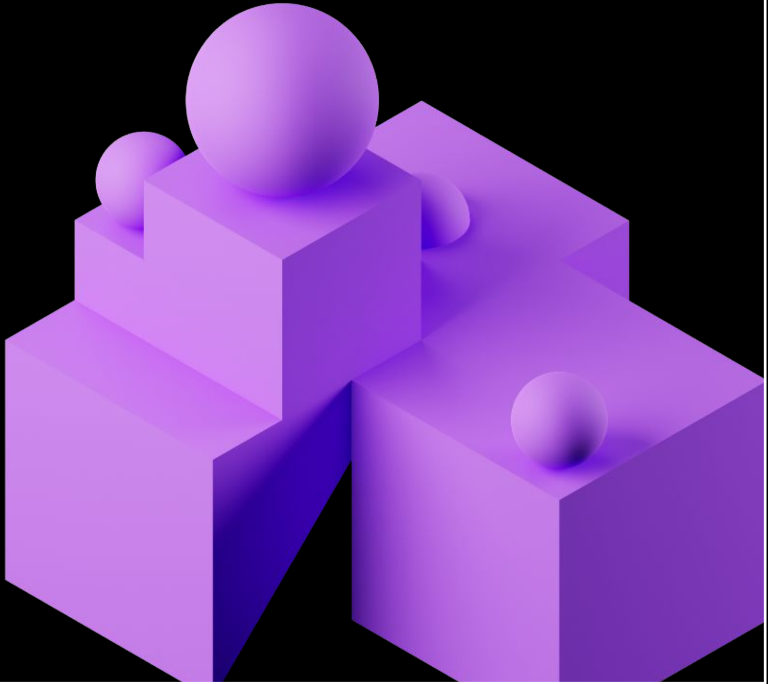
DATA+AI SUMMIT BY databricks

Deep Learning with Databricks



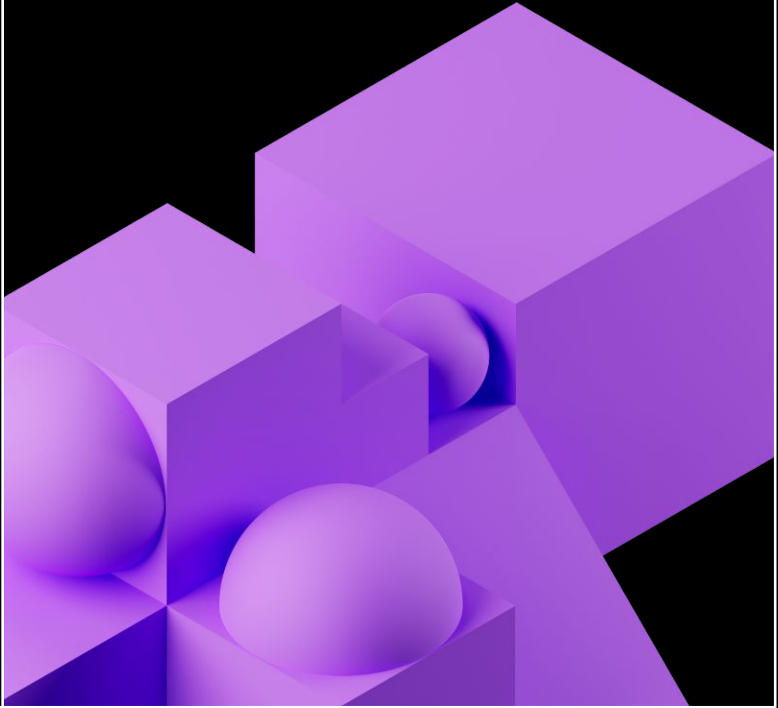
Databricks
2023

Agenda

Deep Learning with Databricks		
	Demo	Lab
Introduction		
Linear Regression with Keras	✔	✔
Callbacks	✔	✔
ML*flow*	✔	✔
Hyperopt	✔	✔
Horovod + Petastorm	✔	✔
Convolutional Neural Networks	✔	
Transfer Learning	✔	✔

LET'S GET STARTED

Deep Learning Overview



Why Deep Learning?

- Performs well on complex datasets like images, sequences, and natural language
- Performs better as data amount increases
- Theoretically can learn any relationship (universal approximation theorem)

Where Does DL Fit In?

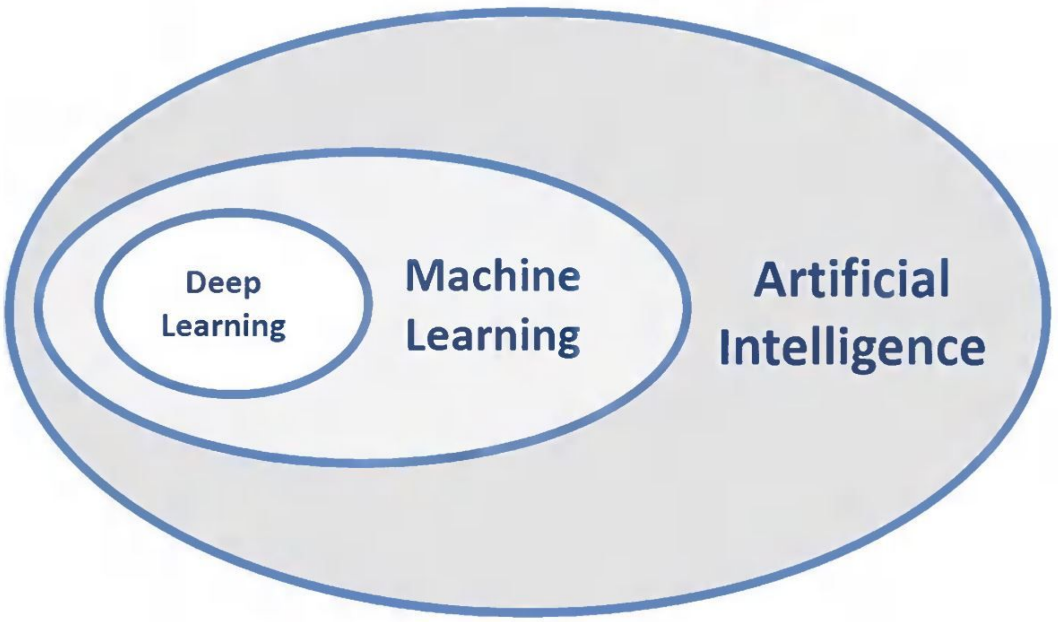
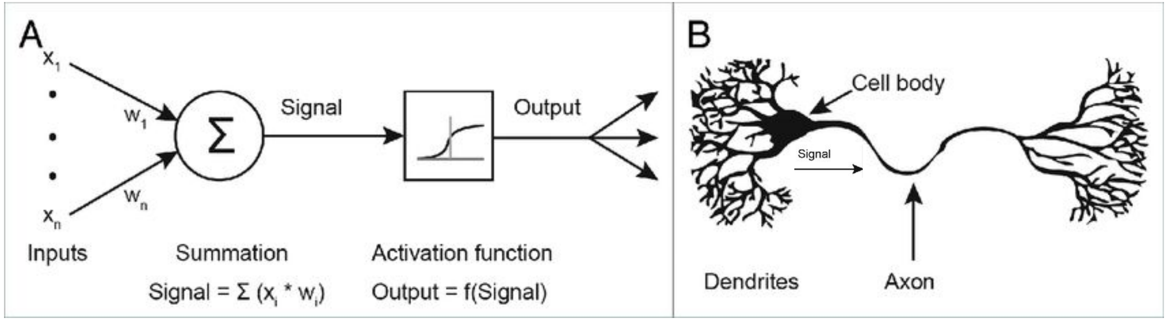


Image source

What Is Deep Learning?

Features learned from data using basic building blocks inspired by the human brain



A) An ***artificial*** neuron

B) A ***biological*** neuron

Image source

Keras

- High-level Python API to build neural networks
- Official high-level API of TensorFlow
- Has over 250,000 users
- Released by François Chollet in 2015
- **Sequential API** (common method) and **Functional API** (to define complex topologies)

