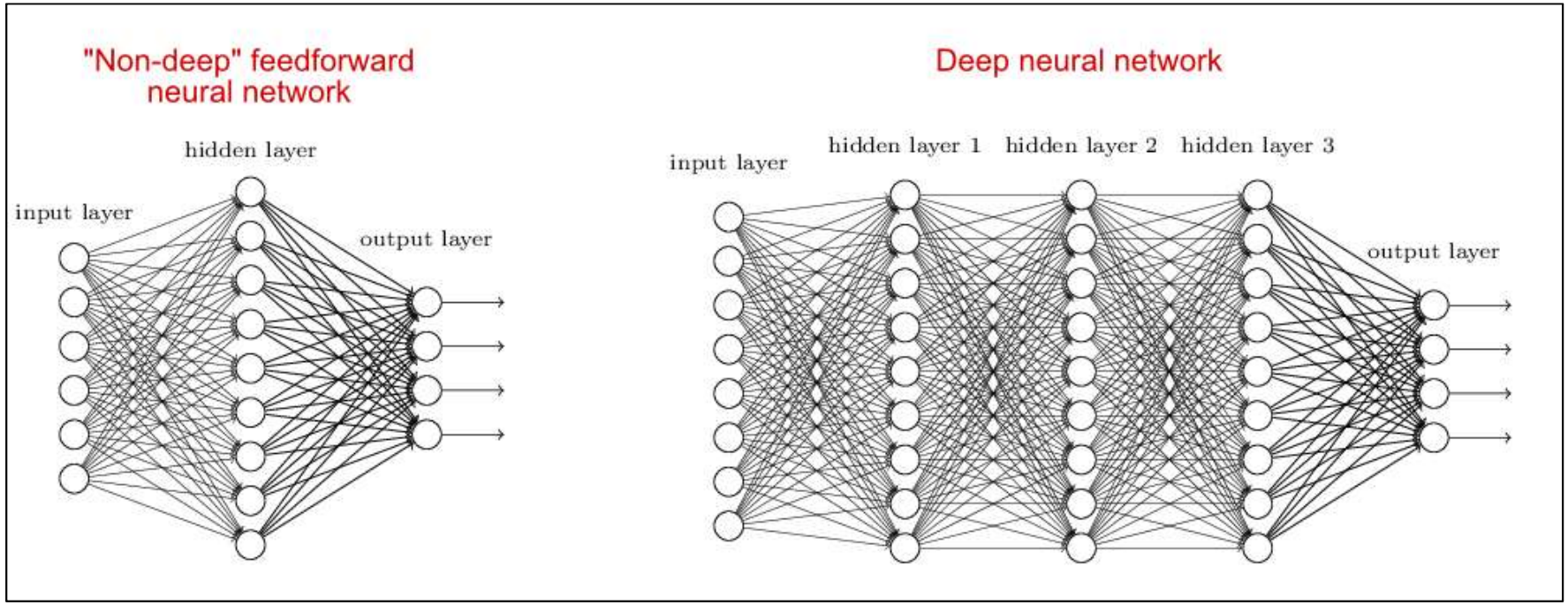
Deep learning is largely a "black box" technique, updating learned weights at each layer

Why is DL popular?

- DL models has been here for a long time
  - Fukushima (1980) – Neo-Cognitron
  - LeCun (1989) – Convolutional Neural Network

- DL popularity grew recently
  - With growth of Big Data
  - With the advent of powerful GPUs

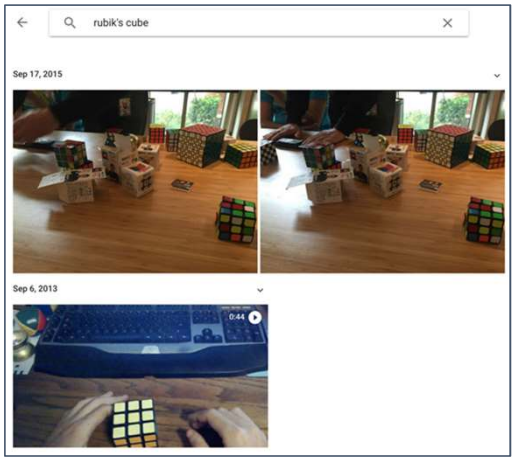
Deep Neural Network (DNN)



Common DNNs

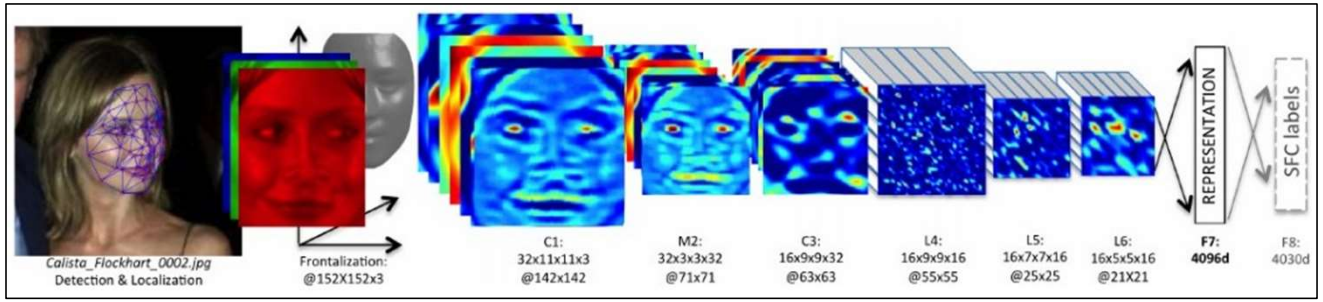
- Deep Convolutional Neural Network (DCNN)
  - To extract representation from images
- Recurrent Neural Network (RNN)
  - To extract representation from sequential data
- Deep Belief Neural Network (DBN)
  - To extract hierarchical representation from a dataset

