

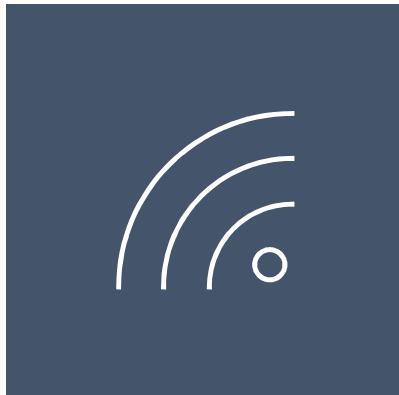


Machine Learning, AI & Data Science Conference

June 11–13
Redmond

20
18

WiFi Information



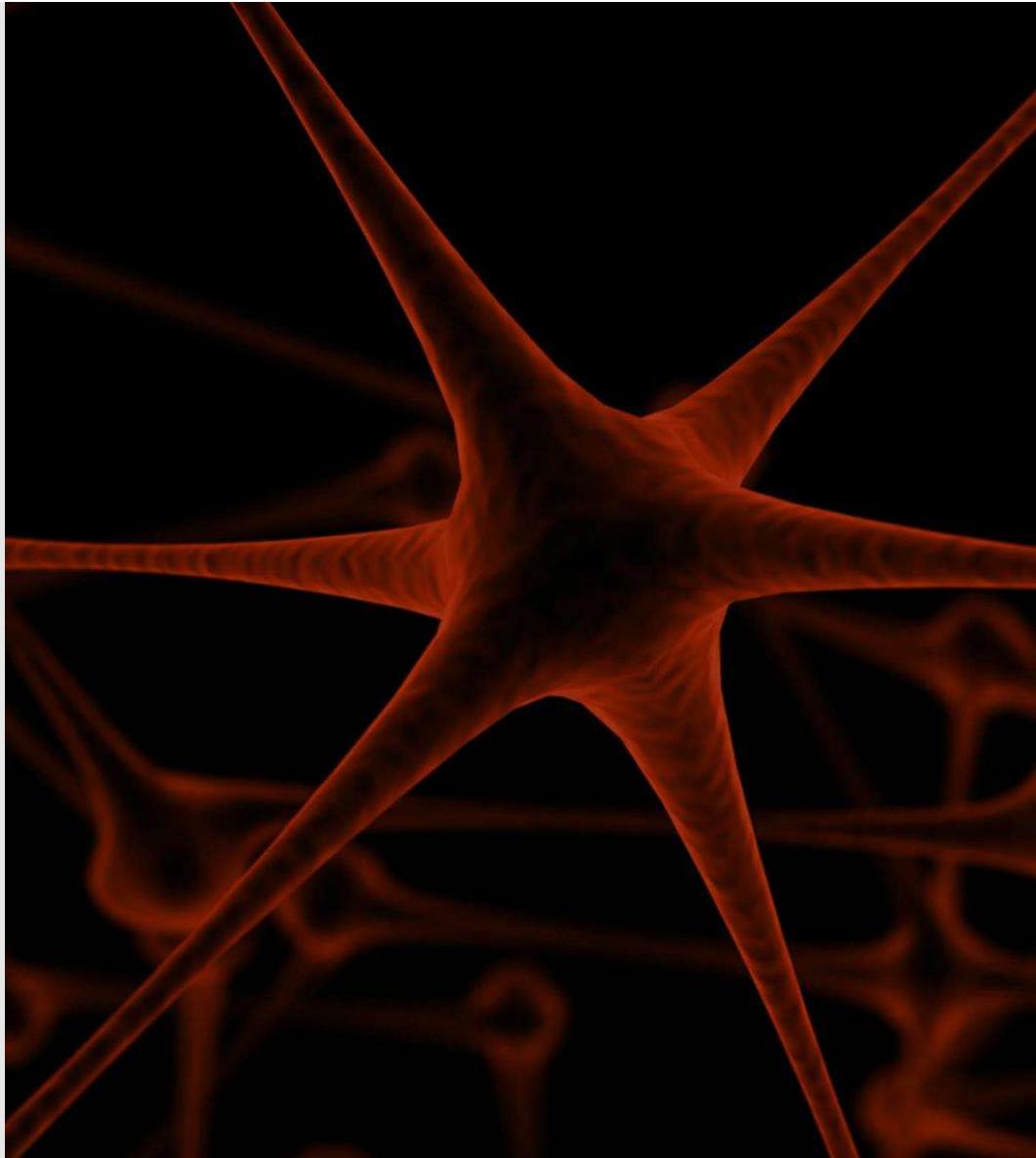
**Network: MLADS2018
Password: MLADS2018!**



De-mystifying Deep Learning

Anusua Trivedi, Avi Thaker

Email: antriv@microsoft.com,
avthaker@microsoft.com



Talk Outline

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

Resources

[MLADS 2018 deeplearning](#)

Traditional ML Vs DL

Traditional ML requires
manual feature
extraction/engineering

Feature extraction for
unstructured data is very
difficult

Deep learning can
automatically learn features
in data

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 learning begins with a little function

It all starts with a humble linear function called a perceptron.

$$\begin{array}{r} \text{weight1} \times \text{input1} \\ \text{weight2} \times \text{input2} \\ \text{weight3} \times \text{input3} \\ \hline \text{sum} \end{array}$$

Perceptron:
If sum > threshold: output 1
Else: output 0

Example: The inputs can be your data. Question: Should I buy this car?

$$\begin{array}{r} 0.2 \times \text{gas mileage} \\ 0.3 \times \text{horsepower} \\ 0.5 \times \text{num cup holders} \\ \hline \text{sum} \end{array}$$

Perceptron:
If sum > threshold: buy
Else: walk

These little functions are chained together

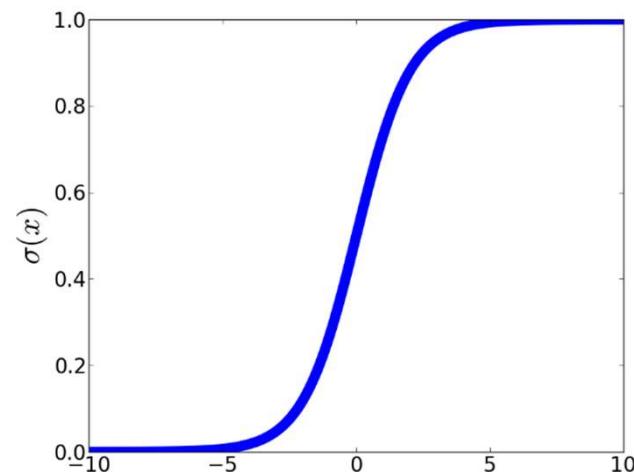
- Deep learning comes from chaining a bunch of these little functions together. Chained together, they are called **neurons**.
- To create a neuron, we add a nonlinearity to the perceptron to get extra representational power when we chain them together.
- Our nonlinear perceptron is sometimes called a sigmoid.

$$\sigma\left(\sum_i w_i x_i + b\right)$$

where

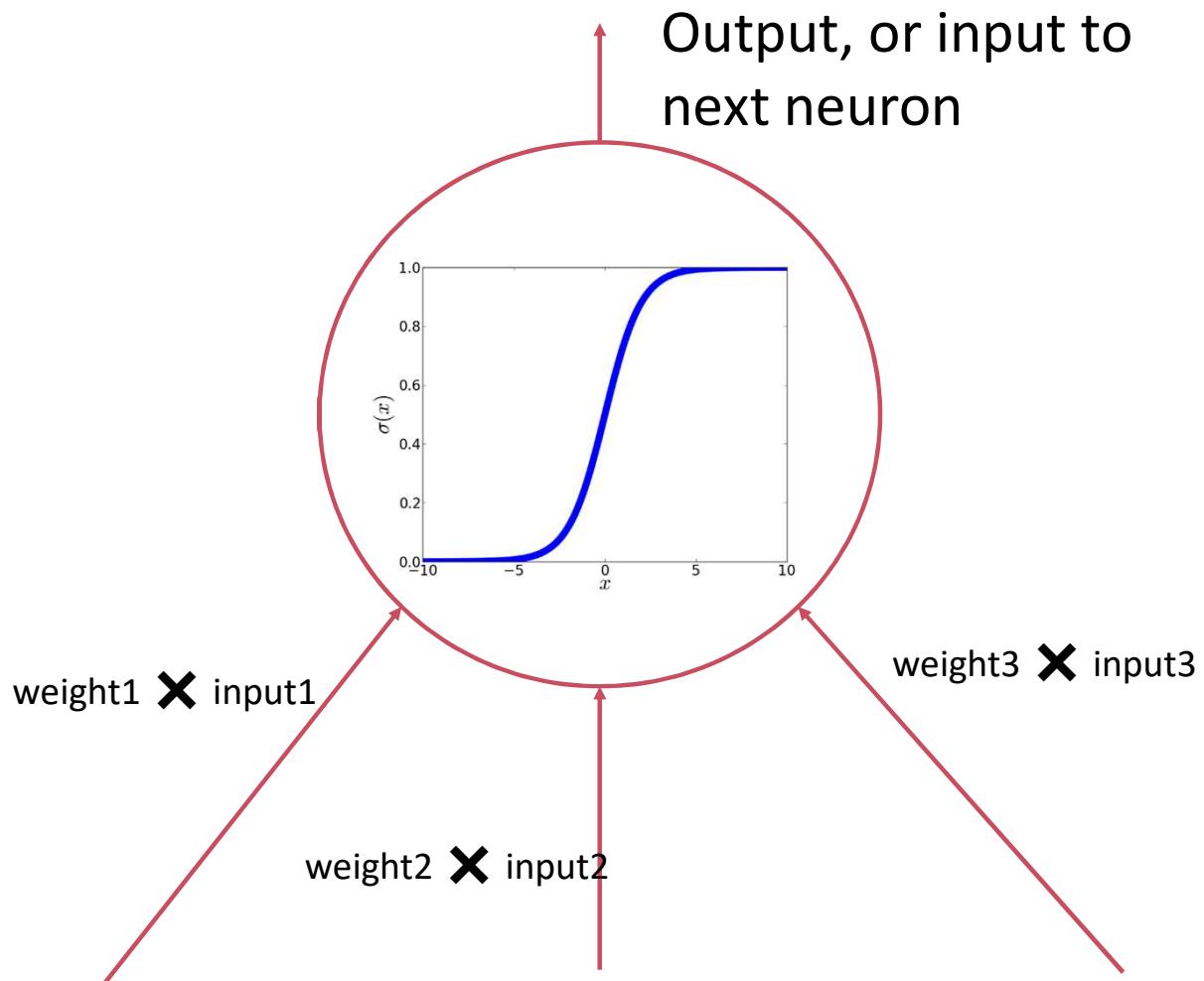
$$\sigma(x) = \frac{1}{1 + \frac{1}{e^x}}$$

The value b just offsets the sigmoid so the center is at 0.