Sequential API Example

```python
import tensorflow as tf
from tensorflow. keras import datasets, layers, models

# Load dataset
mnist = datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Construct a neural network model
model = models. Sequential ()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation=tf.nn.relu))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(10, activation=tf.nn.softmax))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

# Train and evaluate the model
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Source

Hardware Considerations

- GPUs are often the tool of choice for DL training due to their superior parallel execution abilities over CPUs.
- CPUs are easier to program, faster for smaller datasets, and allow more flexibility of use.
- *We'll be using CPUs in this course because they are cheaper for learning.*

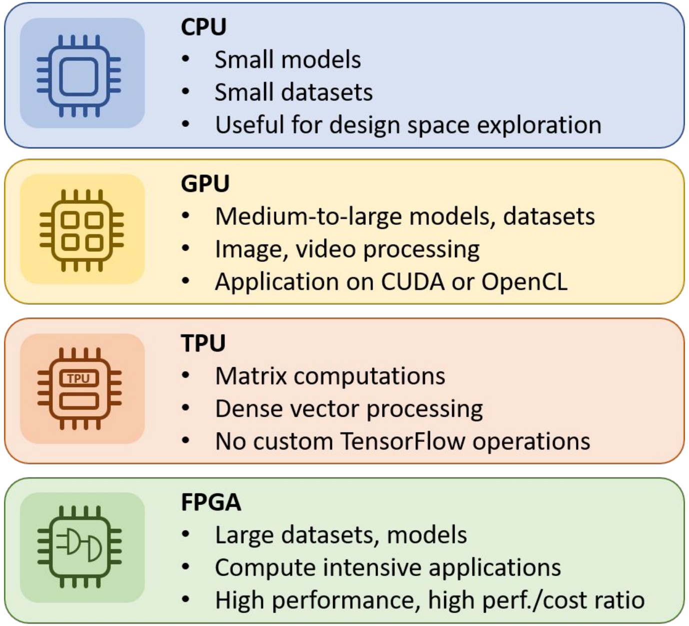
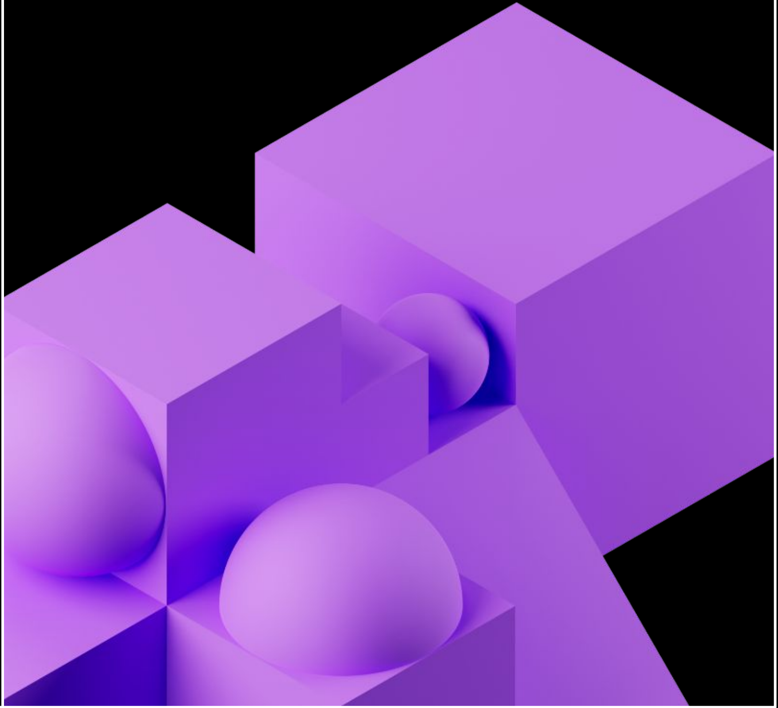


Image source

Neural Network Fundamentals



Layers

- Input layer
- Zero or more hidden layers
- Output layer

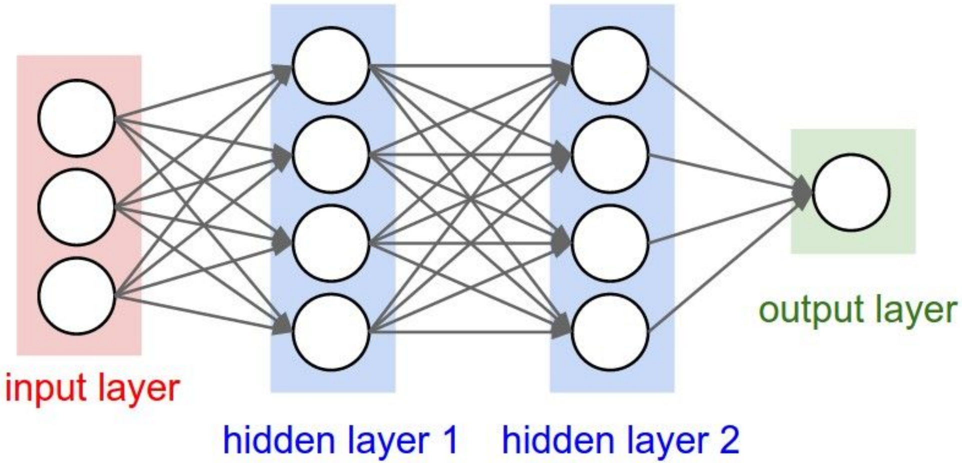


Image source

Backpropagation

While training

- Take an example input
- Feed it forward through the network
- Calculate the error in the prediction
- Use the error to correct the previous layers

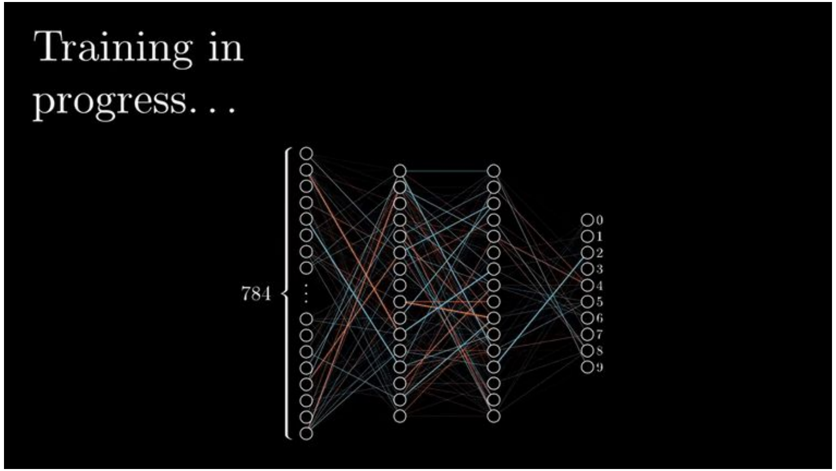
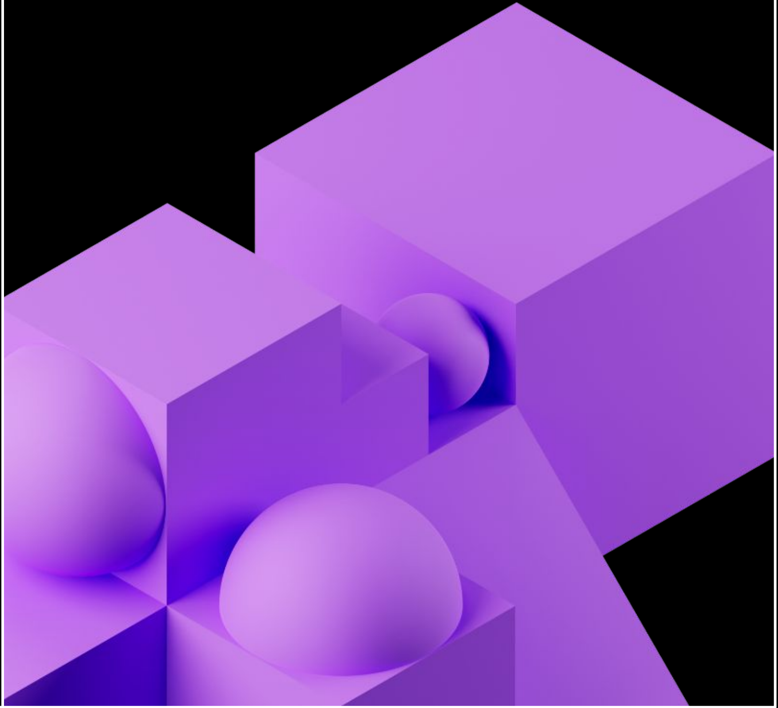


Image source

Linear Regression with Keras



Activation Functions

Common activation functions in neural networks

Provide non-linearity in our neural networks to learn more complex relationships

- Sigmoid
- Tangent
- ReLU (**R**ectified **L**inear **U**nit)
- Leaky ReLU
- Softmax
- ELU (**E**xponential **L**inear **U**nit)

Sigmoid

Activation function

- Saturates and kills gradients