Supervised: ConvNets are everywhere



e.g. Google Photos search

[Goodfellow et al. 2014]



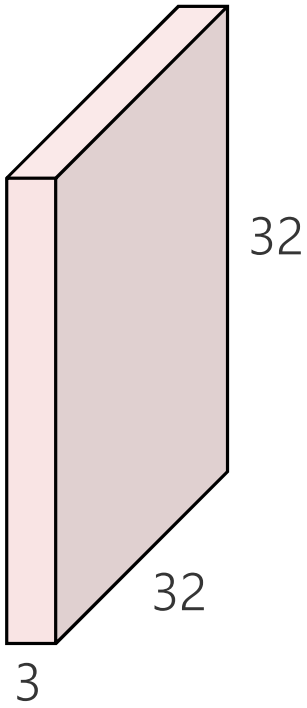
Face Verification, Taigman et al. 2014 (FAIR)

Self-driving cars

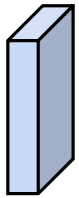
Convolution Layer

32x32x3 image

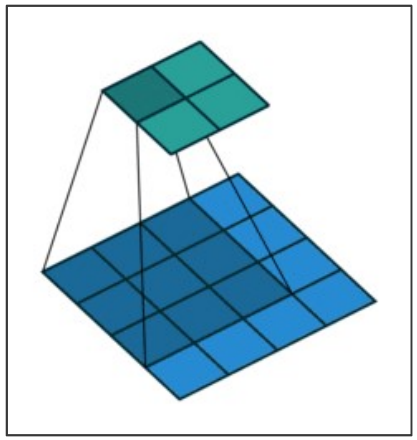
Filters always extend the full depth of the input volume



5x5x3 filter



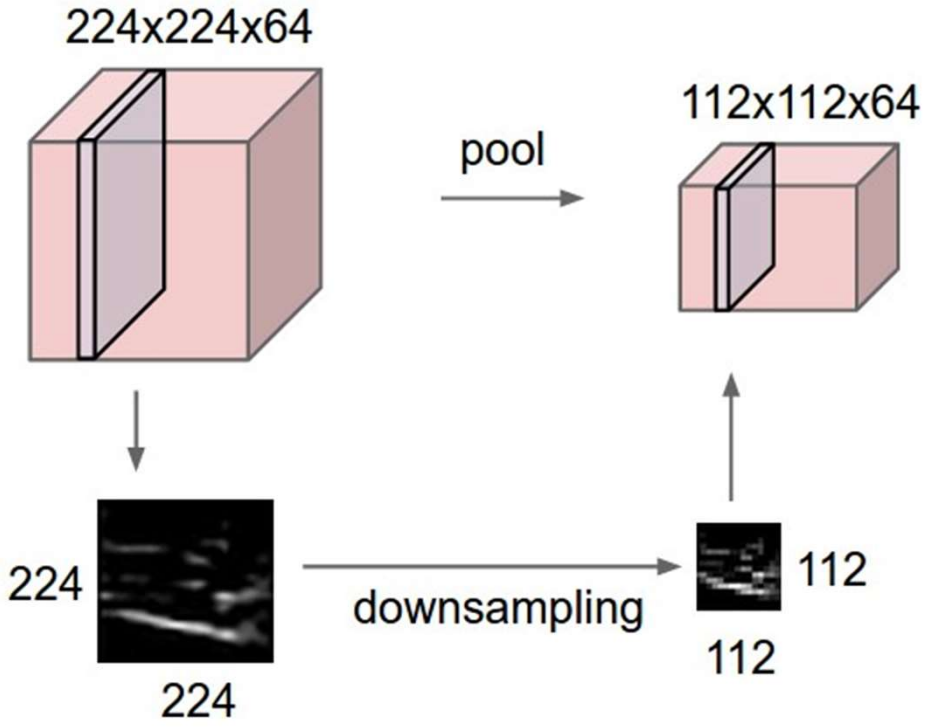
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"