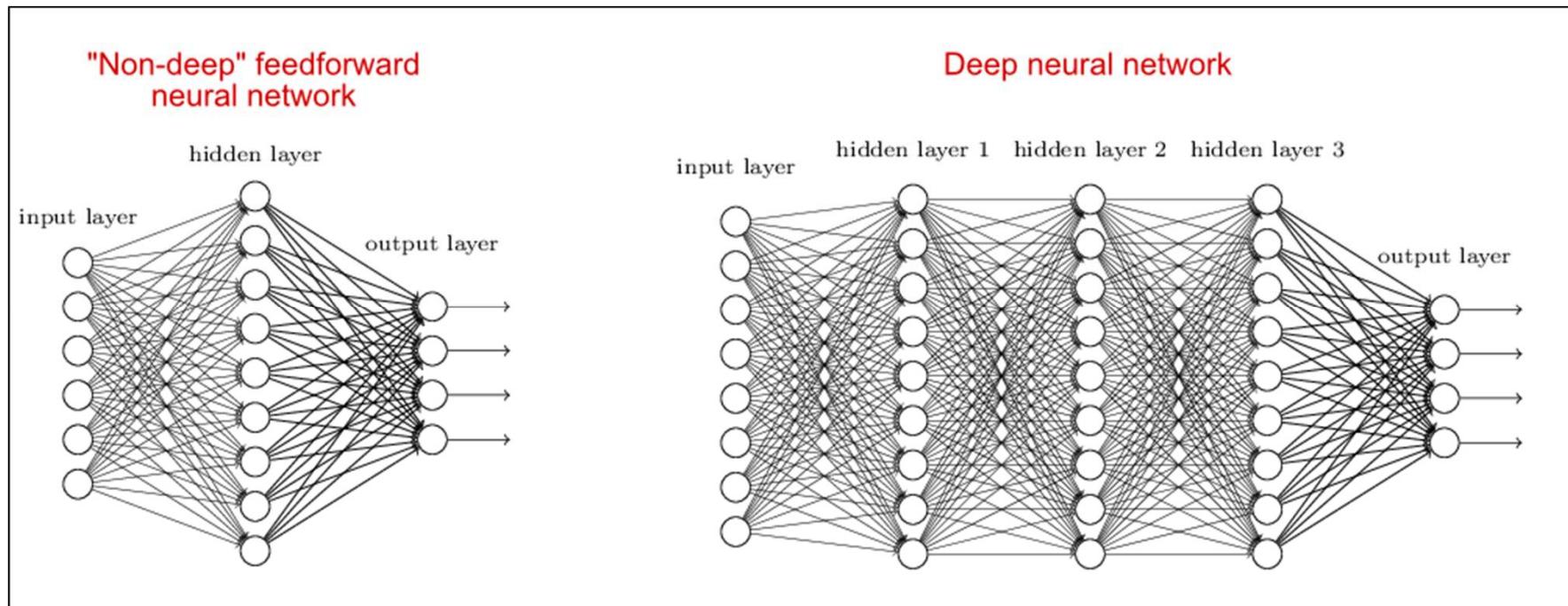


Plot of a sigmoid

Single artificial neuron



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

Comparing Representations

Vector representation

taco = [17.32, 82.9, -4.6, 7.2]

- Vectors have a similarity score.
A taco is not a burrito but similar.
- Vectors have internal structure
[Mikolov et al., 2013].
 $\text{Italy} - \text{Rome} = \text{France} - \text{Paris}$
 $\text{King} - \text{Queen} = \text{Man} - \text{Woman}$
- Vectors are grounded in experience.
- Meaning relative to predictions.

Symbolic representation

taco = *taco*

- Symbols can be the same or not.
- A taco is just as different from a burrito as a Toyota.
- Symbols have no structure.
- Symbols are arbitrarily assigned.
- Meaning relative to other symbols.

Common Open-Source DL Frameworks

Non-symbolic

Torch, Caffe

Manual optimization

Memory intensive

Symbolic

CNTK, MXNET, Theano,
TensorFlow

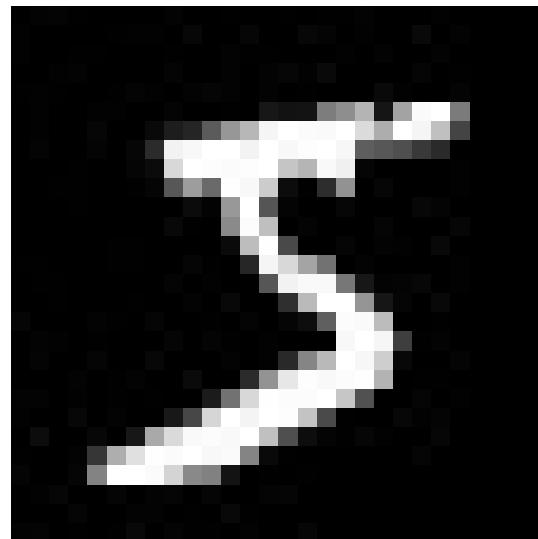
Automatic optimization

Memory reuse

Deep learning and computer vision

Vision is hard

Vision is hard because images are big matrices of numbers.



Example from MNIST
handwritten digit dataset
[LeCun and Cortes, 1998].

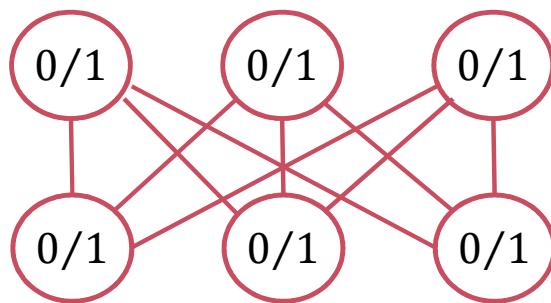
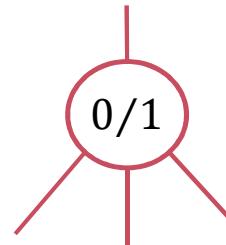
How a computer sees an image

[22, 81, 44, 88, 17, 0, ..., 45]

- Even harder for 3D objects.
- You move a bit, and everything changes.

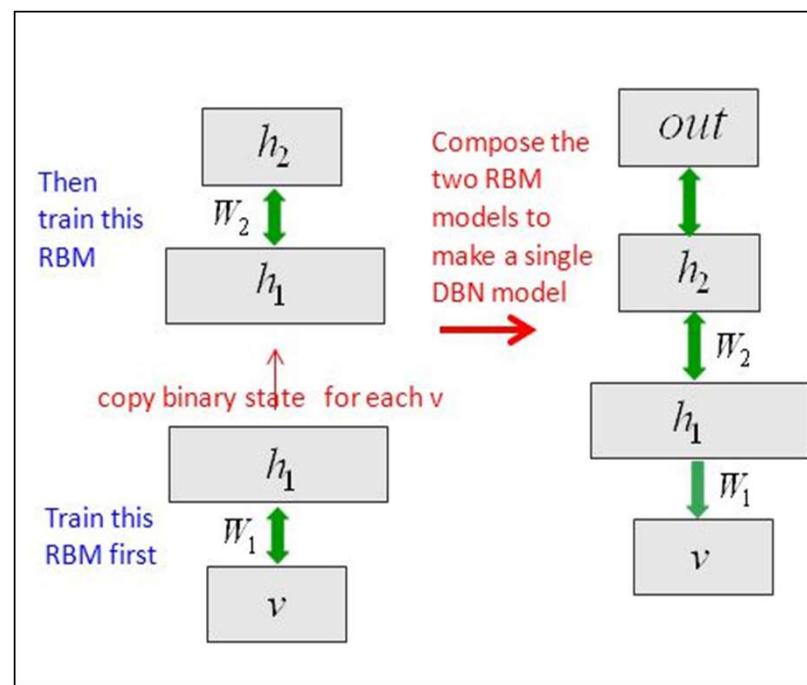
RBM: Unsupervised Model

- ❑ Stochastic binary neuron
- ❑ Probabilistically outputs 0 (turns off) or 1 (turns on) based on the weight of the inputs from on units.

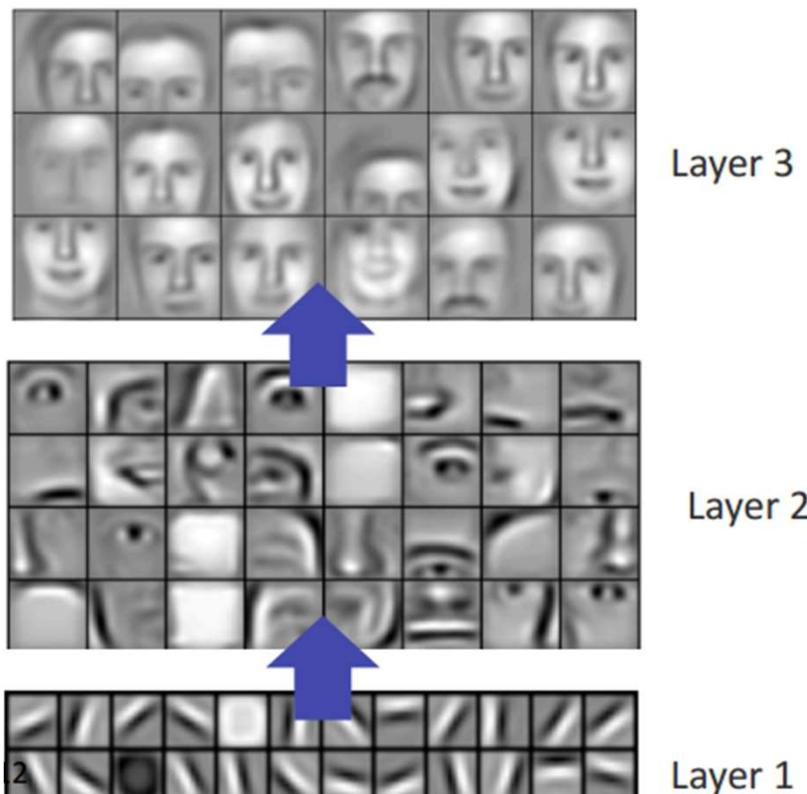


- ❑ Limit connections to be from one layer to the next.
- ❑ Fast because decisions are made locally.
- ❑ Trained in an unsupervised way to reproduce the data.

Stack up the layers to make a DBN



Computer vision, scaling up

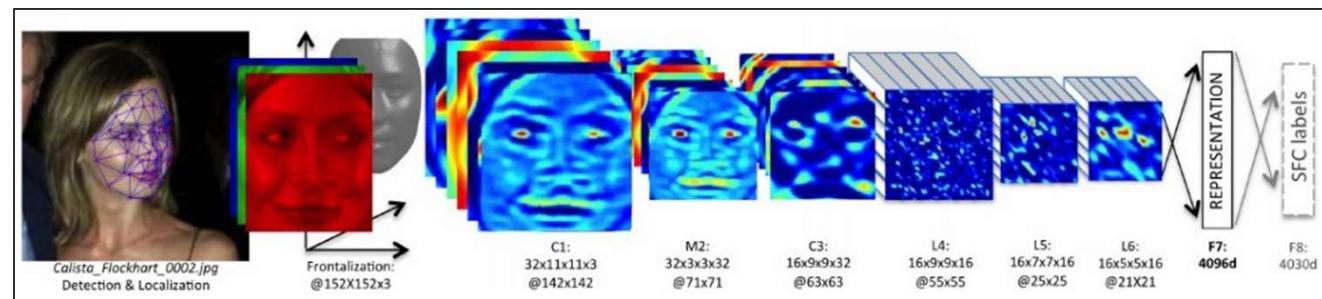
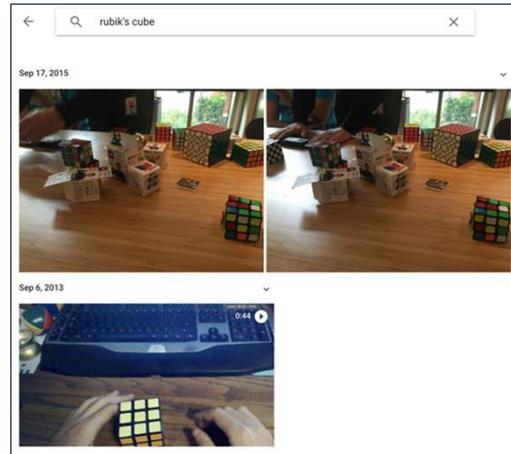


Unsupervised learning was scaled up by Honglak Lee et al. [2009] to learn high-level visual features.

Further scaled up by Quoc Le et al. [2012].

- Used 1,000 machines (16,000 cores) running for 3 days to train 1 billion weights by watching YouTube videos.
- The network learned to identify cats.
- The network wasn't told to look for cats, it naturally learned that cats were integral to online viewing.