- Not zero-centered along the y-axis
- Last layer activation function for binary classification models

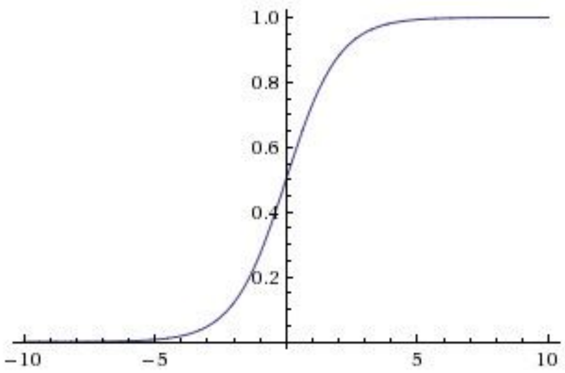
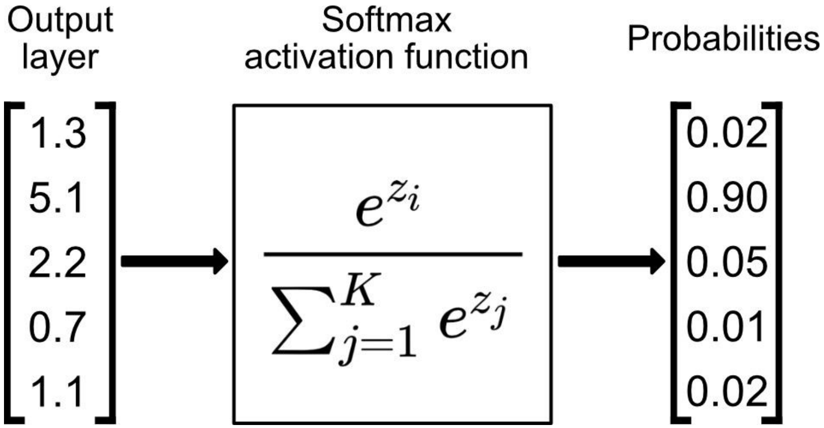


Image source

Softmax

Activation function

- Convert logits to probabilities
- Last layer activation function for multi-class classification



Hyperbolic Tangent (Tanh)

Activation function

- Zero centered!
- BUT, like the sigmoid, its activations saturate

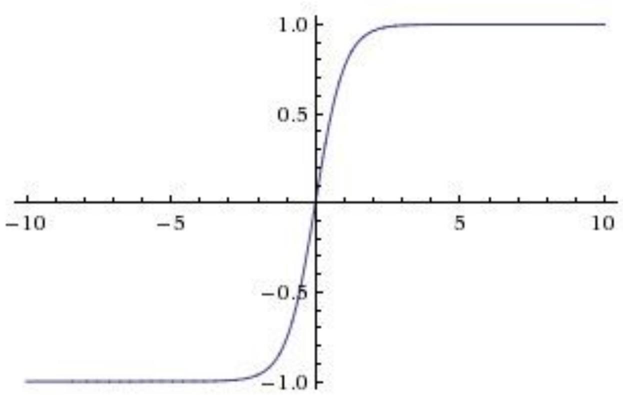
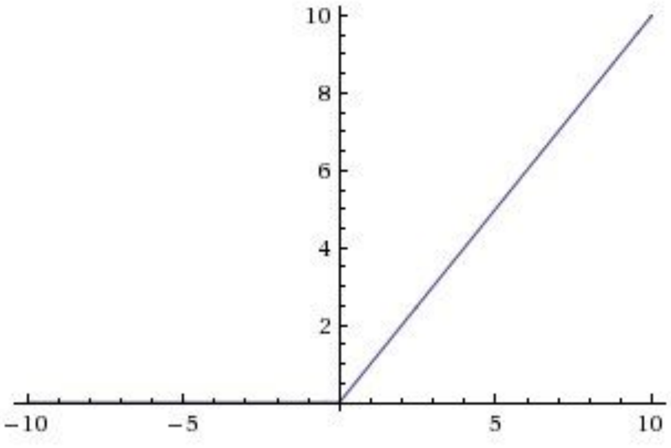


Image source

ReLU (Rectified Linear Unit)

Activation function

- BUT, gradients can still go to zero



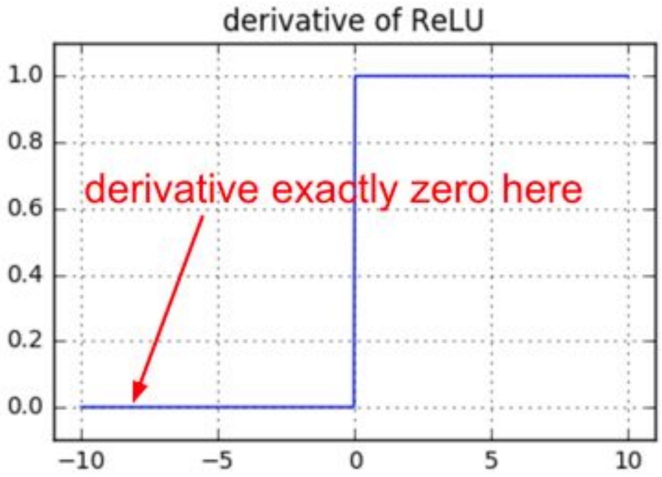
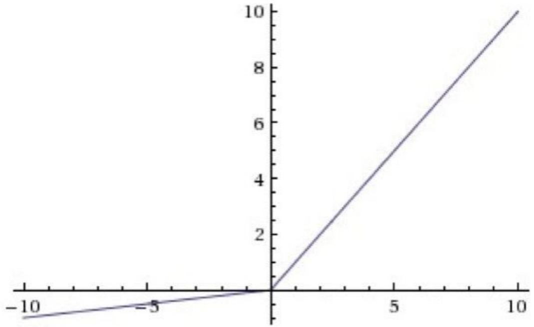


Image source

Leaky ReLU

Activation function

- For $x < 0$: $f(x) = \alpha * x$
- For $x >= 0$: $f(x) = x$



These functions are not differentiable at 0, so we set the derivative to 0 or average of left and right derivative.

Image source

Visual Comparison

Activation Functions

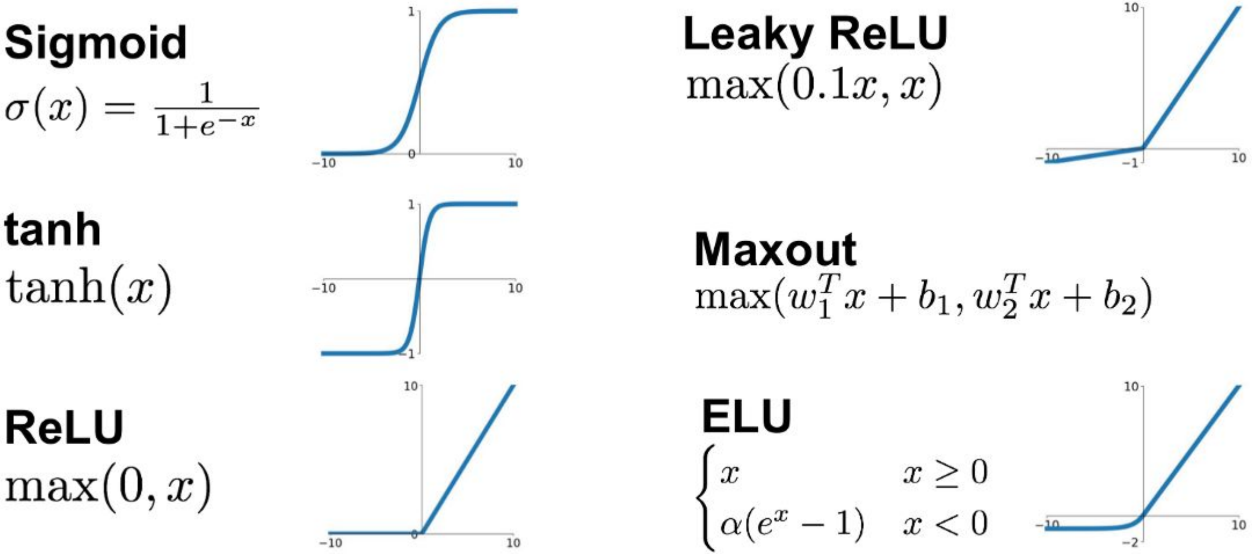
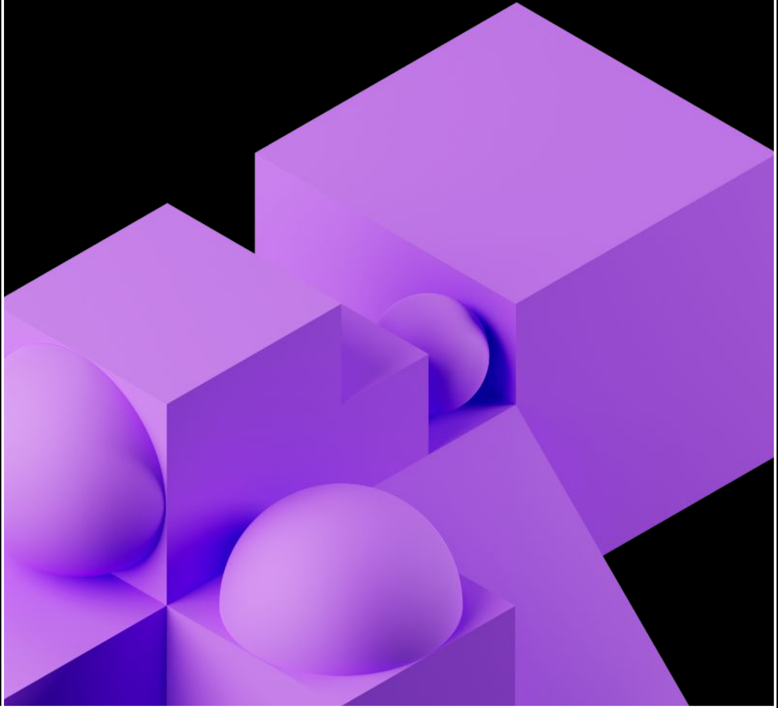


Image source

Optimizers



Stochastic Gradient Descent (SGD)

- Choosing a proper learning rate can be difficult

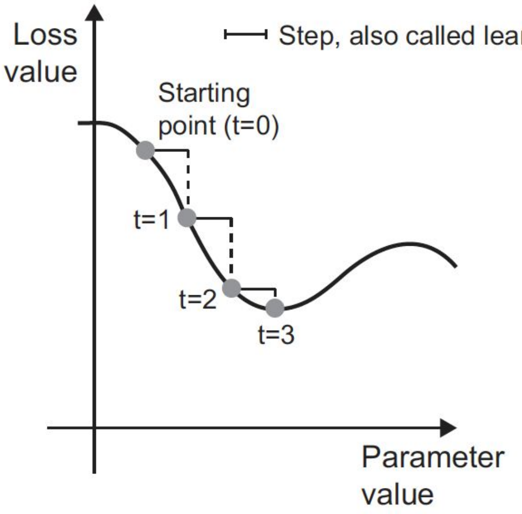


Figure 2.11 SGD down a 1D loss curve (one learnable parameter)

Image source

Stochastic Gradient Descent

- Easy to get stuck in local minima

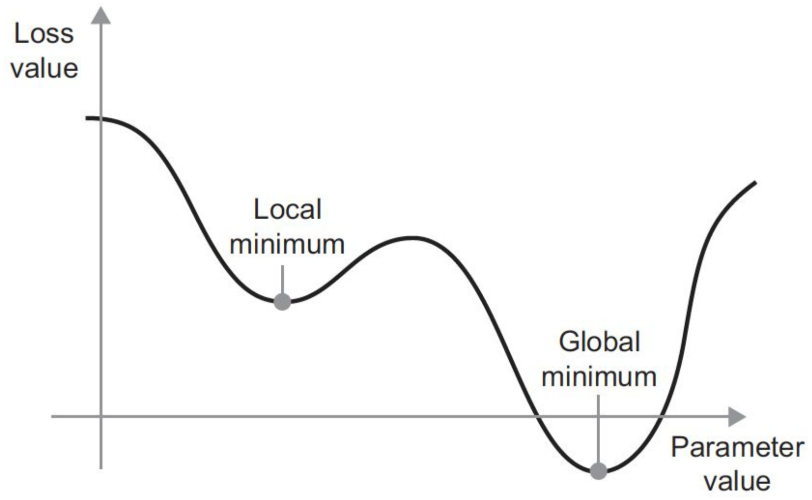


Figure 2.13 A local minimum and a global minimum

Image source

Momentum

- Accelerates SGD: Like pushing a ball down a hill
- Take average of direction we've been heading (current velocity and acceleration)
- Limits oscillating back and forth, gets out of local minima

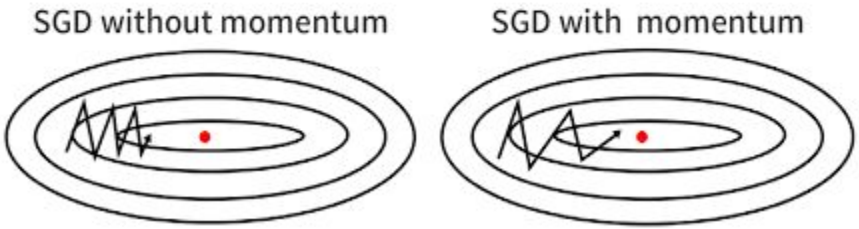


Image source