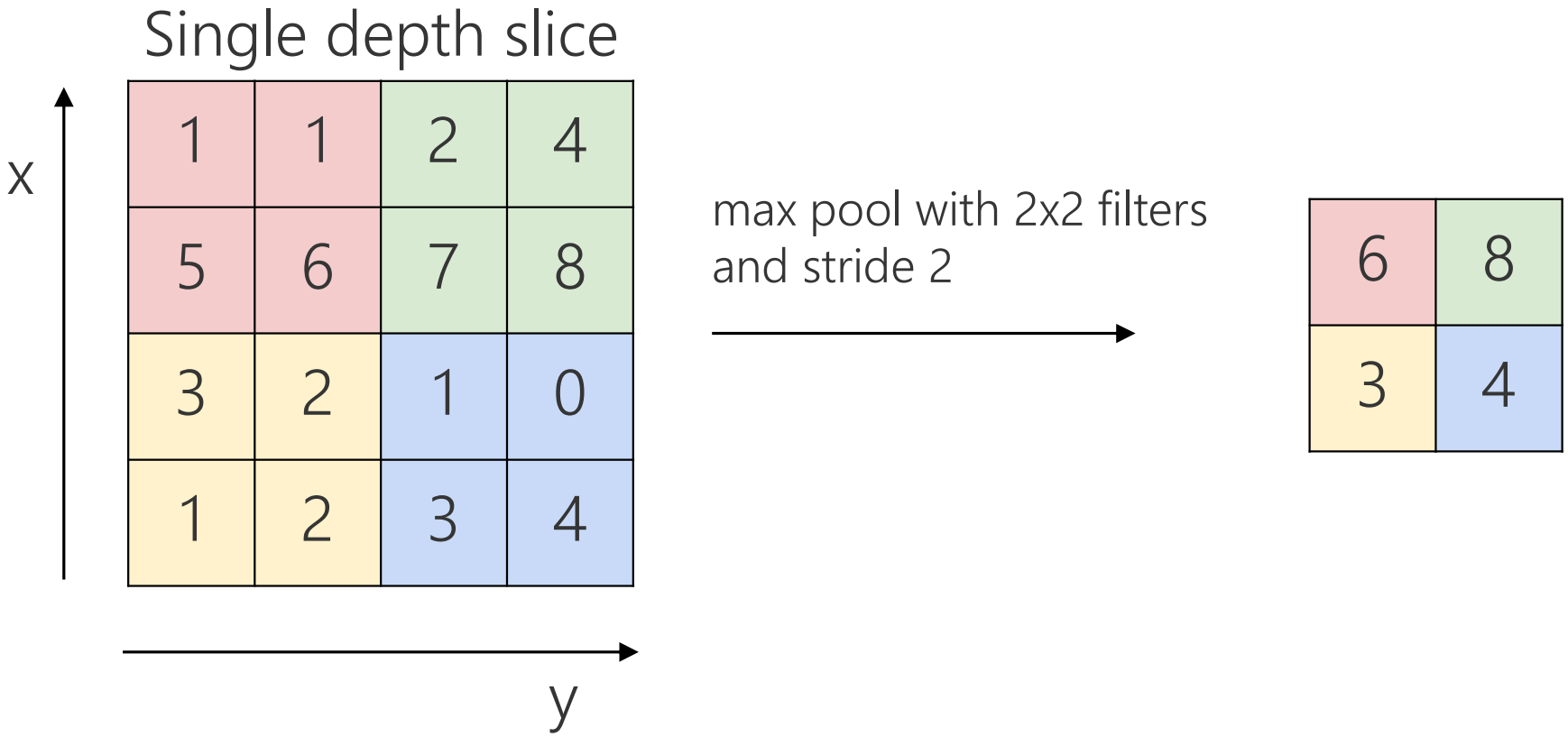
Convolving a 3 X 3 kernel over a 4 X 4 input

Pooling layer

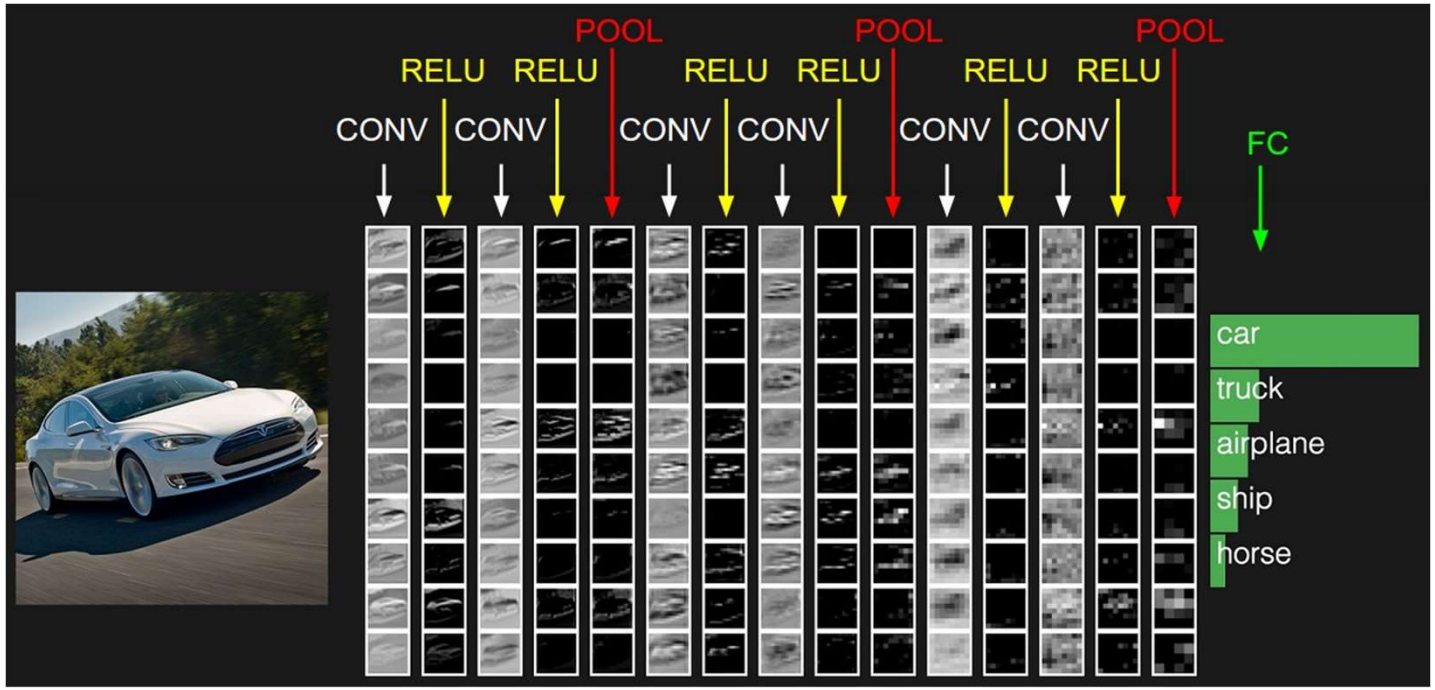
- makes the representations smaller and more manageable
- operates over each activation map independently