Supervised: ConvNets are everywhere



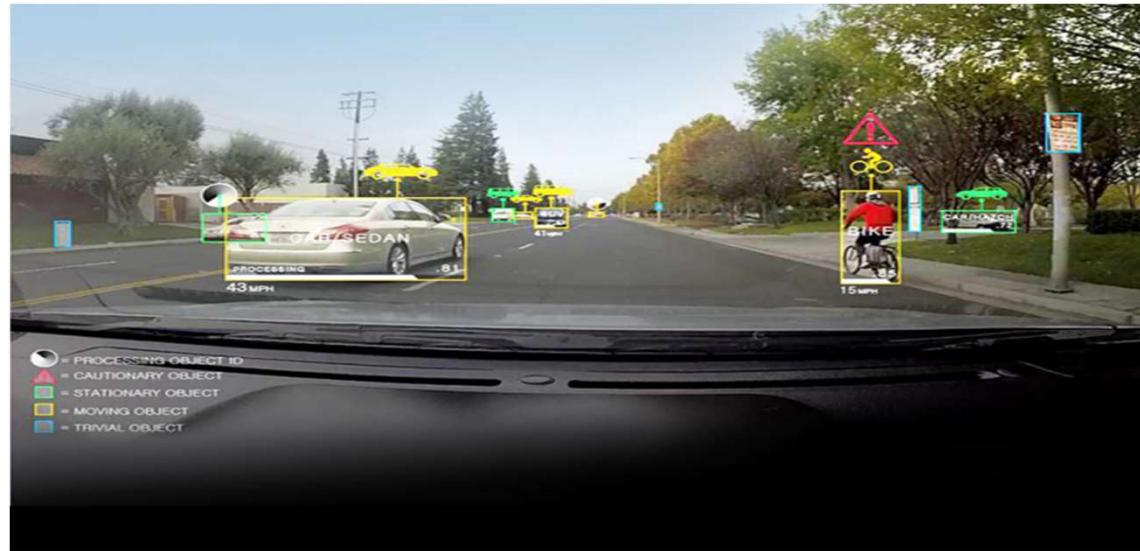
Face Verification, Taigman et al. 2014 (FAIR)

e.g. Google Photos search



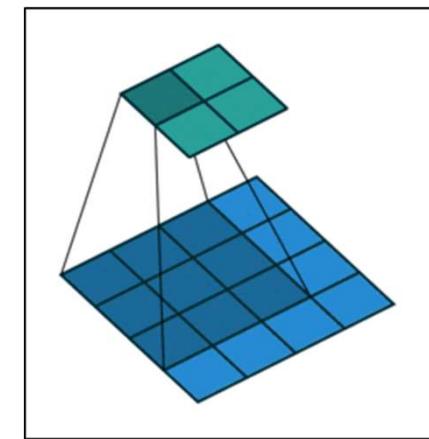
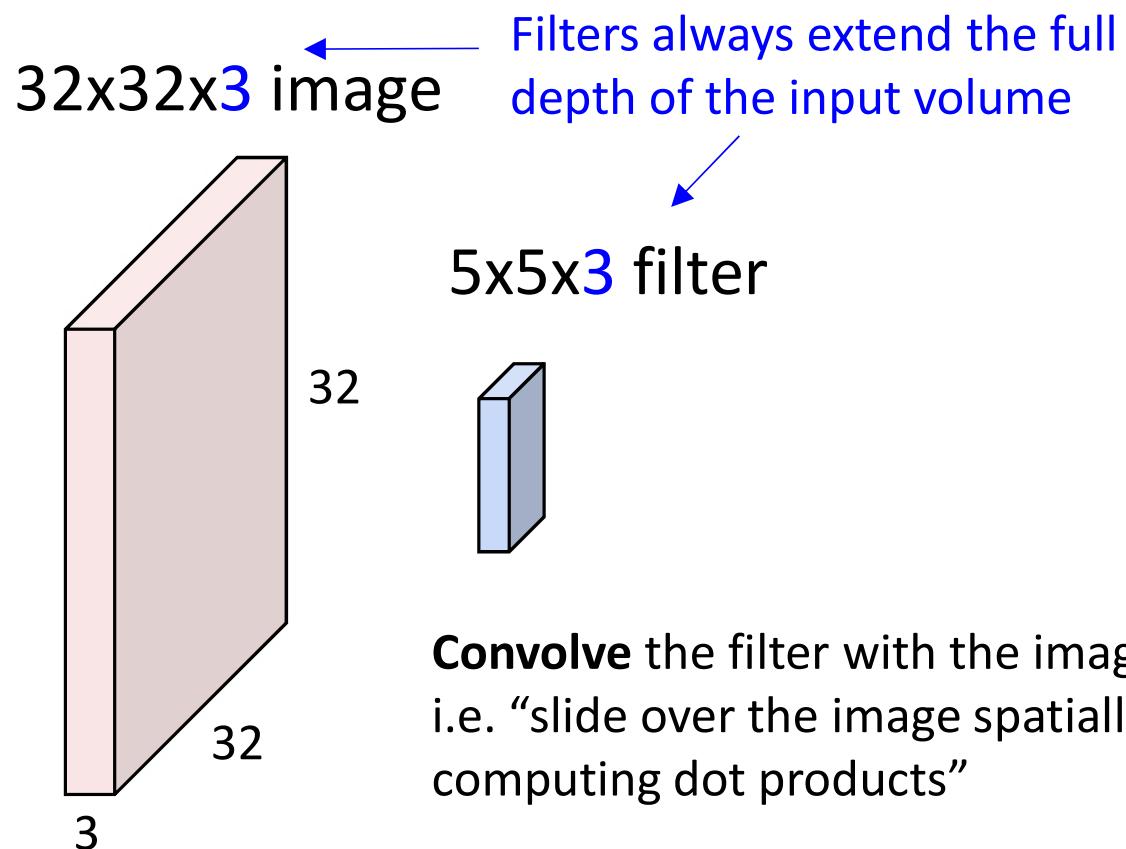
[Goodfellow et al. 2014]

*Andrej Karpathy's recent presentation



Self-driving cars

Convolution Layer



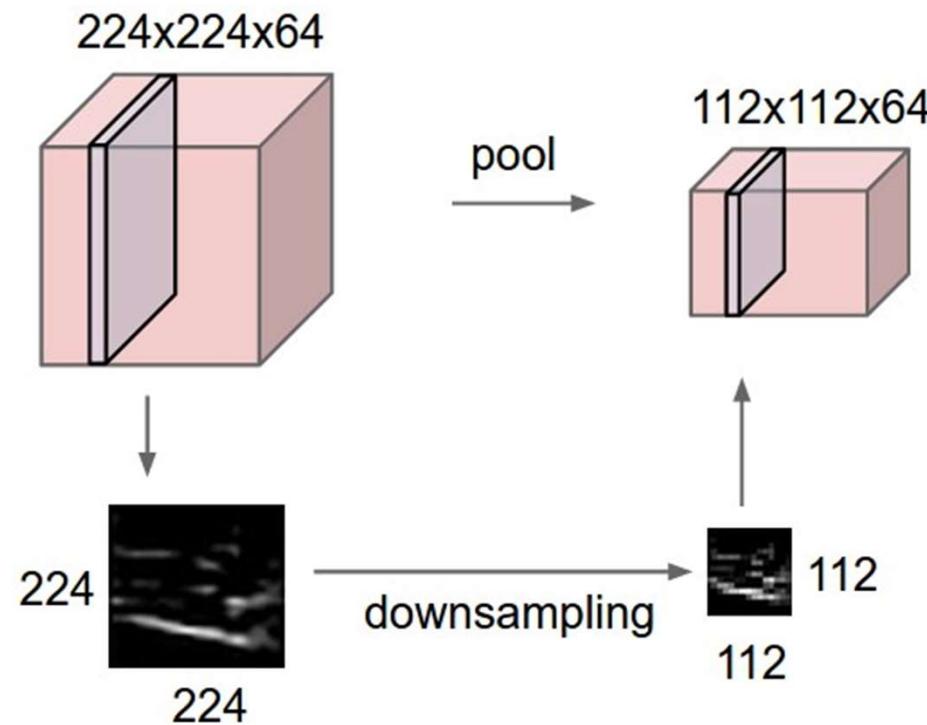
Convolving a 3 X 3 kernel over a
4 X 4 input