ADAM (ADAptive Moment Estimation)

Recommended default choice

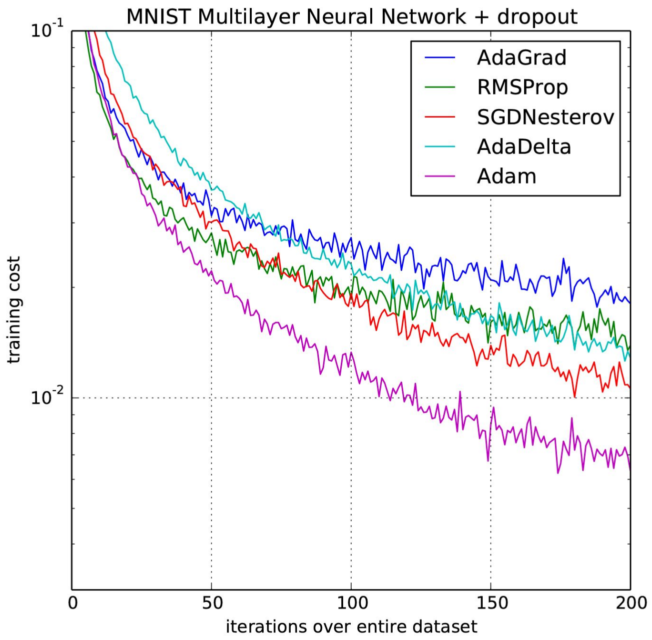


Image source

ADAM: Momentum + RMSProp

- Momentum
  - Uses a moving average of the gradient
- RMSProp
  - Divides the learning rate element-wise by the root of the moving average of the squared gradient
  - More uniform learning steps, small gradients are impactful, faster convergence

$$w_t = w_{t-1} - \frac{learning\ rate}{\sqrt{moving\ average\ of\ squared\ gradient}} * moving\ average\ of\ gradient$$

Image source

DEMO:

KERAS

Notebook: 01 Keras Fundamentals → 01.1-Linear Regression + 01.2-Keras

LAB:

KERAS

Notebook: 01-Keras Fundamentals → 01.2L-Keras Lab

DEMO:

CALLBACKS

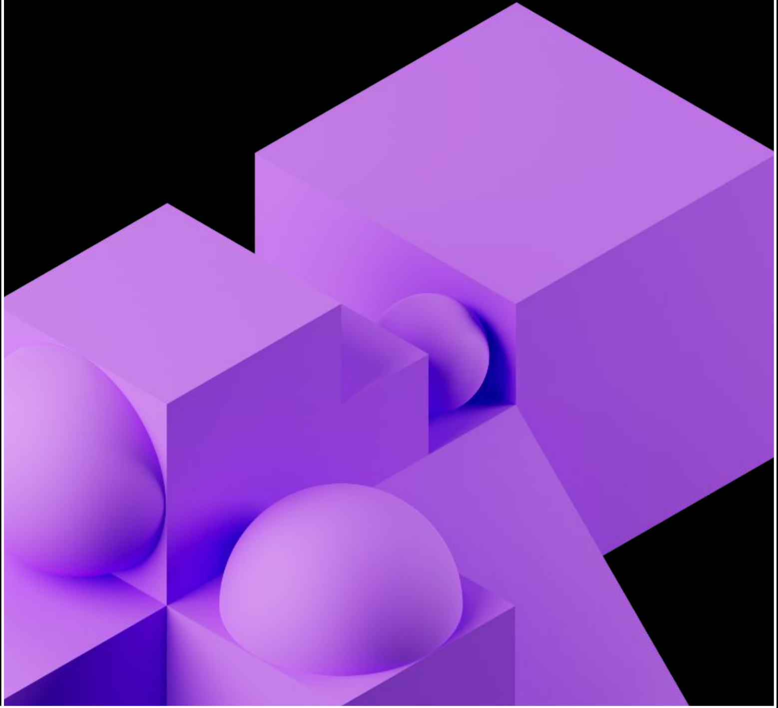
Notebook: 01 Keras Fundamentals → 01.4-Callbacks

LAB:

CALLBACKS

Notebook: 01 Keras Fundamentals → 01.4L - Callbacks Lab

ML*flow*



Core Machine Learning Issues

Modern ML lifecycle comes with many challenges

- Keeping track of experiments or model development
- Reproducing code
- Comparing models
- Standardization of packaging and deploying models

MLflow addresses these issues.

MLflow Components

The four components of MLflow

mlflow Tracking

Record and query experiments: code, data, config, results

mlflow Projects

Packaging format for reproducible runs on any platform

mlflow Models

General model format that supports diverse deployment tools

mlflow Model Registry

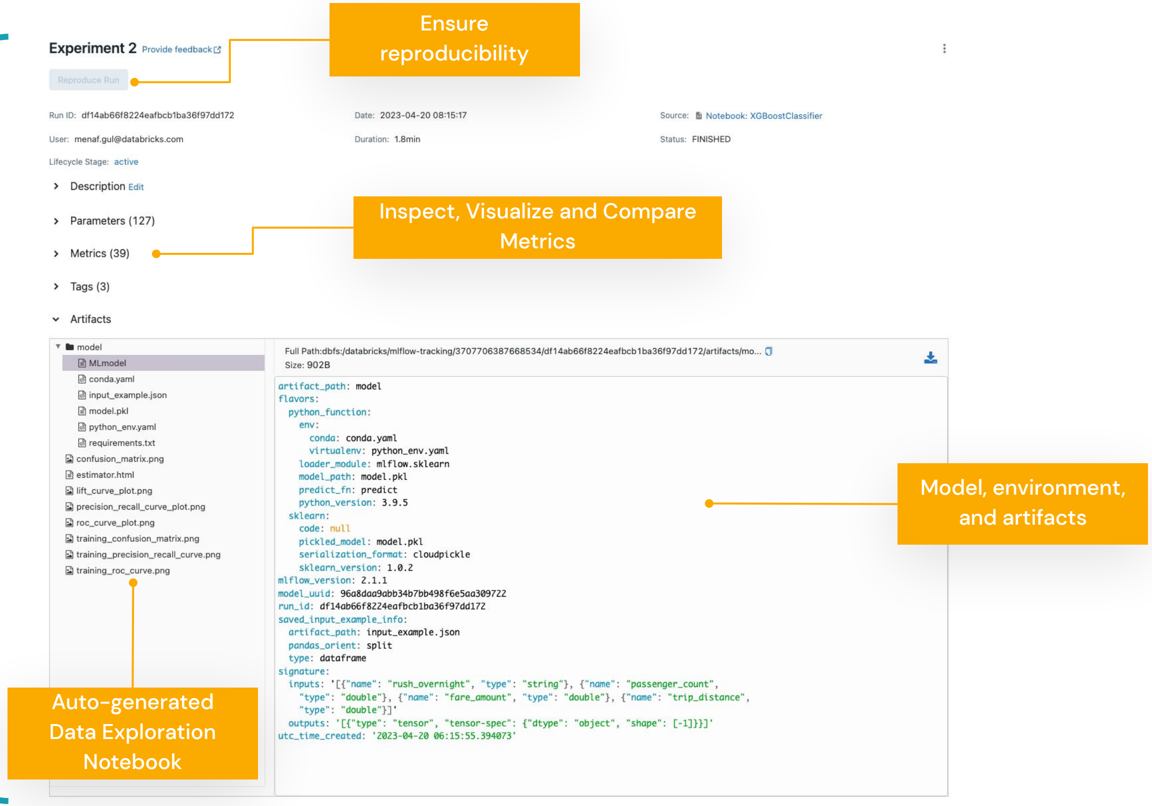
Centralized and collaborative model lifecycle management

APIs: CLI, Python, R, Java, REST

MLflow Tracking and Auto-logging

`mlflow.autolog()`

Track ML development with one line of code: parameters, metrics, data lineage, model, and environment.



DEMO:

MLFLOW

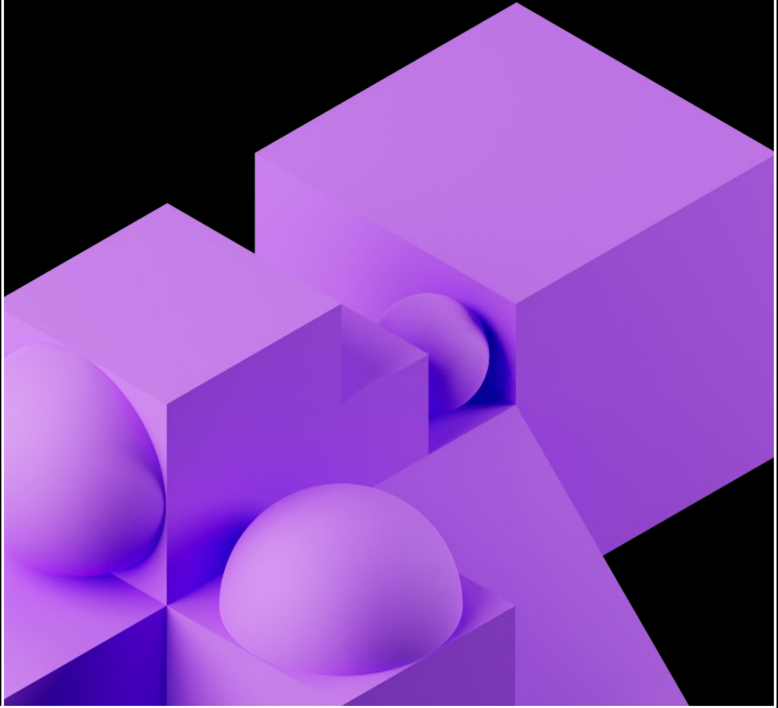