Max Pooling

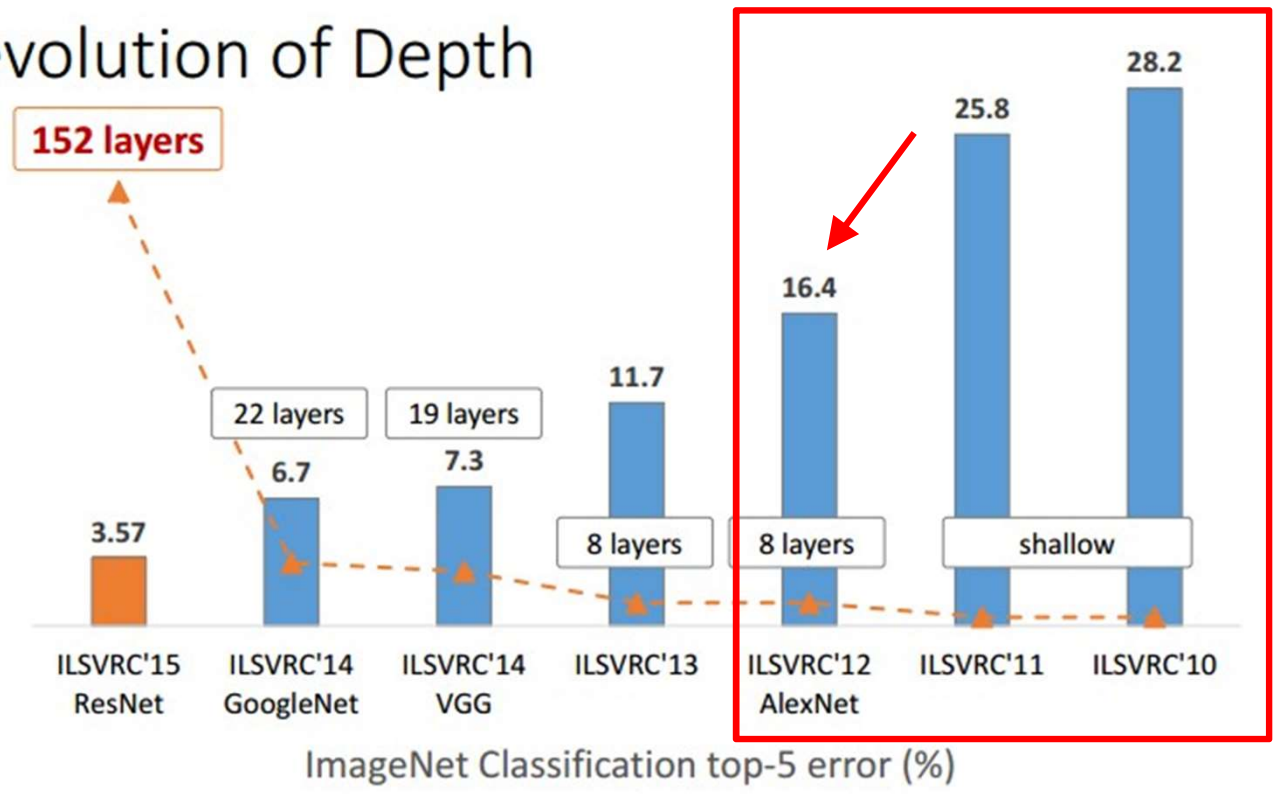

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Fully Connected Layer (FC layer)



Microsoft Research

Revolution of Depth



ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Transfer Learning & Fine-tuning

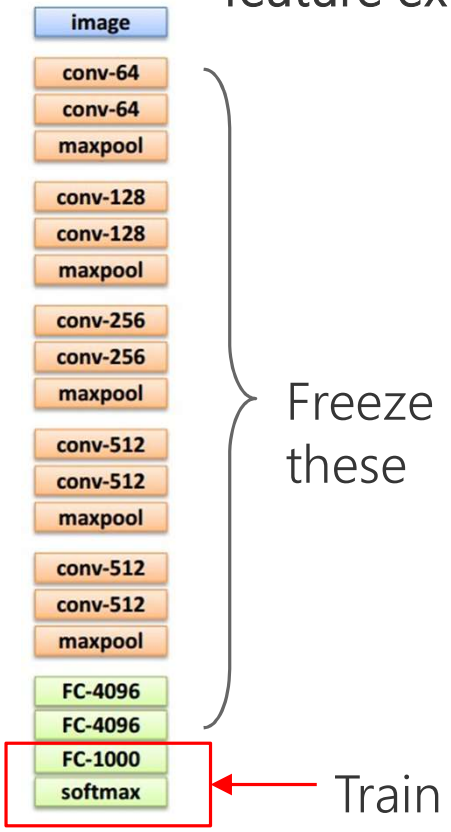
Training data	1000s to millions of labeled images
Computation	Compute intensive
Training Time	Days to Weeks for real problems
Model accuracy	High (can over fit to small datasets)