*Andrey Karpathy's recent presentation

*<http://iamaaditya.github.io/2016/03/one-by-one-convolution/>

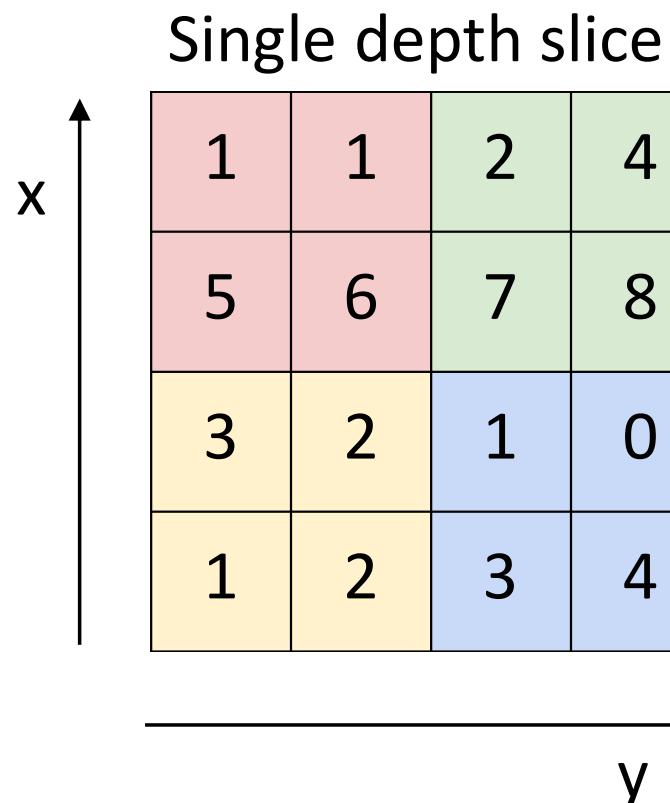
Pooling layer

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



*Andrej Karpathy's recent presentation

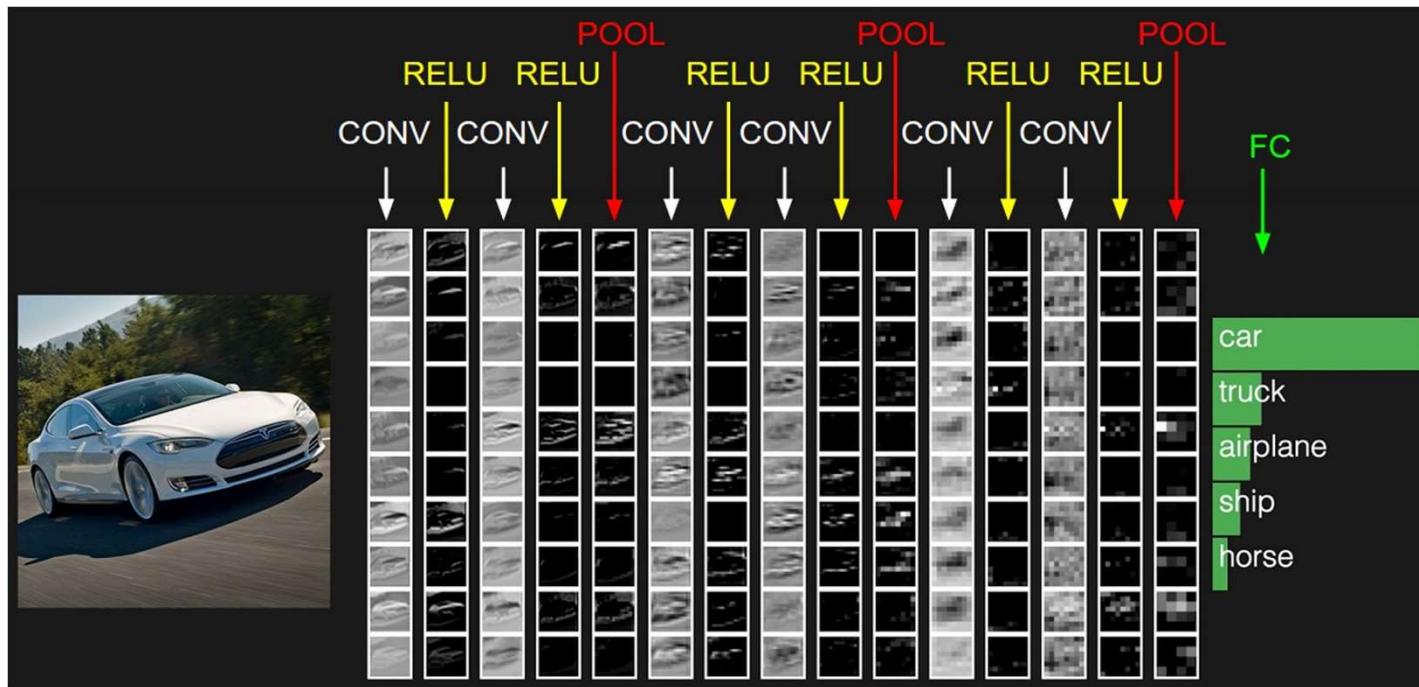
Max Pooling



max pool with 2x2 filters
and stride 2

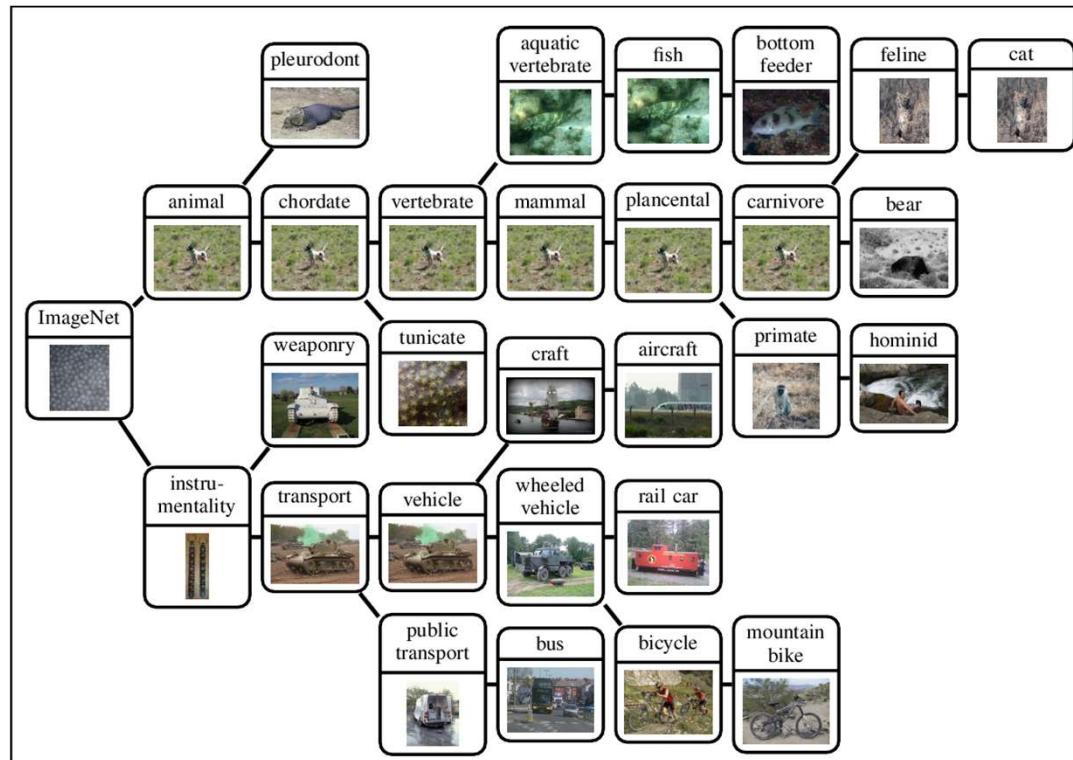
6	8
3	4

Fully Connected Layer (FC layer)



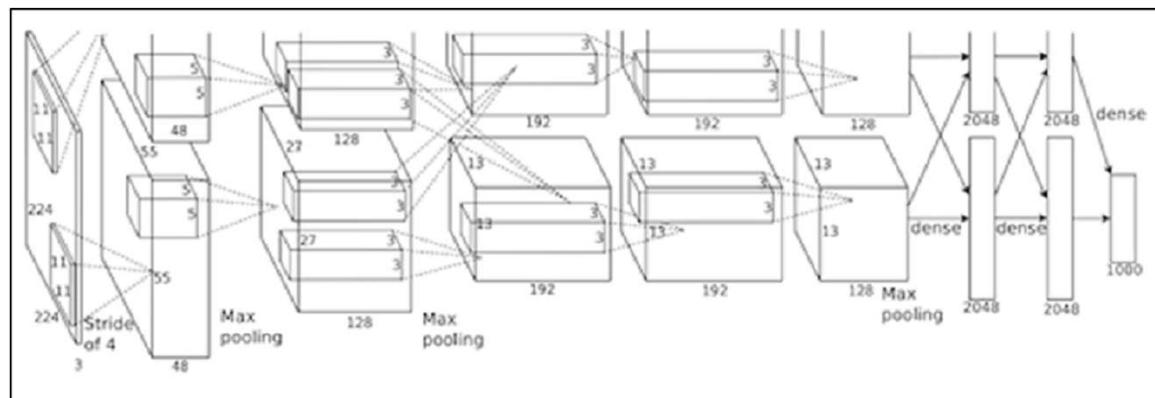
*Andrej Karpathy's recent presentation

IMAGENET



*<http://groups.inf.ed.ac.uk/calvin/imagenet/prototypes.html>

AlexNet

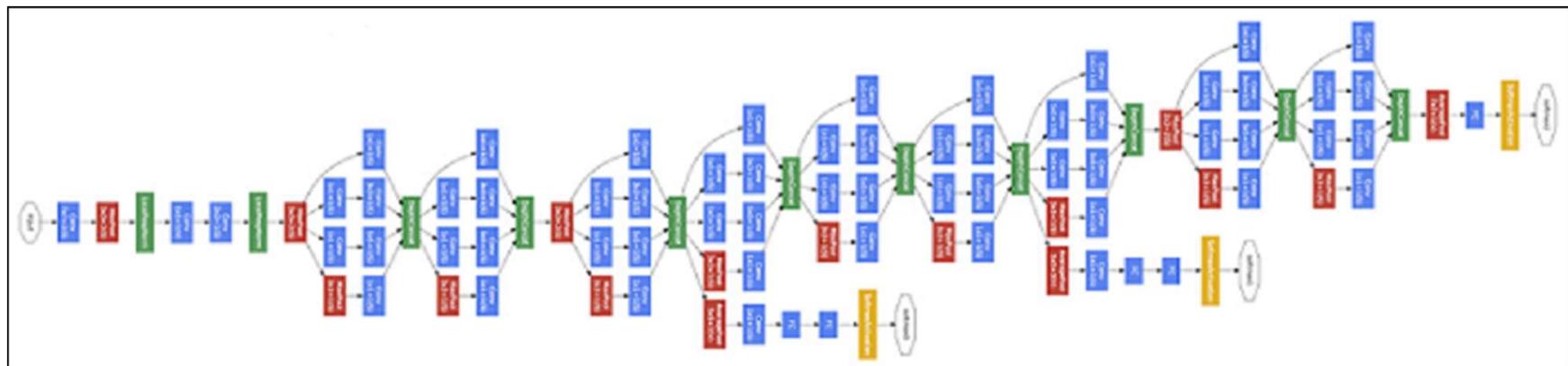


*<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

VGGNet

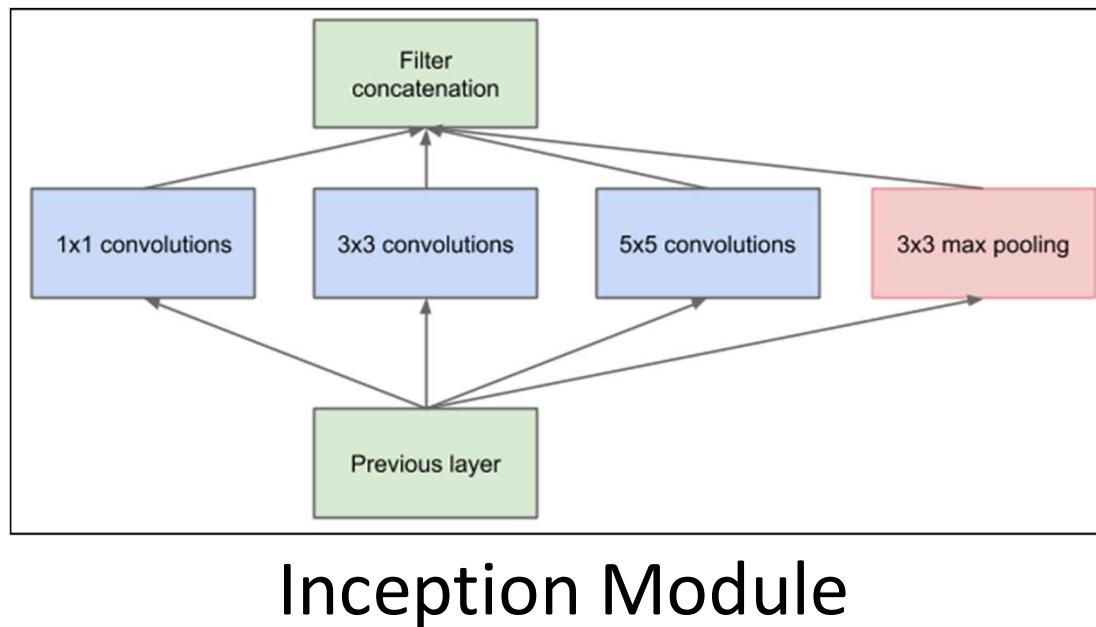
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