Notebook: 02-MLflow → 02.1-MLflow

LAB:

MLFLOW

Notebook: 02-MLflow → 02.1L-MLflow Lab

Hyperparameter Tuning

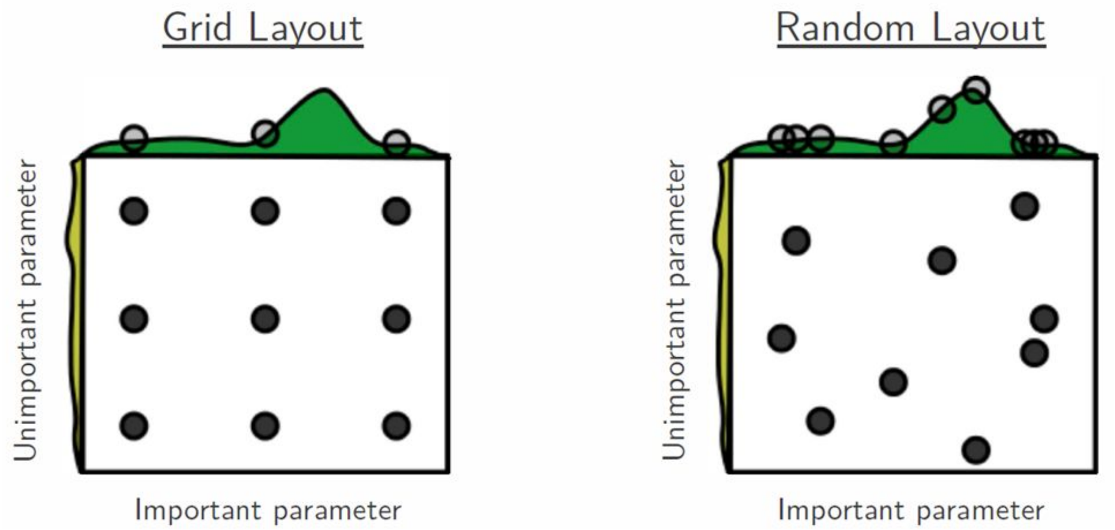


Problems with Grid Search

- Exhaustive enumeration is expensive
- Manually determined search space
- Past information on good hyperparameters isn’t used
- So what do you do if…
  - You have a training budget
  - You have many hyperparameters to tune
  - You want to pick your hyperparameters based on past results

Optimizing Hyperparameter Values

Random Search



Random Search generally outperforms grid search

Image source

Hyperopt

HYPEROPT

- Open-source Python library
- Optimization over *awkward search spaces* (real-valued, discrete, and conditional dimensions)
- Supports serial or parallel optimization
- Spark integration
- Core algorithms for optimization:
    - Random Search
    - Adaptive Tree of Parzen Estimators (TPE)

Paper

DEMO:

HYPEROPT

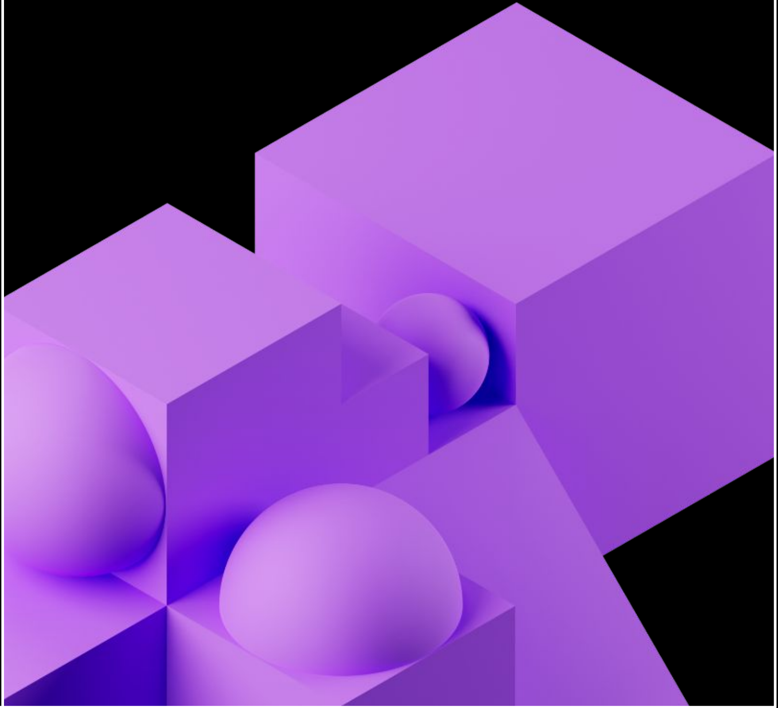
Notebook: 03-Hyperopt → 03.1-Hyperopt

LAB:

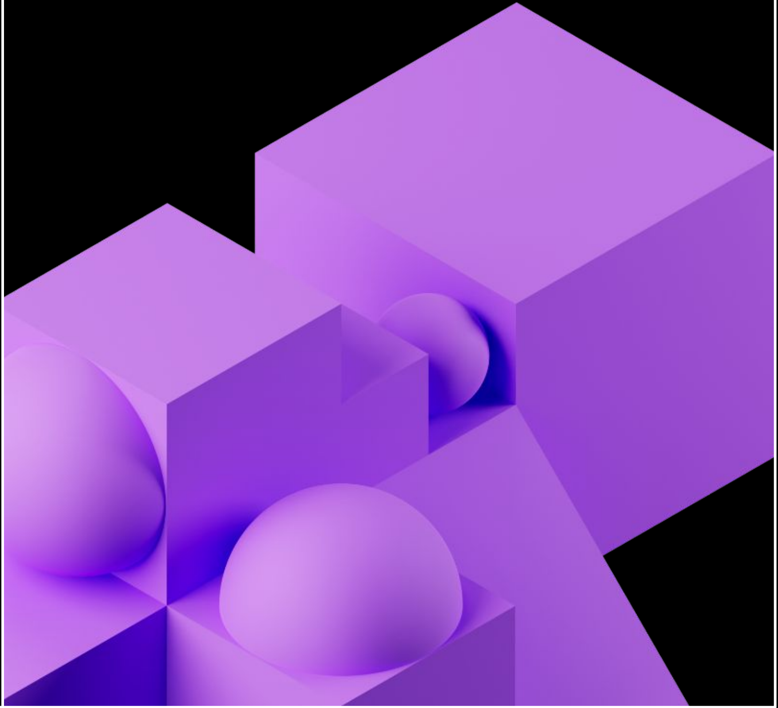
HYPEROPT

Notebook: 03-Hyperopt → 03.1L-Hyperopt Lab

Petastorm



Horovod



Horovod

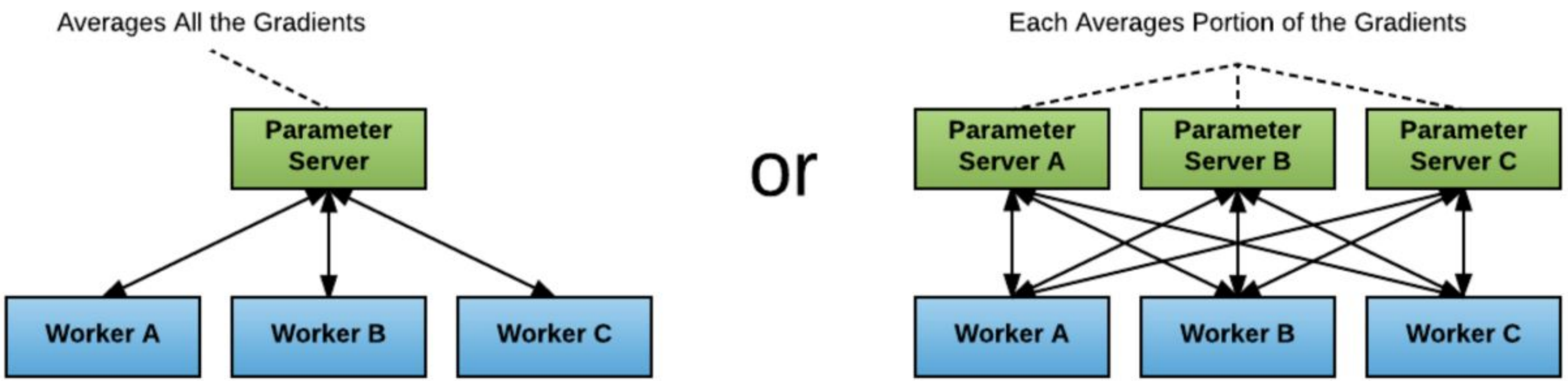
Like this Russian dance, everybody dances in a circle and each person only touches their neighbor.

Image source

Horovod

- Open-sourced by Uber in 2017
- Simplifies distributed neural network training
- Supports TensorFlow, Keras, PyTorch, and Apache MXNet

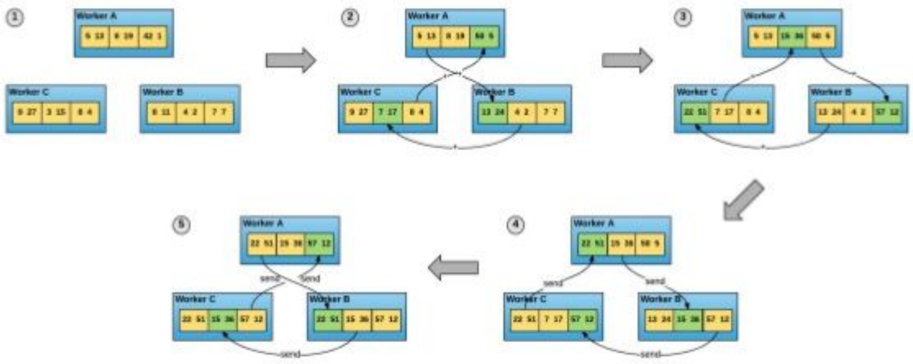
Classical Parameter Server



[Image source](#)

Ring All-Reduce

Horovod Technique



Patarasuk, P., & Yuan, X. (2009). Bandwidth optimal all-reduce algorithms for clusters of workstations. *Journal of Parallel and Distributed Computing*, 69(2), 117-124. doi:10.1016/j.jpdc.2008.09.002

UBER