Training data	100s to 1000s of labeled images (small)
Computation	Moderate computation
Training Time	Seconds to minutes
Model accuracy	Good, depends on the pre-trained CNN model

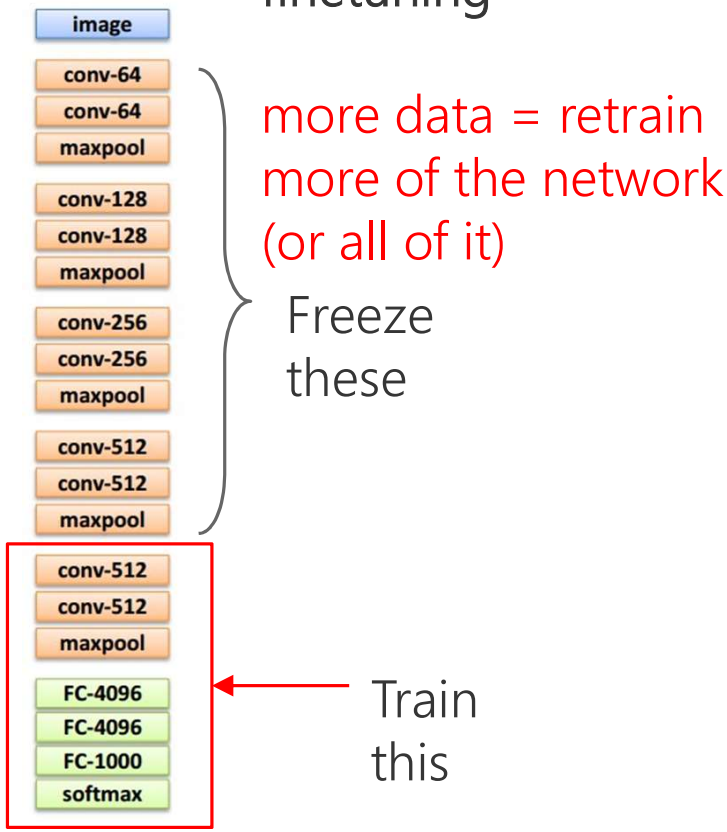
Transfer Learning

1. Train on Imagenet

2. Small dataset: **feature extractor**

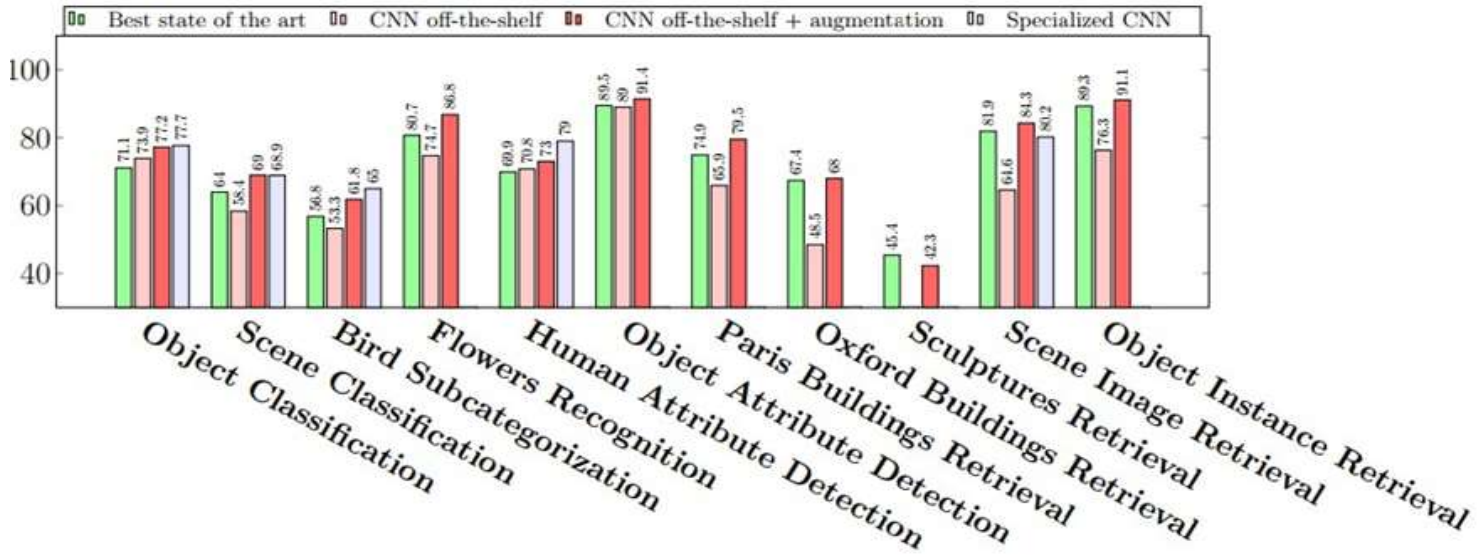