GoogLeNet

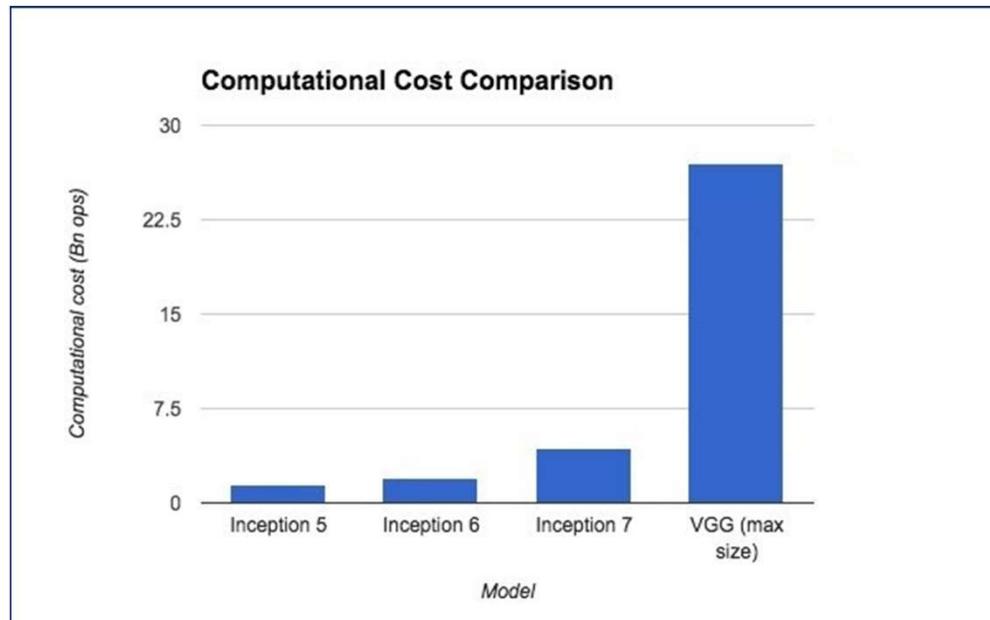


*http://www.csc.kth.se/~roelof/deepdream/bvlc_gnet.html

GoogLeNet uses Inception

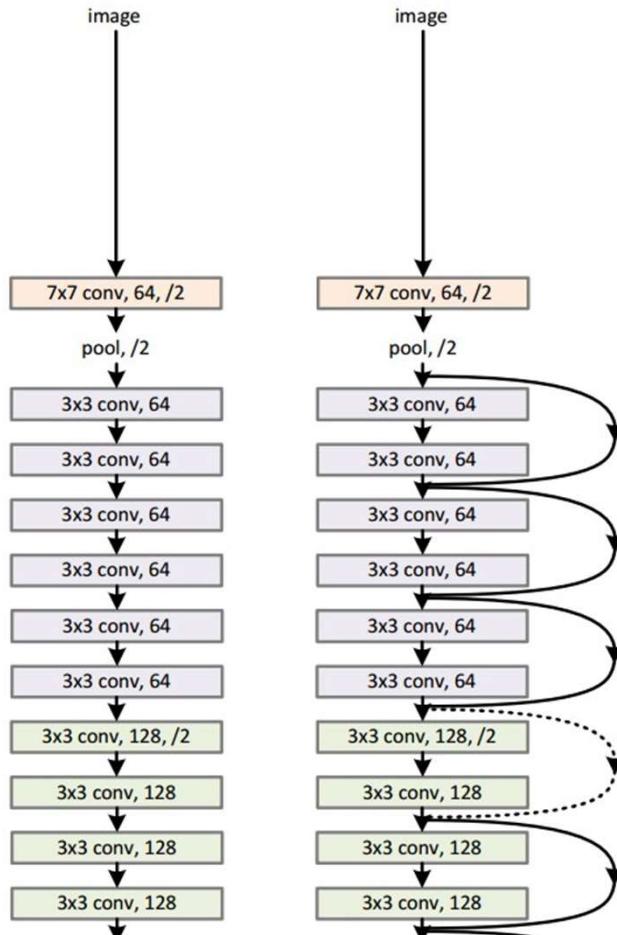


Inception Performance comparison



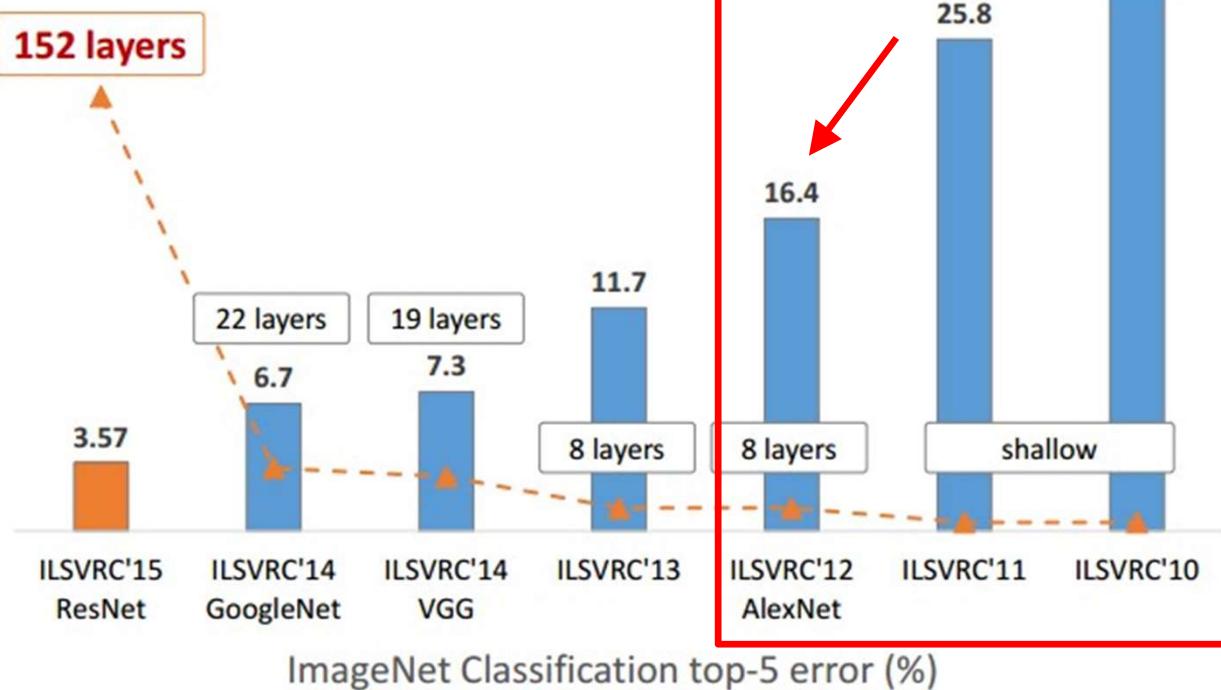
Microsoft ResNet

34-layer plain 34-layer residual



*<https://arxiv.org/pdf/1512.03385v1.pdf>

Revolution of Depth



Deep learning and natural language processing

Deep learning enables sub-symbolic processing

I <i>
bought <bought>
a <a>
car <car>
. < . >

You have to remember to represent “purchased” and “automobile.”

What about “truck”?

How do you encode the meaning of the entire sentence?

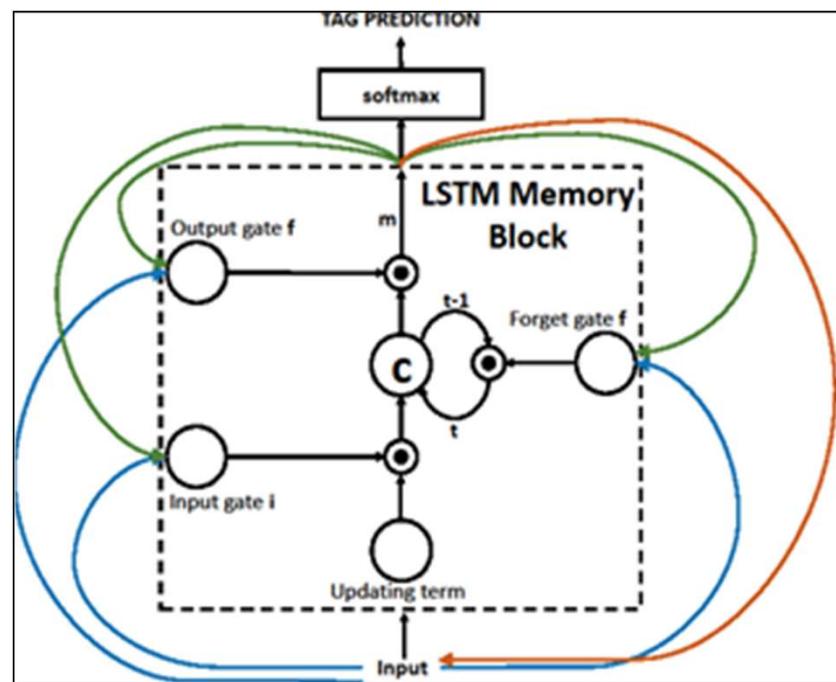
But what about a sentence?

Algorithm for generating vectors for sentences

1. Make the sentence vector be the vector for the first word.
2. For each subsequent word, combine its vector with the sentence vector.
3. The resulting vector after the last word is the sentence vector.

Can be implemented using a recurrent neural network (RNN)

LSTM RNN



What can you do with a Sentence Vector?

You can feed it to a classifier.

You can unwind in the other direction to do machine translation.

Called a seq2seq model, or Neural Machine Translation, or encoder-decoder model.

Deep learning and question answering

Bob went home.

Tim went to the junkyard.

Bob picked up the jar.

Bob went to town.

Where is the jar? A: town

Memory Networks [Weston et al., 2014]: Updates memory vectors based on a question and finds the best one to give the output.

The office is north of the yard.

The bath is north of the office.

The yard is west of the kitchen.

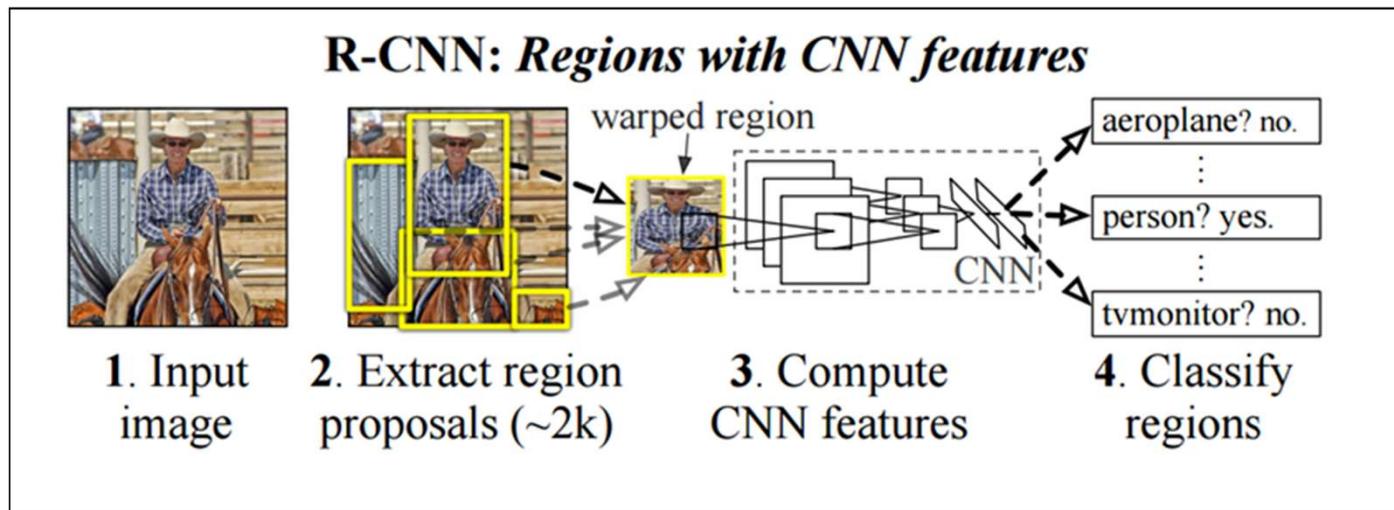
How do you go from the office to the kitchen? A: south, east

Neural Reasoner [Peng et al., 2015]: Encodes the question and facts in many layers, and the final layer is put through a function that gives the answer.

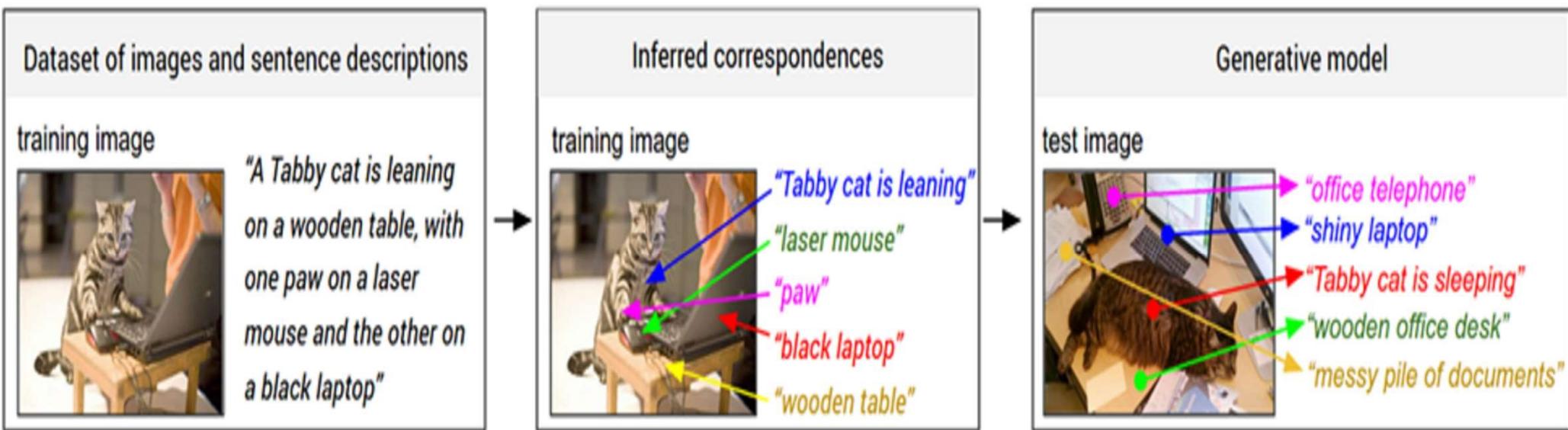
For details, please attend my session tomorrow on “Transfer Learning Text”

Other commonly used DNNs

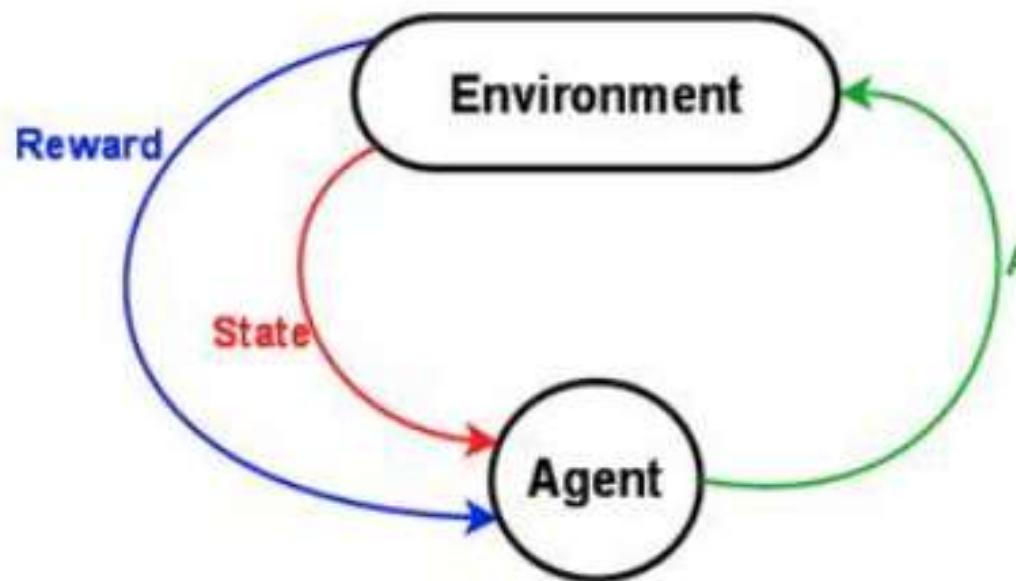
Region Based CNN (RCNN)