```
# Only one line of code change!
optimizer = hvd.DistributedOptimizer(optimizer)
```

Image source

DEMO:

PETASTROM

Notebook: 04-Petastrom and Horovod → 04.1-Petastrom for Large Data

DEMO:

HOROVOD

Notebook: 04-Petastrom and Horovod → 04.2-Horovod for Distributed Training

LAB:

HOROVOD

Notebook: 04-Petastrom and Horovod → 04.2L-Horovod with Petastrom Lab

Convolutional Neural Networks

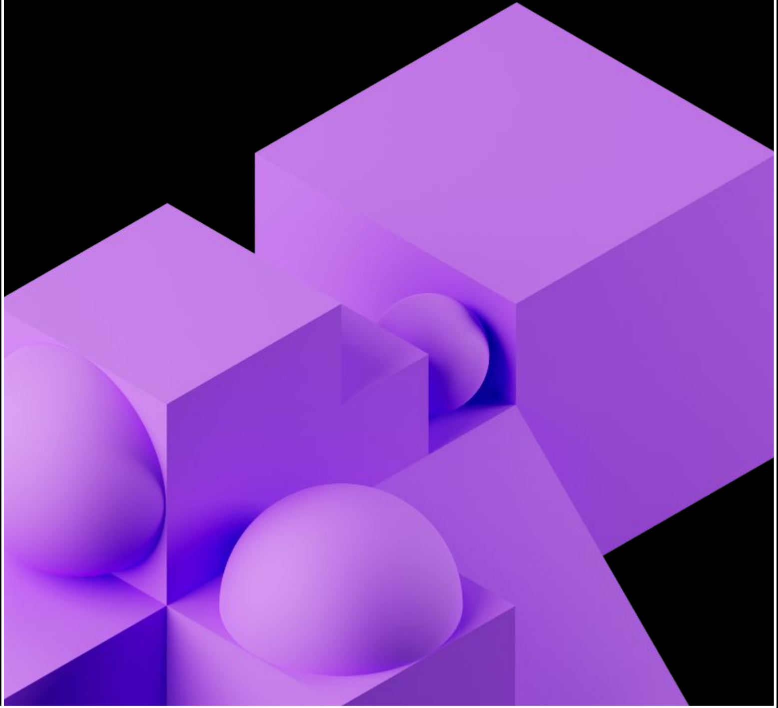


Image source

ImageNet Challenge

- Classify images in one of 1000 categories
- 2012 Deep Learning breakthrough with AlexNet: 16% top-5 test error rate (next closest was 25%)

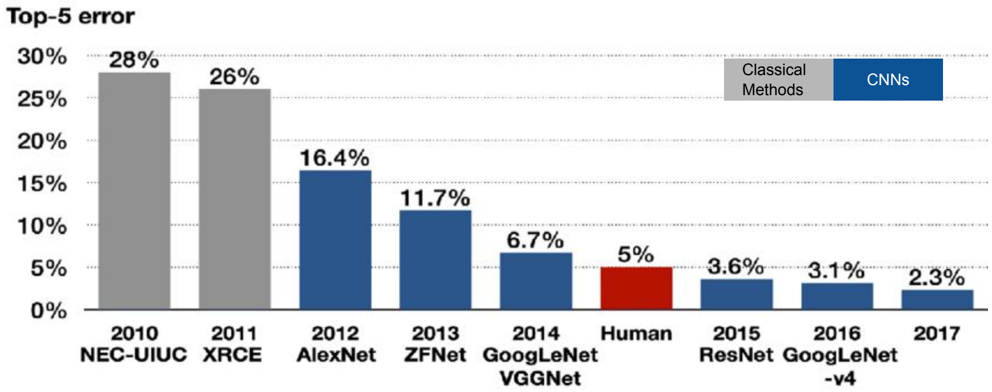
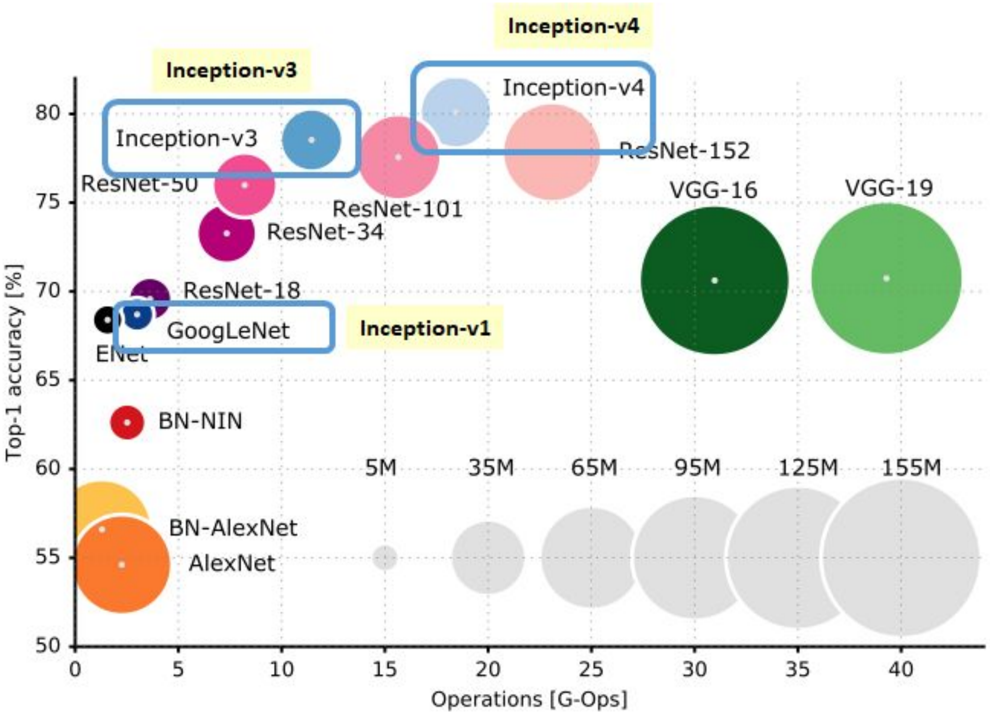


Image source

Top-1 Accuracy against Number of Operations



VGG16 (2014)

- One of the most widely used architectures for its simplicity

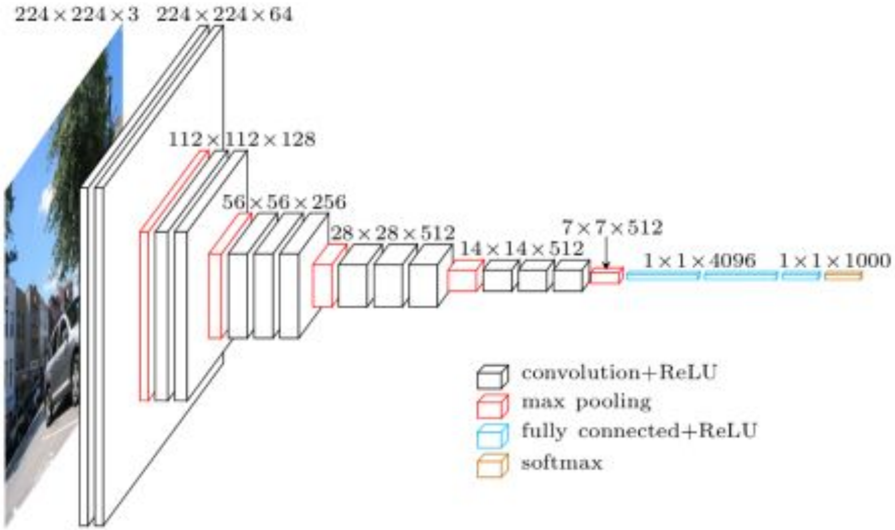


Image source

Convolutions

- Focus on spatial correlation (fewer parameters to learn)
- Filter/kernel slides across input image (often 3x3 and 5x5)

Image Kernels Visualization

CS 231 Convolutional Networks

Convolution

- Kernel size: Field of view
- Stride: Step size of kernel
- Padding: How to handle border
  - Valid: No padding (crops)
  - Same: Pads it so input_size = output_size

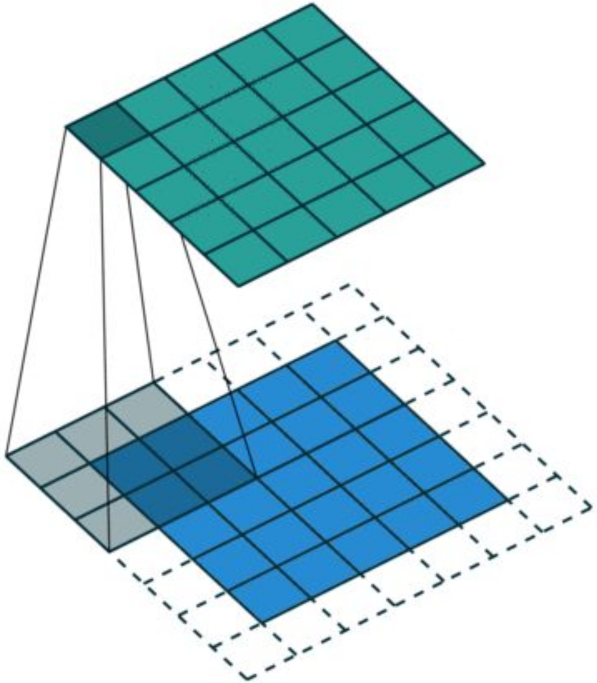


Image source

Max Vs Avg. Pooling

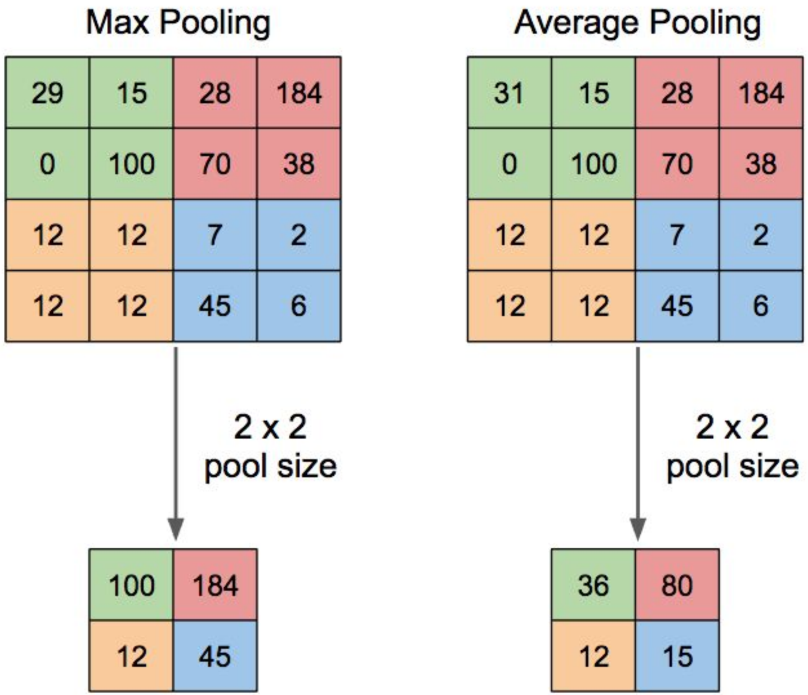


Image source

VGG-16 Summary

Compute block1_conv1:

3*3*3*64 + 64 = 1,792 params

How to compute block1_conv2?

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
block1_conv1 (Convolution2D)	(None, 224, 224, 64)	1792	input_1[0][0]
block1_conv2 (Convolution2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0]
block2_conv1 (Convolution2D)	(None, 112, 112, 128)	73856	block1_pool[0][0]
block2_conv2 (Convolution2D)	(None, 112, 112, 128)	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0]
block3_conv1 (Convolution2D)	(None, 56, 56, 256)	295168	block2_pool[0][0]
block3_conv2 (Convolution2D)	(None, 56, 56, 256)	590080	block3_conv1[0][0]
block3_conv3 (Convolution2D)	(None, 56, 56, 256)	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][0]
block4_conv1 (Convolution2D)	(None, 28, 28, 512)	1180160	block3_pool[0][0]
block4_conv2 (Convolution2D)	(None, 28, 28, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Convolution2D)	(None, 28, 28, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][0]