3. Medium dataset: **finetuning**

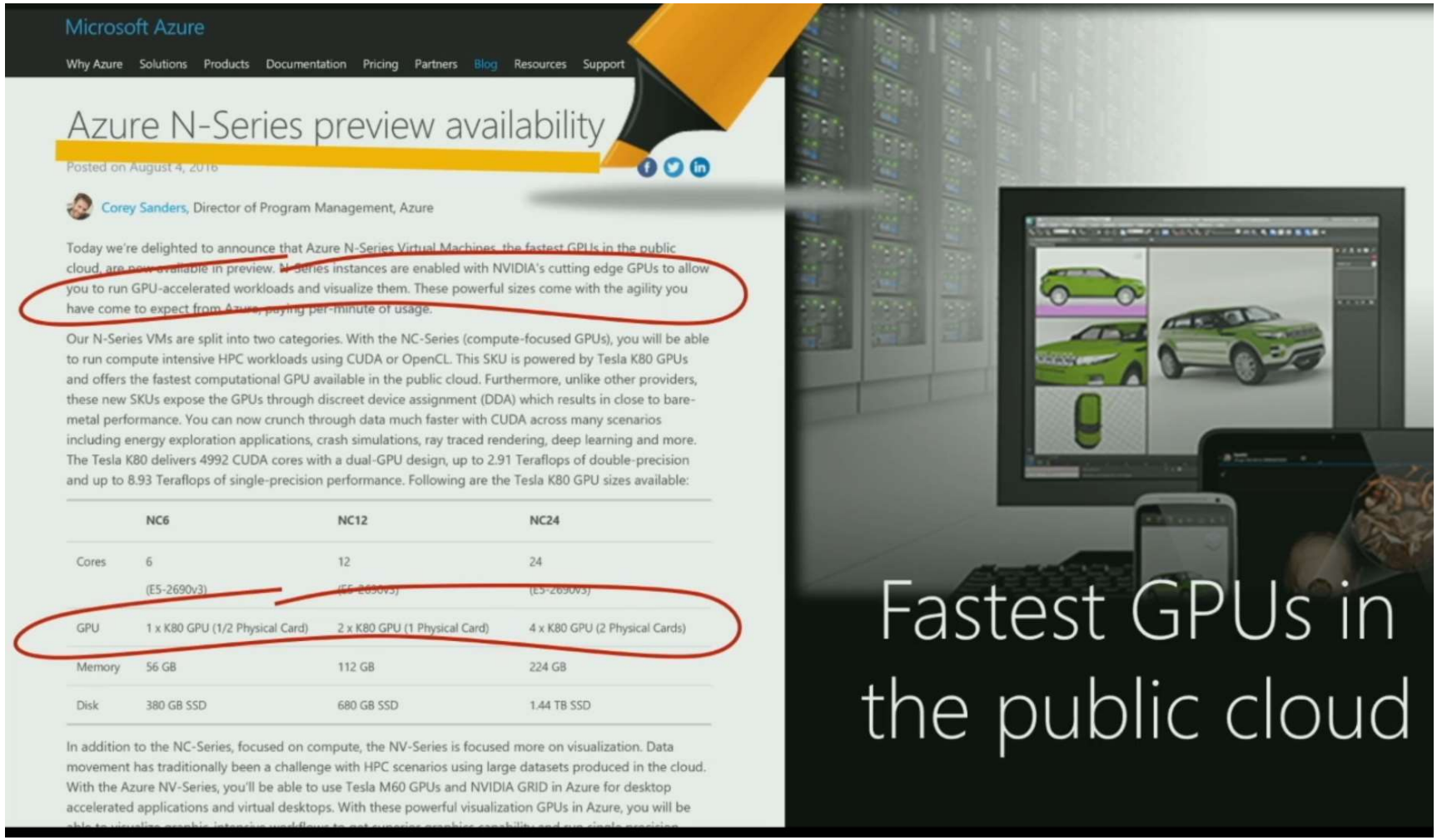


Transfer Learning

CNN Features off-the-shelf: an Astounding Baseline for Recognition
[Razavian et al, 2014]

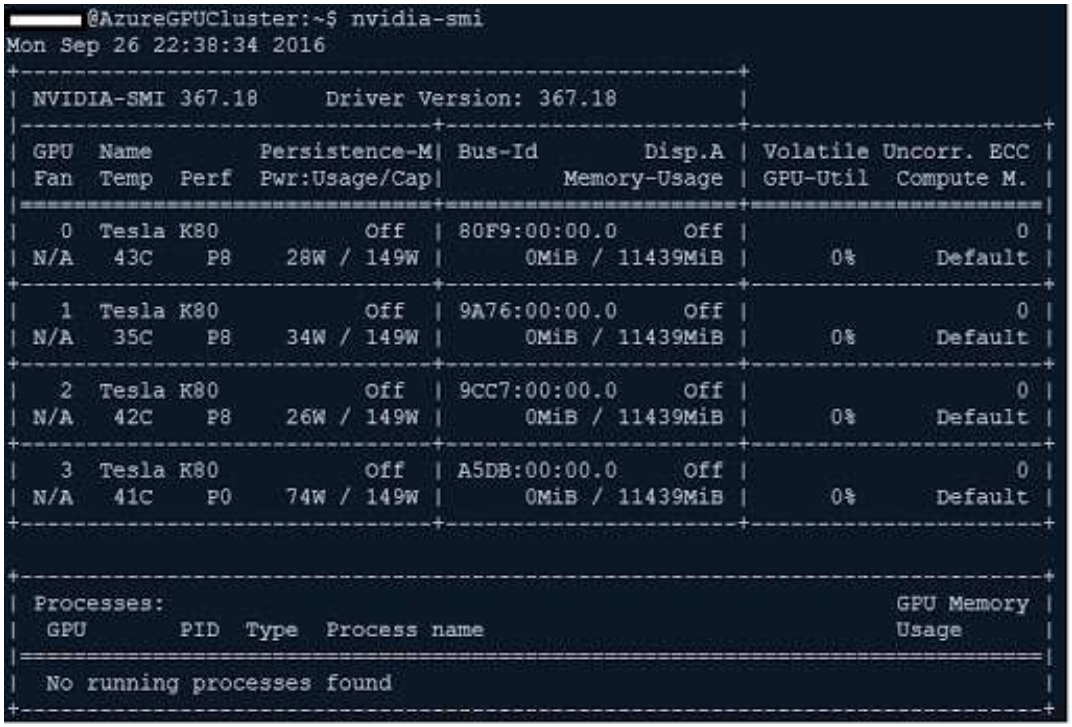


GPUs in Azure



Azure GPU DSVM

- Used Ubuntu 16.04 N-Series VM
- NC24 VM with 4 NVIDIA Tesla K80



The Microsoft Cognitive Toolkit (CNTK)