Generating Image Descriptions (CNN-RNN)



Deep Reinforcement Learning

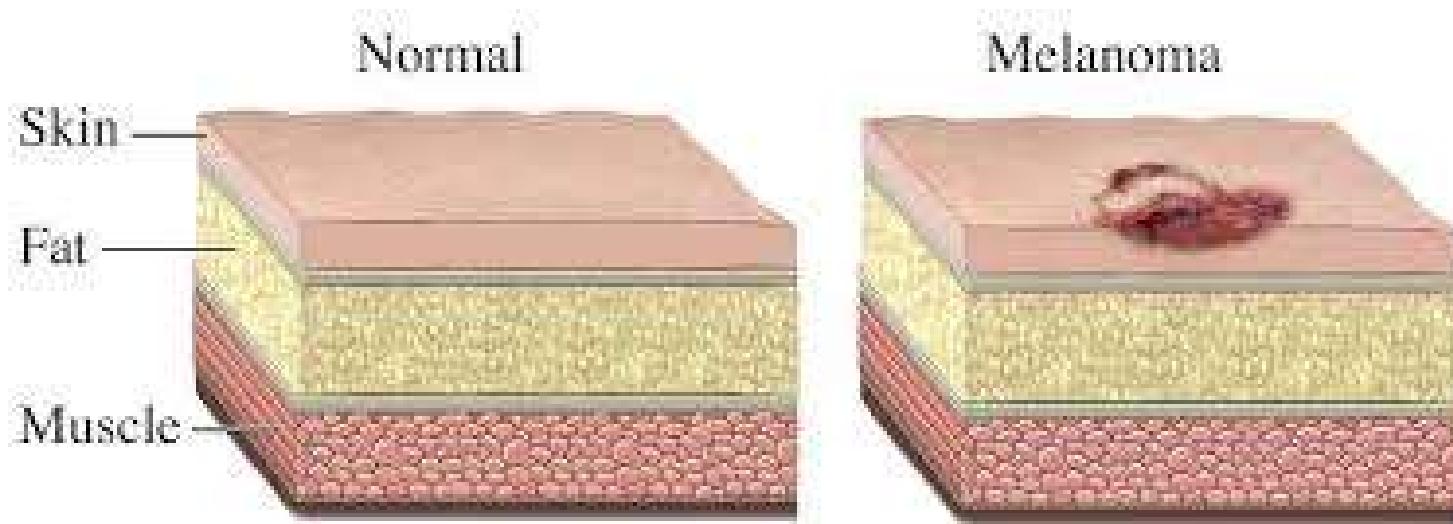


Deep Learning – Some path breaking use-cases

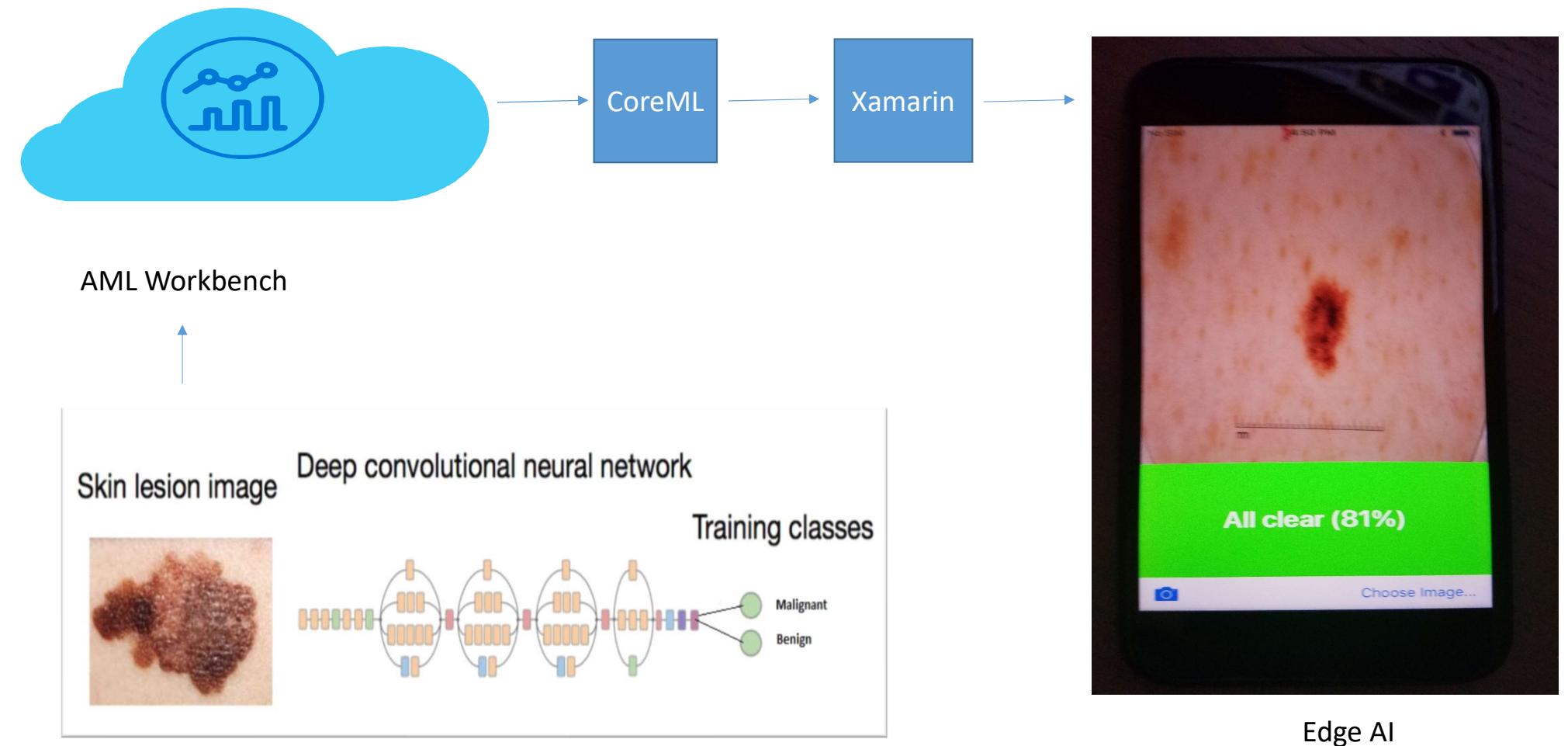
Advanced Melanoma Screening and Detection

- ❑ Researchers at the University of Michigan are putting advanced image recognition to work, detecting one of the most aggressive, but treatable in early stages, types of cancer.
- ❑ The team trained a neural network to isolate features (texture and structure) of moles and suspicious lesions for better recognition and detecting melanoma known to date.

Skin Cancer



Realtime Edge AI: Skin Cancer Detection



Neural Networks in Finance

- ❑ Trading and risk management are two areas where we would expect to see developments for neural networks
- ❑ Popular technical indicators along with neural networks and genetic algorithms are used to generate buy and sell signals for each stock and for portfolios of stocks

Example: Predicting the 2016 Presidential Election, and the Resulting Market Impact

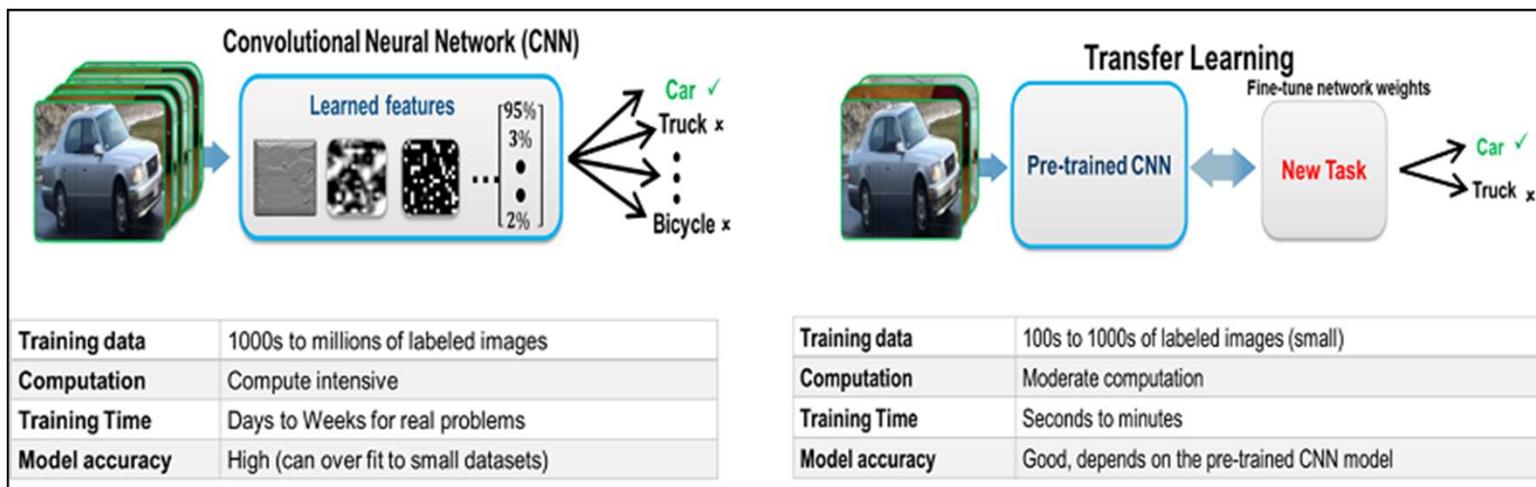
- Use a DNN and features to predict the Trump election and the expected market reaction
- Python utilizing Tensorflow
- https://github.com/athaker/econ_136
- Identify the sources of data and the resulting outputs
- What features do you think are most important to the model?

Candidate	Party	Electoral Votes	Popular Votes
Donald J. Trump	Republican	304	62,980,160
Hillary R. Clinton	Democratic	227	65,845,063
Gary Johnson	Libertarian	0	4,488,931
Jill Stein	Green	0	1,457,050
Evan McMullin	Independent	0	728,830



Increasing Re-usability of Deep Learning models

Transfer Learning & Fine-tuning

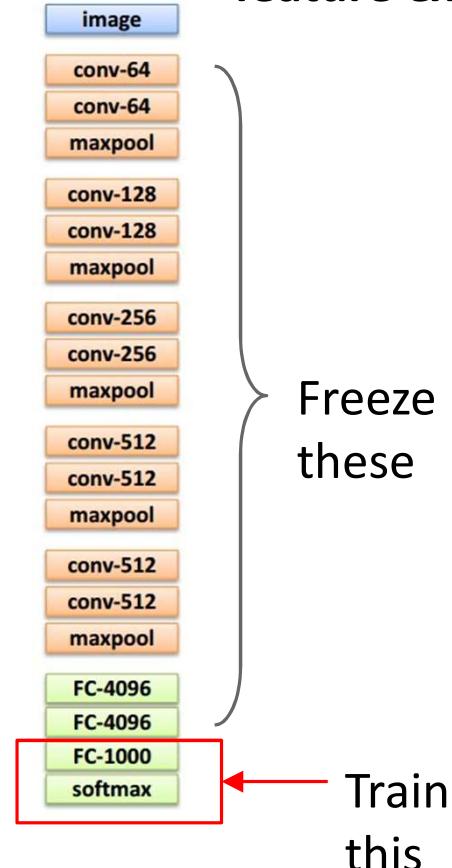


Transfer Learning

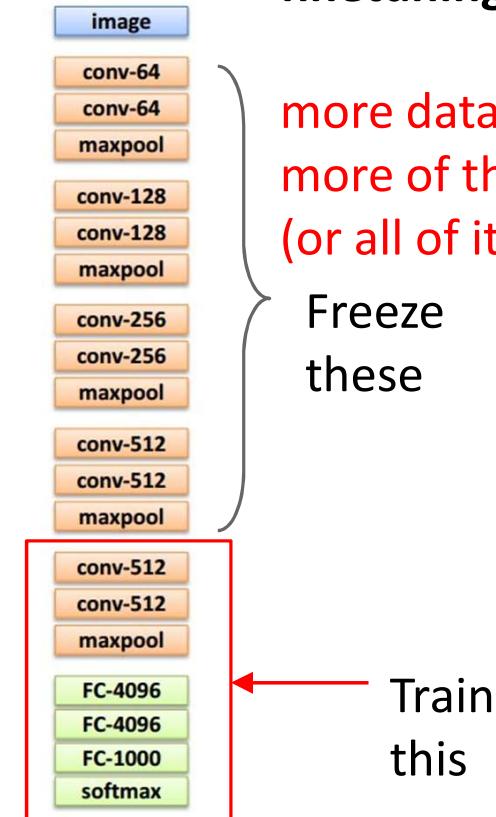
1. Train on
Imagenet



2. Small dataset:
feature extractor



3. Medium dataset:
finetuning



more data = retrain
more of the network
(or all of it)