block5_conv1 (Convolution2D)	(None, 14, 14, 512)	2359808	block4_pool[0][0]
block5_conv2 (Convolution2D)	(None, 14, 14, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Convolution2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][0]
flatten (Flatten)	(None, 25088)	0	block5_pool[0][0]
fc1 (Dense)	(None, 4096)	102764544	flatten[0][0]
fc2 (Dense)	(None, 4096)	16781312	fc1[0][0]
predictions (Dense)	(None, 1000)	4097000	fc2[0][0]

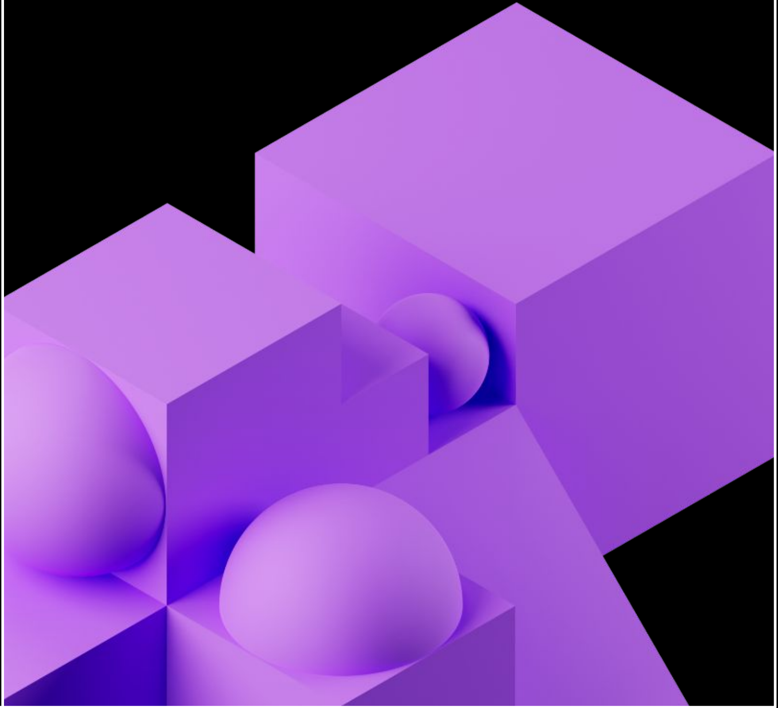
Total params: 138357544

DEMO:

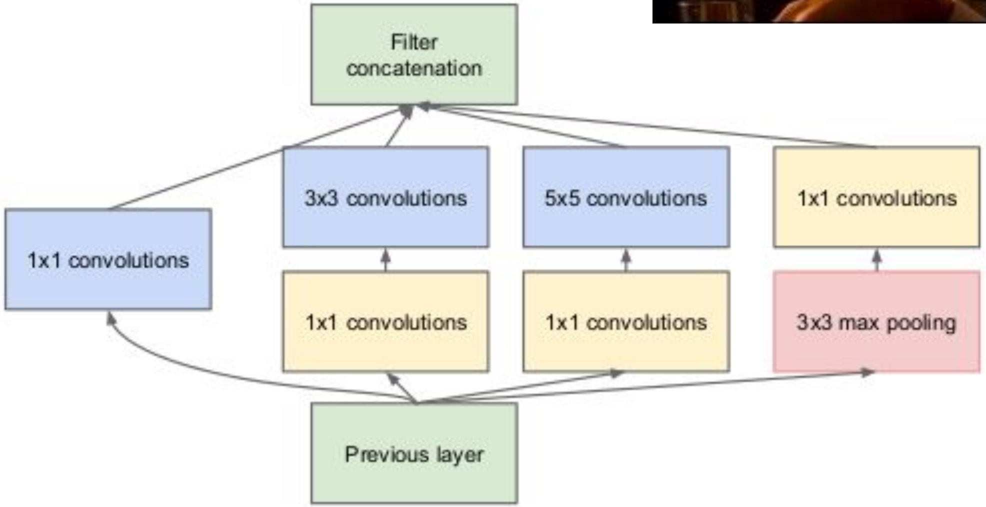
CNN

Notebook: 06-Convolutional Neural Networks → 06.1-Distributed Inference with CNNs

Evolution of CNN Architectures



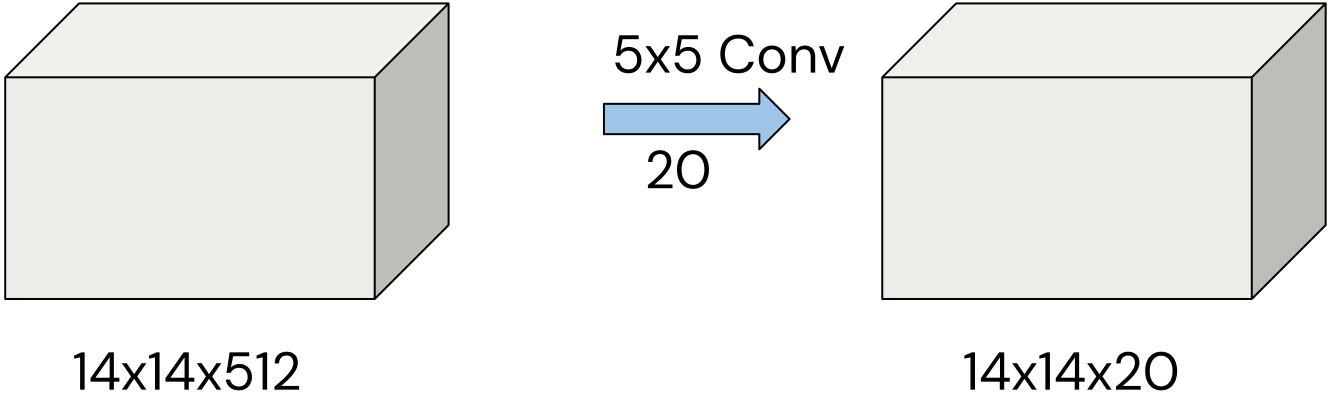
GoogLeNet/Inception - 2014



GoogLeNet/Inception

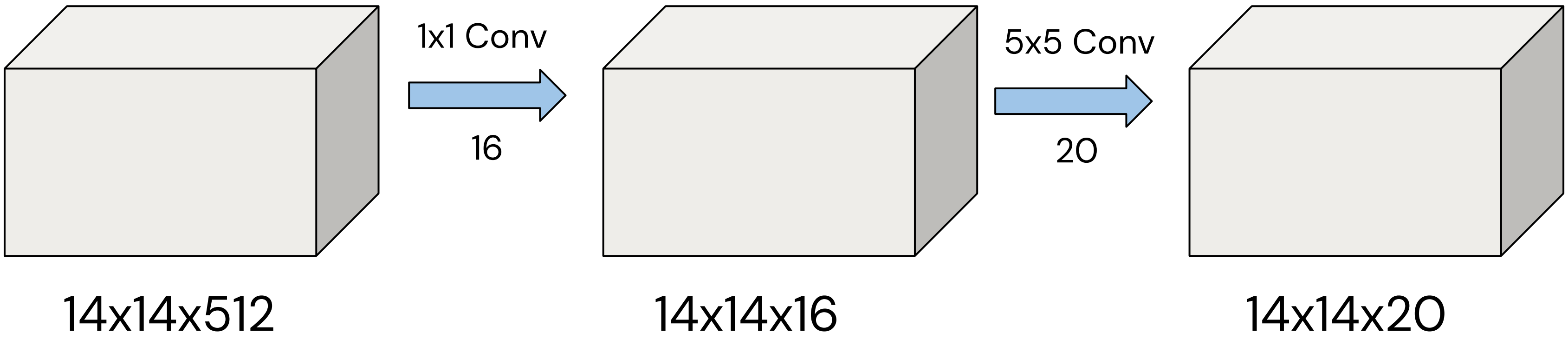
- Uses 1X1 convolutions to reduce dimension of input channels, then applies larger kernel
- Inception module:
  - Different sizes/types of convolutions for same input (effectively learns different features)
  - Stacks all the outputs
- Replaces the fully-connected layers at the end with global average pooling

Without 1x1 Convolution



Number of operations: (14x14x512) x (5x5x20) = 50,176,000

With 1x1 Convolution



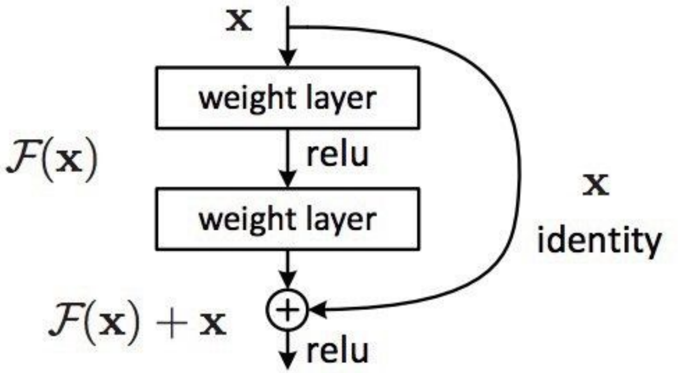
Number of operations for 1x1: (14x14x512)x(1x1x16) = 1,605,632

Number of operations for 5x5: (14x14x16)x(5x5x20) = 1,568,000

Total number of operations: 3,173,632 – just 6.3% of original parameters!

ResNet (Residual Network) – 2015

“Direct mappings are hard to learn…instead of trying to learn an underlying mapping from x to H(x), learn the difference between the two, or the “residual.”



There are many other newer models

Network	Year	Top-5 ImageNet Accuracy	# of Params	Floating Point Operations
AlexNet	2012	84.7%	62M	1.5B
VGGNet	2014	92.3%	138M	19.6B
Inception v1	2014	93.3%	6.4M	2B
ResNet-152	2015	95.5%	60.3M	11B
Inception v3	2015	94.4%	23.8M	-
XCeption	2016	94.5%	22.8M	-
NasNet	2017	95.3%	22.6M	-
MobileNet	2017	89.5%	4.24M	-
EfficientNet B5	2019	96.7%	30M	9.9B

Larger, Newer Models != Better

- MIT and Amazon researchers found ~3.4% errors across 10 benchmarking datasets. Label errors documented here: https://labelerrors.com

MNIST given label:
8

We guessed: **9**