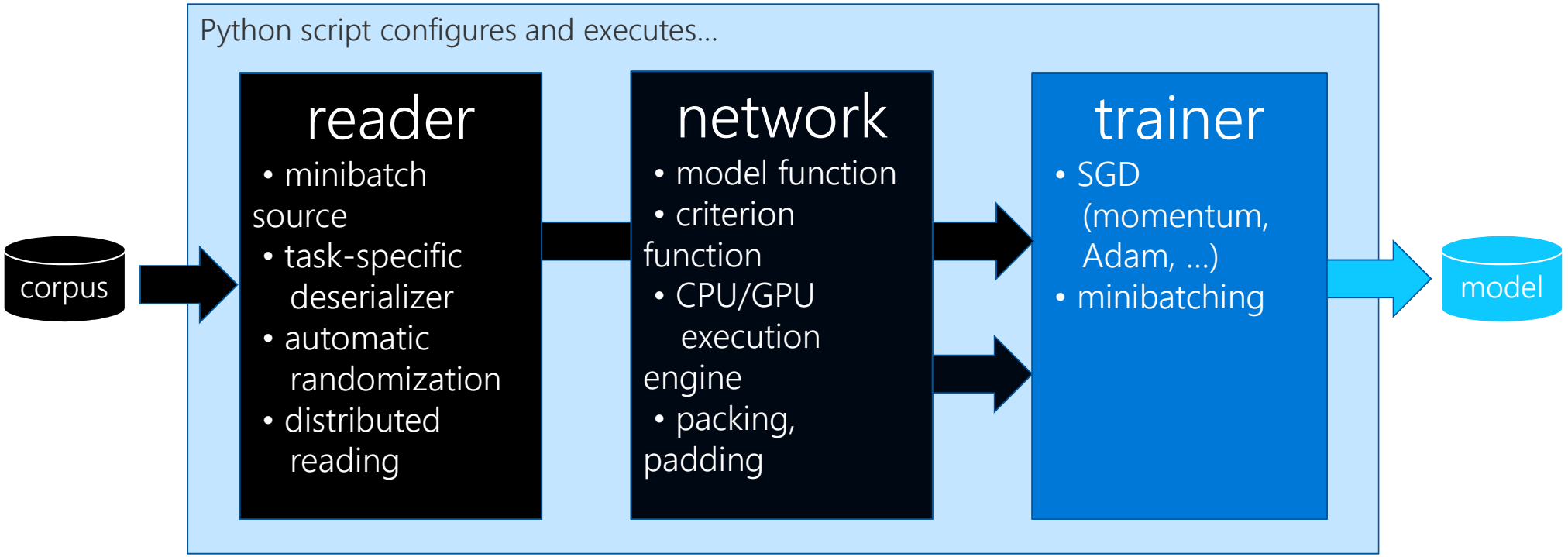
- CNTK expresses (nearly) **arbitrary neural networks** by composing simple building blocks into complex **computational networks**, supporting relevant network types and applications.

- CNTK is **production-ready**: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.

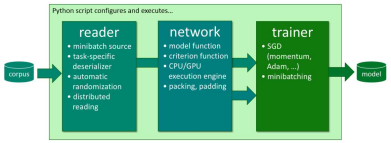
Microsoft Cognitive Toolkit

- ease of use
  - *what*, not *how*
  - powerful stock library
- fast
  - optimized for NVidia GPUs & libraries
  - best-in-class multi-GPU/multi-server algorithms
- flexible
  - powerful & composable Python and C++ API
- Linux and Windows
- Development is completely in the open