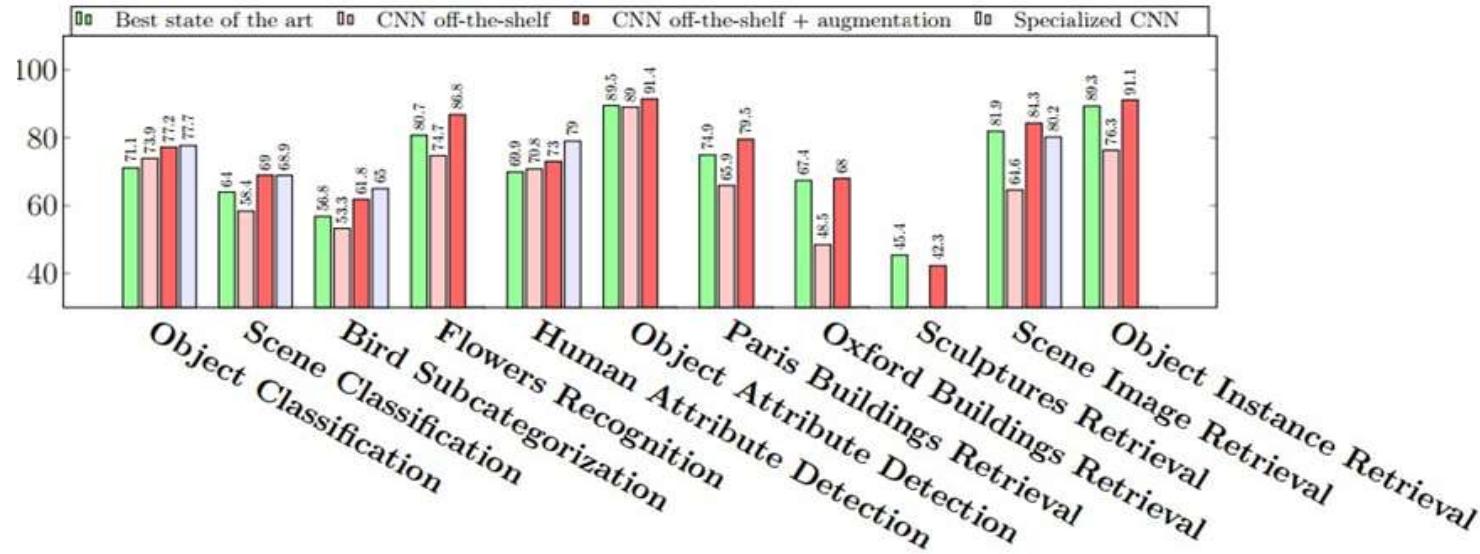
Freeze
these

Train
this

*Andrej Karpathy's recent presentation

Transfer Learning

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



*Andrej Karpathy's recent presentation

GPUs in Azure

Microsoft Azure

Why Azure Solutions Products Documentation Pricing Partners [Blog](#) Resources Support

Azure N-Series preview availability

Posted on August 4, 2016

[Corey Sanders](#), Director of Program Management, Azure

Today we're delighted to announce that Azure N-Series Virtual Machines, the fastest GPUs in the public cloud, are now available in preview. N-Series instances are enabled with NVIDIA's cutting edge GPUs to allow you to run GPU-accelerated workloads and visualize them. These powerful sizes come with the agility you have come to expect from Azure, paying per-minute of usage.

Our N-Series VMs are split into two categories. With the NC-Series (compute-focused GPUs), you will be able to run compute intensive HPC workloads using CUDA or OpenCL. This SKU is powered by Tesla K80 GPUs and offers the fastest computational GPU available in the public cloud. Furthermore, unlike other providers, these new SKUs expose the GPUs through discreet device assignment (DDA) which results in close to bare-metal performance. You can now crunch through data much faster with CUDA across many scenarios including energy exploration applications, crash simulations, ray traced rendering, deep learning and more. The Tesla K80 delivers 4992 CUDA cores with a dual-GPU design, up to 2.91 Teraflops of double-precision and up to 8.93 Teraflops of single-precision performance. Following are the Tesla K80 GPU sizes available:

	NC6	NC12	NC24
Cores	6	12	24
(E5-2690v3)		(E5-2690v3)	(E5-2690v3)
GPU	1 x K80 GPU (1/2 Physical Card)	2 x K80 GPU (1 Physical Card)	4 x K80 GPU (2 Physical Cards)
Memory	56 GB	112 GB	224 GB
Disk	380 GB SSD	680 GB SSD	1.44 TB SSD

In addition to the NC-Series, focused on compute, the NV-Series is focused more on visualization. Data movement has traditionally been a challenge with HPC scenarios using large datasets produced in the cloud. With the Azure NV-Series, you'll be able to use Tesla M60 GPUs and NVIDIA GRID in Azure for desktop accelerated applications and virtual desktops. With these powerful visualization GPUs in Azure, you will be able to visualize and interact with your data faster, more easily, and more intuitively.



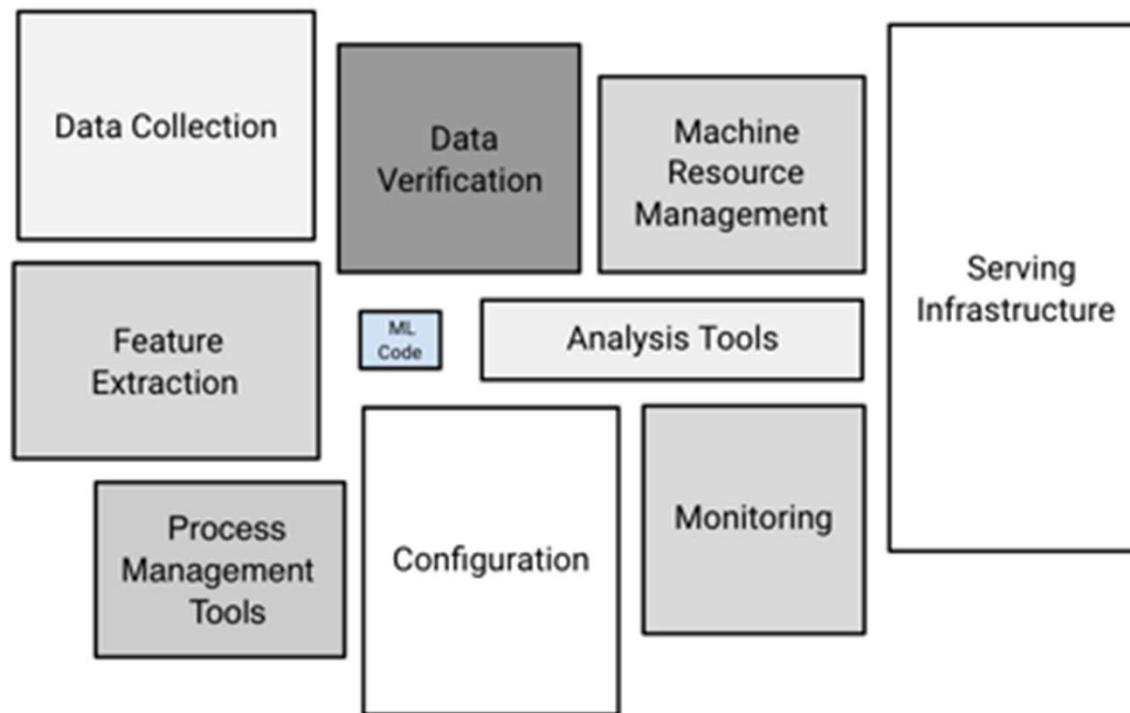
Fastest GPUs in the public cloud

Azure GPU VM

- ❑ NVIDIA Tesla K80
- ❑ NVIDIA P100
- ❑ NVIDIA V100

```
██████████ @AzureGPUCluster:-$ nvidia-smi
Mon Sep 26 22:38:34 2016
+-----+
| NVIDIA-SMI 367.18      Driver Version: 367.18 |
+-----+
| GPU  Name      Persistence-M| Bus-Id      Disp.A  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|-----+
|  0  Tesla K80          Off  | 80F9:00:00.0  Off   |                0 |
| N/A   43C    P8    28W / 149W |     0MiB / 11439MiB |    0%     Default |
+-----+
|  1  Tesla K80          Off  | 9A76:00:00.0  Off   |                0 |
| N/A   35C    P8    34W / 149W |     0MiB / 11439MiB |    0%     Default |
+-----+
|  2  Tesla K80          Off  | 9CC7:00:00.0  Off   |                0 |
| N/A   42C    P8    26W / 149W |     0MiB / 11439MiB |    0%     Default |
+-----+
|  3  Tesla K80          Off  | A5DB:00:00.0  Off   |                0 |
| N/A   41C    P0    74W / 149W |     0MiB / 11439MiB |    0%     Default |
+-----+
+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name        Usage        |
|-----+
| No running processes found            |
+-----+
```

ML/DL is Only a Small Fraction, Toolkits speed up the process



<https://developers.google.com/machine-learning/crash-course>

Toolkits for Practical re-usability of DL Models

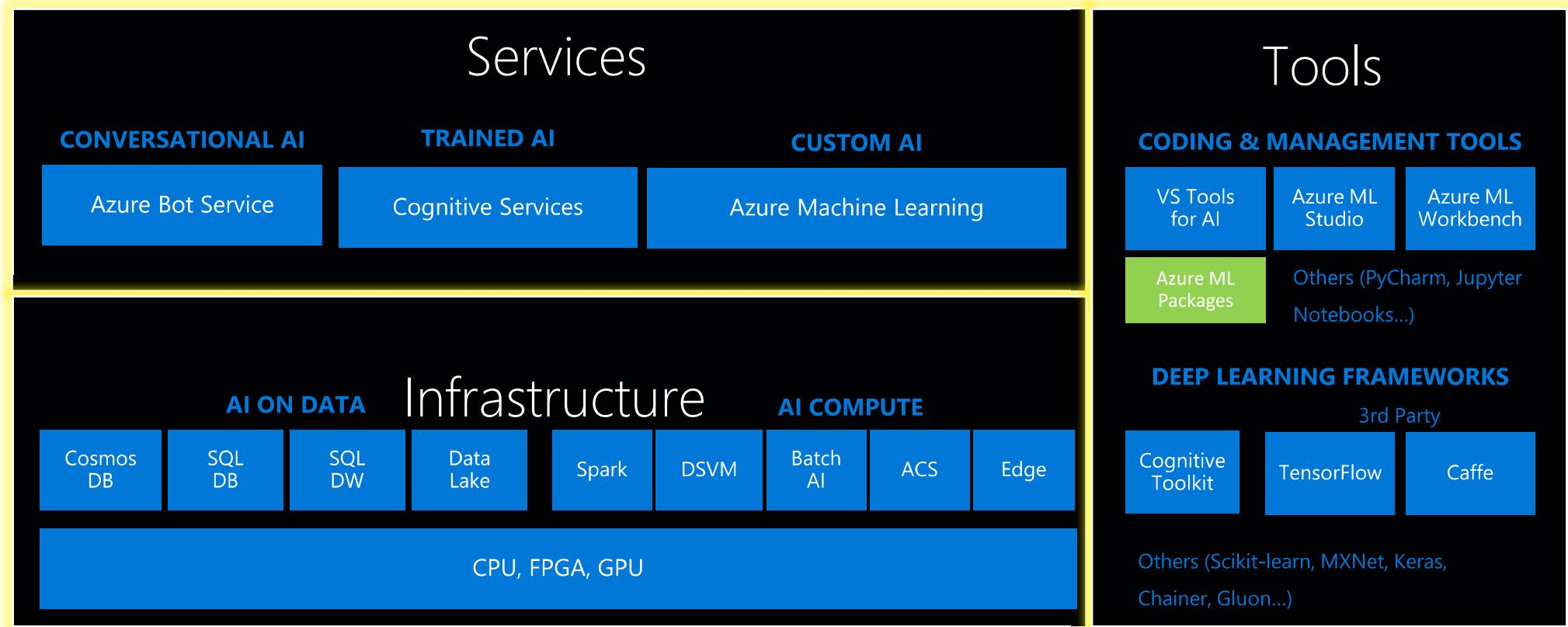


Azure ML Packages



The Microsoft AI platform

Cloud-powered AI for every developer



What are Azure ML Packages?