Anatomy of a training job



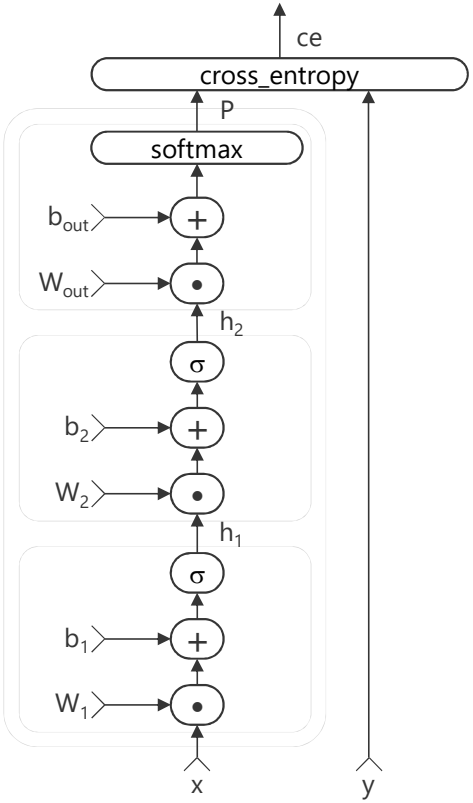
how to: reader

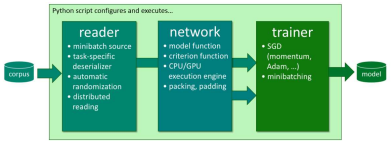



```
def create_reader(map_file, mean_file, is_training):
    # image preprocessing pipeline
    transforms = [
        ImageDeserializer.crop(crop_type='Random', ratio=0.8, jitter_type='uniRatio')
        ImageDeserializer.scale(width=image_width, height=image_height, channels=num_channels,
                                interpolations='linear'),
        ImageDeserializer.mean(mean_file)
    ]
    # deserializer
    return MinibatchSource(ImageDeserializer(map_file, StreamDefs(
        features = StreamDef(field='image', transforms=transforms), '
        labels   = StreamDef(field='label', shape=num_classes)
    )), randomize=is_training, epoch_size = INFINITELY_REPEAT if is_training else FULL_DATA_SWEEP)
```

- automatic on-the-fly randomization important for large data sets
- readers compose, e.g. image → text caption

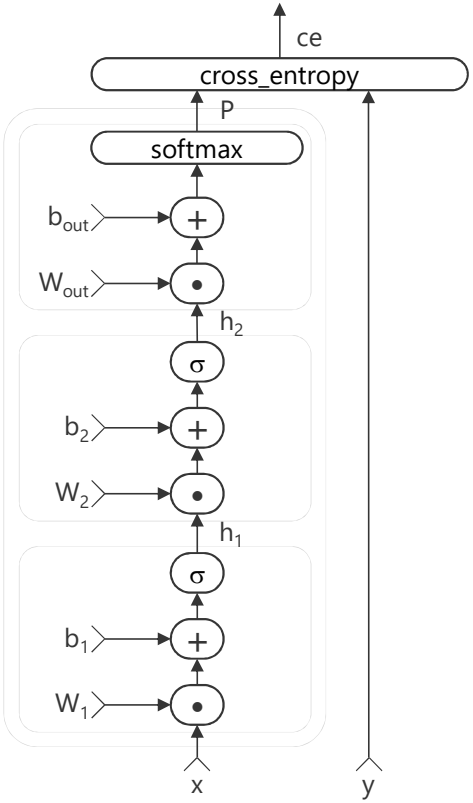
how to: network

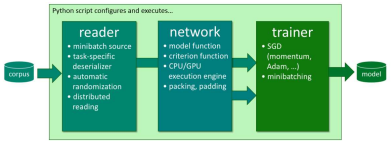





```
M = 40 ; N = 512 ; J = 9000   # feat/hid/out dim
x = Input(M) ; y = Input(J)   # feat/labels

from layers import *
model = Sequential ([
      Dense(N), sigmoid,
      Dense(N), sigmoid,
      Dense(J), softmax
])
P = model(x)
ce = cross_entropy(P, y)
```

how to: network

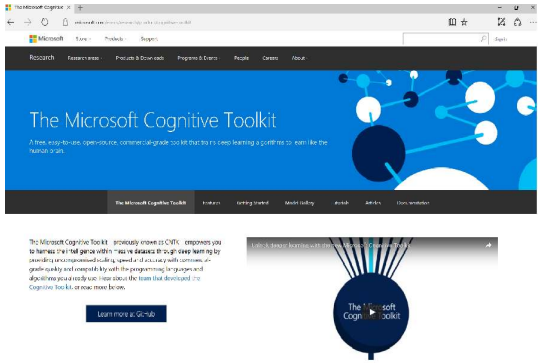


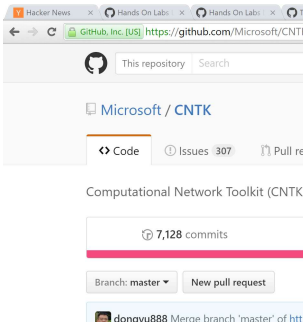


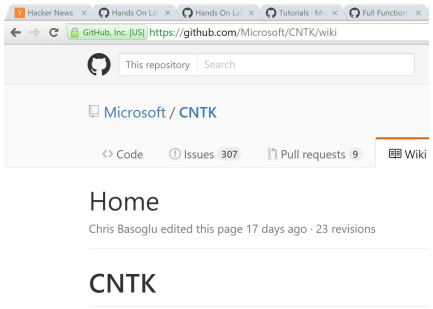

```
M = 40 ; N = 512 ; J = 9000 ; L = 2
x = Input(M) ; y = Input(J)  # feat/labels

from layers import *
model = Sequential ([
      [ Dense(N, activation=sigmoid)
          for i in range(L) ],
      Dense(J, activation=softmax)
])
P = model(x)
ce = cross_entropy(P, y)
```

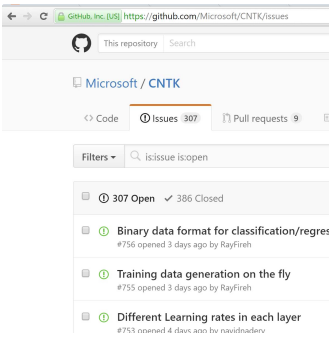
Microsoft Cognitive Toolkit

- Github: `https://github.com/Microsoft/CNTK`
- Wiki: `https://github.com/Microsoft/CNTK/wiki`
- Issues: https://github.com/Microsoft/CNTK/issues
- `cntkhelp@Microsoft.com`









Hands-on