- Python packages extensions for Azure Machine Learning enabling data scientist and AI developers to build high AI quality pipelines for

Computer
Vision

Text
Analytics

Forecasting

- Provides high level APIs for data preparation, augmentation, training, evaluating and deployment.
- Model experimentation and comparison, run history, model management and deployment through AML Workbench

Easily build and deploy models on Microsoft AML platform

Value Proposition

Accurate

Packages support State of the Art algorithms and include optimized parameters to provide out of the box highly accurate models.

Flexible

Powerful and composable Python API. Optimize the best algorithm for your application. Provides control over all the aspects that are available to the power user.

Scalable

Train models at large scale with multi GPU

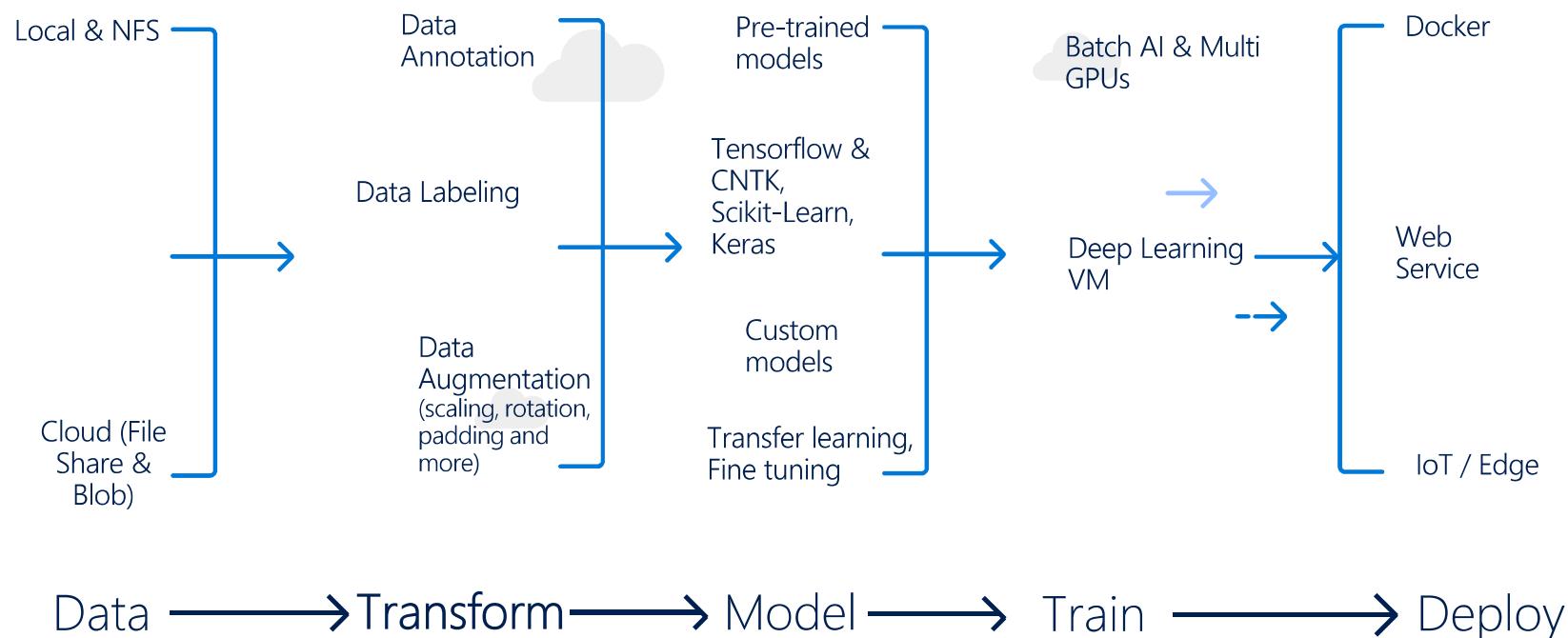
Time to Solution

Automates many tedious tasks (data alignment, cleaning and preparation), easily model experimentations and deploy in production.

Best in class package in Quality, Flexibility, Scale and Time to solution.

Deep Dive on AML Package For Computer Vision

AML Packages - Pipeline



Sample Code (less than 20 lines)

```
# create a dataset from a directory with folders representing different classes
dataset = Dataset.create_from_dir(dataset_name, dataset_location)

# split the full dataset into a train and test set
train_set_orig, test_set = Splitter(dataset).split(train_size=.8, stratify='label')

# Image augmentation
rotate_and_flip = augmenters.Sequential([augmenters.Affine(rotate=(-45, 45)), augmenters.Fliplr(.5)])
crop = augmenters.Sequential([augmenters.Crop(percent=(0, .1))])
train_set = augment_dataset(train_set_orig, [rotate_and_flip, crop])

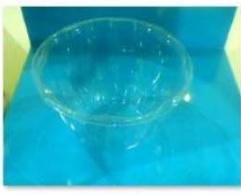
# create the model
model = TransferLearningModel(num_classes = num_classes, base_model_name = 'ResNet18_ImageNet_CNTK')

# train the model using cntk
num_epochs = 45
mb_size = 32
t = Trainer(model, train_set, num_epochs=num_epochs, mb_size=mb_size)
t.start_training()
```

Image Classification

Prepare Dataset

1



bowl



cup



cutlery

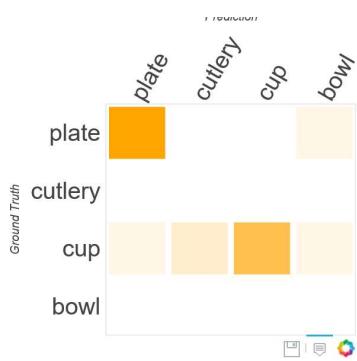
2

Model Definition & Training

```
# CVTK default parameters (224 x 224 pixels resolution, Resnet-18)
lr_per_mb = [0.05]*7 + [0.005]*7 + [0.0005]
mb_size = 32
input_resoluton = 224
base_model_name = "ResNet18_ImageNet_CNTK"
```

3

Evaluate the model



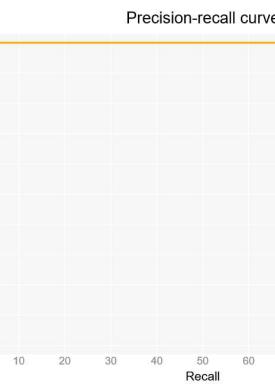
Score the model

```
# Parse result from json string
import numpy as np
parsed_result = CNTKTLModel.parse_serialized_result(r.text)
print("Parsed result:", parsed_result)
# Map result to image class
class_index = np.argmax(np.array(parsed_result))
print("Class index:", class_index)
dnn_model.class_map
print("Class label:", dnn_model.class_map[class_index])
```

Parsed result: '[0.07414770871400833, -0.03261350095272064, -2.57250714302063, Class index: 0 Class label: bowl



4



5

Deploy the model

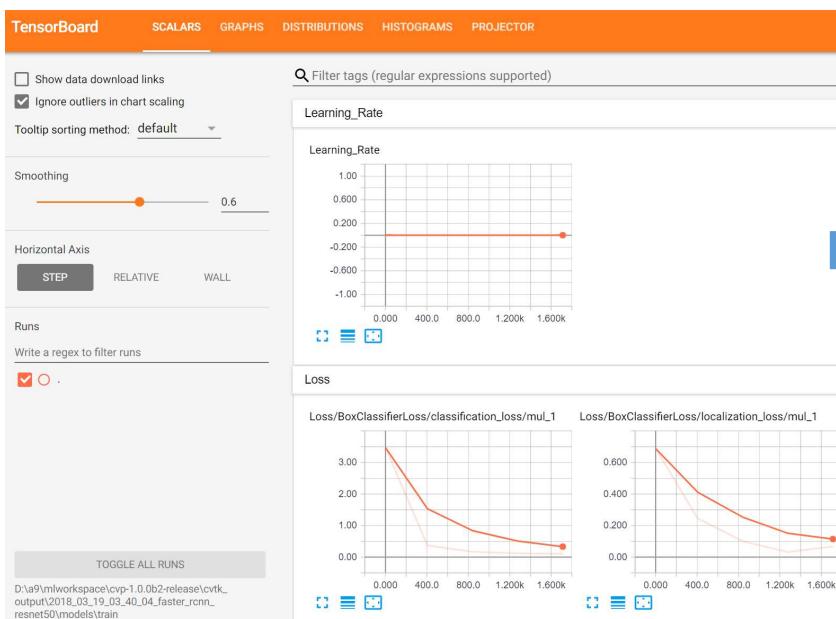
Deploy the model as an Azure web service, Dockers Container or IoT Edge

The evaluation module provides functionality to evaluate the performance of the trained model. Some of the evaluation metrics :
accuracy, precision and recall, confusion matrix

Object Detection

Object Detection is one of the main problems in Computer Vision. The Computer Vision Package makes it easy to perform all these steps easily and with high accuracy , and internally uses Tensorflow's implementation of Faster R-CNN

Train the model



Score an image

```
image_path = '../sample_data/liebherr/liebherr_val/JPEGImages/WIN_20160803_11_46_03_Pro.jpg'  
# scores = my_detector.score_image(frozen_model_path, image_path)  
detections_dict = detection_utils.score(frozen_model_path, image_path)
```

Object 0: label=butter , score=0.99,
Object 1: label=tomato , score=0.97,
Object 2: label=gerkin , score=0.96,
Object 3: label=pepper , score=0.95,
Object 4: label=eggBox , score=0.94,
Object 5: label=tomato , score=0.94,
Object 6: label=avocado , score=0.94,
Object 7: label=champagne , score=0.59,

Found 8 objects in image ../sample_data/



Computer Vision Simplified in Python

High Accuracy out of the box

Include optimized parameters and pre trained weights to provide out of the box highly accurate models.

```
# CVTK default parameters (224 x 224 pixels resolution, Resnet-18)
lr_per_mb = [0.05]*7 + [0.005]*7 + [0.0005]
mb_size = 32
input_resoluton = 224
base_model_name = "ResNet18_ImageNet_CNTK"

# CVTK suggested parameters for 500 x 500 pixels resolution, Resnet-50
# (see in the Appendix "How to improve accuracy", last row in table)
# lr_per_mb = [0.01] * 7 + [0.001] * 7 + [0.0001]
# mb_size = 8
# input_resoluton = 500
# base_model_name = "ResNet50_ImageNet_CNTK"

# Initialize model
dnn_model = CNTKTLModel(train_set.labels,
                        base_model_name=base_model_name,
                        image_dims = (3, input_resoluton, input_resoluton))
```

Time To Solution

Get from a data set to a highly accurate deployed model quickly

```
# Optional azure machine learning deployment cluster name (environment name) and resource group
# If you don't provide here, It will use the current deployment environment (you can check with
azureml_rscgroup = "<resource group>"
cluster_name = "<cluster name>"

# If you provide the cluster information, it will use the provided cluster to deploy.
# Example: deploy_obj = AMLDeployment(deployment_name=deployment_name, associated_DNNModel=dnn_m
# aml_env="cluster", cluster_name=cluster_name, resource_group=azurem

# Create deployment object
deploy_obj = AMLDeployment(deployment_name=deployment_name, aml_env="cluster", associated_DNNMod

# Check if the deployment name exists, if yes remove it first.
if deploy_obj.is_existing_service():
    AMLDeployment.delete_if_service_exist(deployment_name)

# create the webservice
print("Deploying to Azure cluster...")
deploy_obj.deploy()
print("Deployment DONE")
```

What customers are saying ?



“Using the AML Package for Computer Vision we were able to reach 97% accuracy on our production images and deploying the models to the Azure Cluster Servers is incredibly fast and easy ” (Jocelyn Desbiens, Ph.D)

“The Azure Machine Learning Package for Computer Vision package different parts (data load, annotation, split, augmentation, training, evaluation, etc.) into separate modules, which reduces the coding effort for data scientist when building the model. In addition, the visualization utils provide a straightforward and visual understanding of the image data and model evaluation results.” (MS Data Scientist)

“The Azure Machine Learning Package for Computer Vision model out of the box with 10 epochs outperforms 4 out of the 6 human experts on a cell identification problem.” (MS ADS Data Sceintist)

Demo – Object Detection

Drone Demo –Powered by AML Packages

<https://youtu.be/EruH-4Fvecc?t=249>

Thank you. Questions?