MTurk consensus: **9**

ID: 947

ImageNet given label:
tick

We guessed: **yellow garden spider**

MTurk consensus: **yellow garden spider**

ID: 00004095

CIFAR-10 given label:
airplane

We guessed: **ship**

MTurk consensus: **ship**

ID: 1718

Smaller models outperform on corrected labels

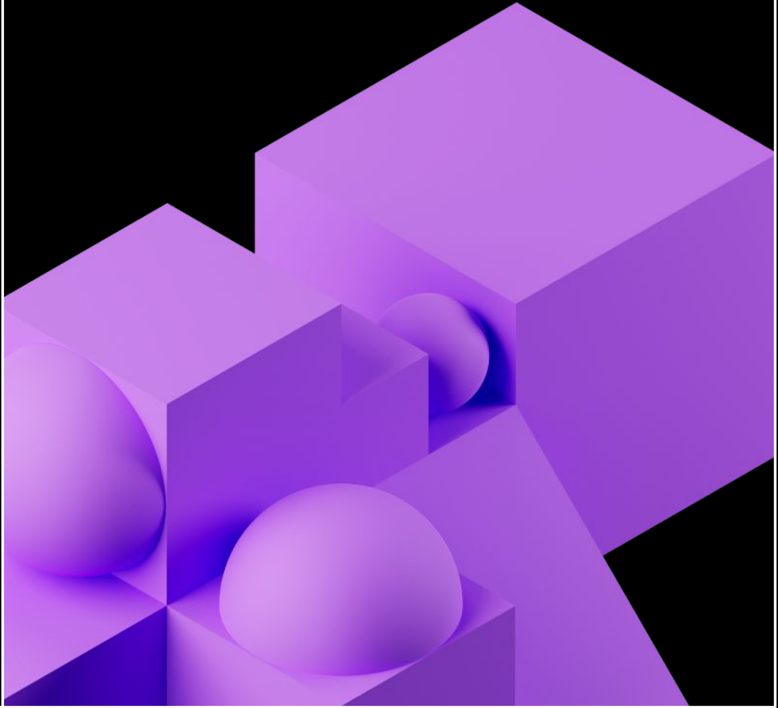
- Rankings of small models pre-trained on ImageNet increased
  - NasNet ranking drops from 1/34 -> 29/34
  - XCeption drops from 2/34 to 24/34
  - ResNet-18 increases from 34/34 -> 1/34
  - ResNet-50 increases from 20/34 -> 2/34
- Same trend occurs on the 13 models pre-trained on CIFAR-10
  - VGG-11 > VGG-19

Network	Year	Top-5 ImageNet Accuracy	# of Params	Floating Point Operations
AlexNet	2012	84.7%	62M	1.5B
VGGNet	2014	92.3%	138M	19.6B
Inception v1	2014	93.3%	6.4M	2B
ResNet-152	2015	95.5%	60.3M	11B
Inception v3	2015	94.4%	23.8M	-
XCeption	2016	94.5%	22.8M	-
NasNet	2017	95.3%	22.6M	-
MobileNet	2017	89.5%	4.24M	-
EfficientNet B5	2019	96.7%	30M	9.9B

Detrimental impact of errors increases with model size

- Lower-capacity models
  - Provide regularization benefits
  - Are more resistant to learning asymmetric distribution of noisy labels
- Large models overfit to
  - specific benchmarks
  - quirks of the original label annotators
- Implications:
  - We only see the original test accuracy
  - But we should pick our models based on the corrected test accuracy
  - **SOTA on research data != SOTA on real production data**

Transfer Learning



Transfer Learning

- IDEA: Intermediate representations learned for one task may be useful for other related tasks

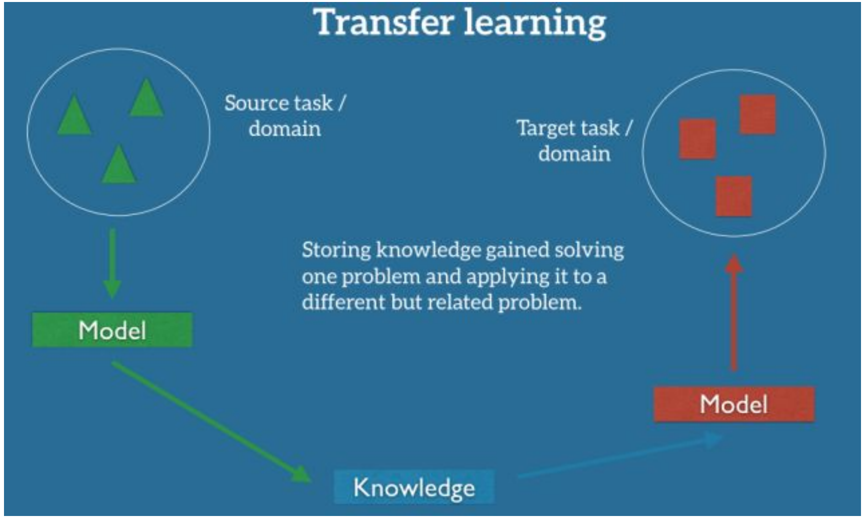


Image source

When To Use Transfer Learning?

	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

Reference: Andrej Karpathy's Transfer Learning

DEMO:

TRANSFER LEARNING

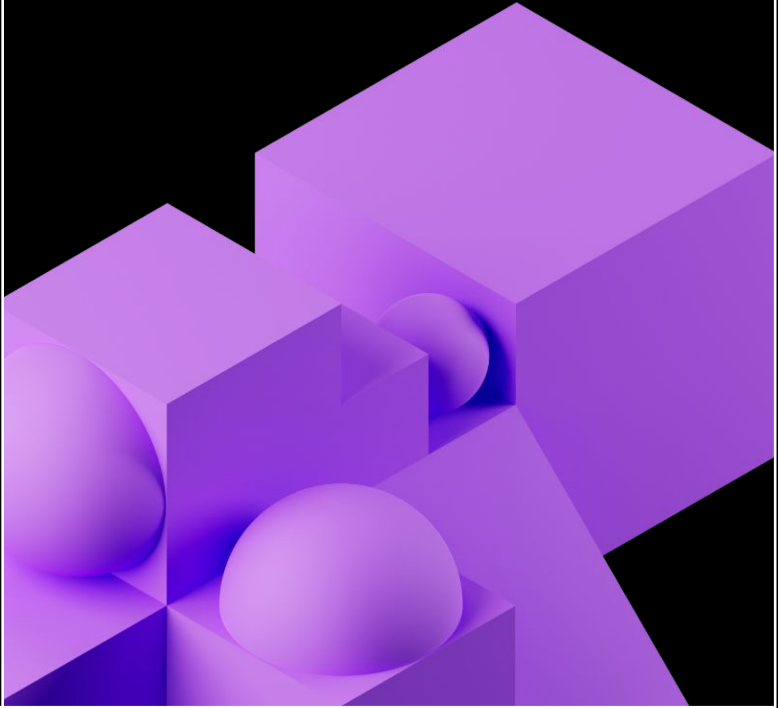
Notebook: 07-Transfer Learning → 07.1-Transfer Learning with Data Generators

LAB:

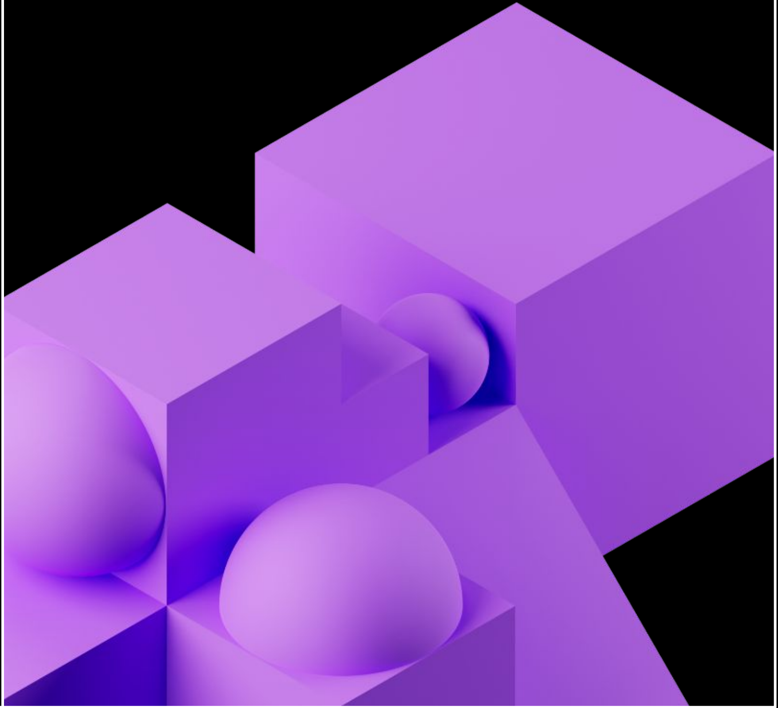
TRANSFER LEARNING

Notebook: 07-Transfer Learning → 07.2L - Transfer Learning for CNNs Lab

Questions?



Summary and Next Steps

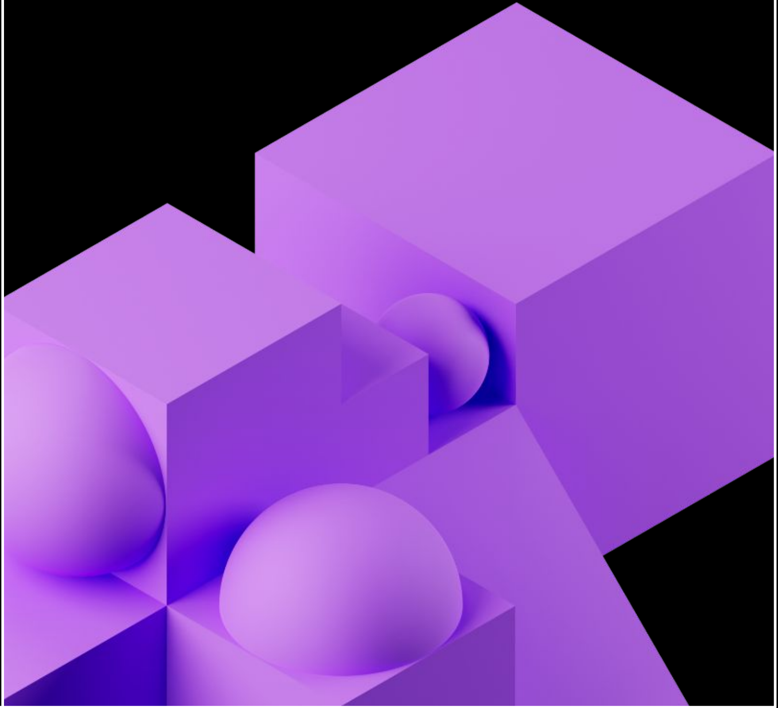


THANK YOU!

databricks

Appendix

Model Interpretability



SHAP (SHapley Additive exPlanations)

- Shapley Values

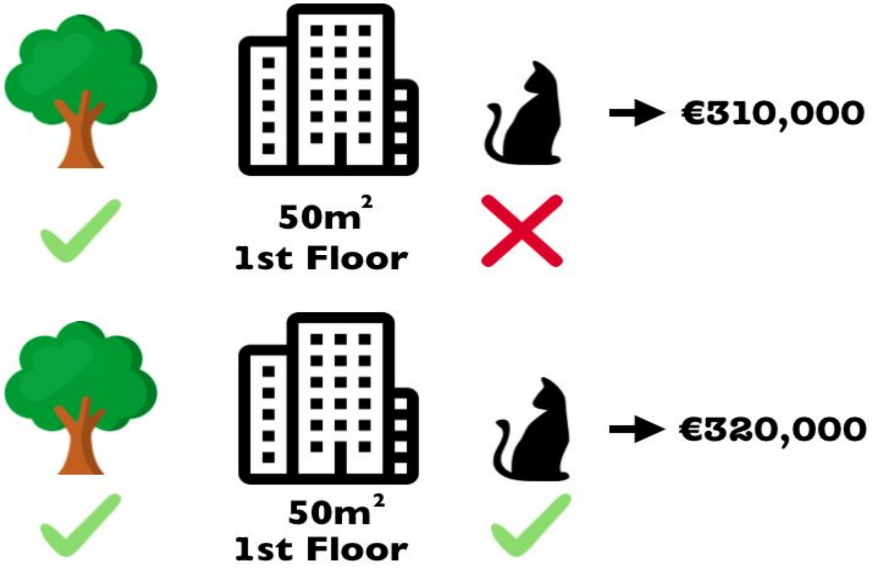


Image source

SHAP

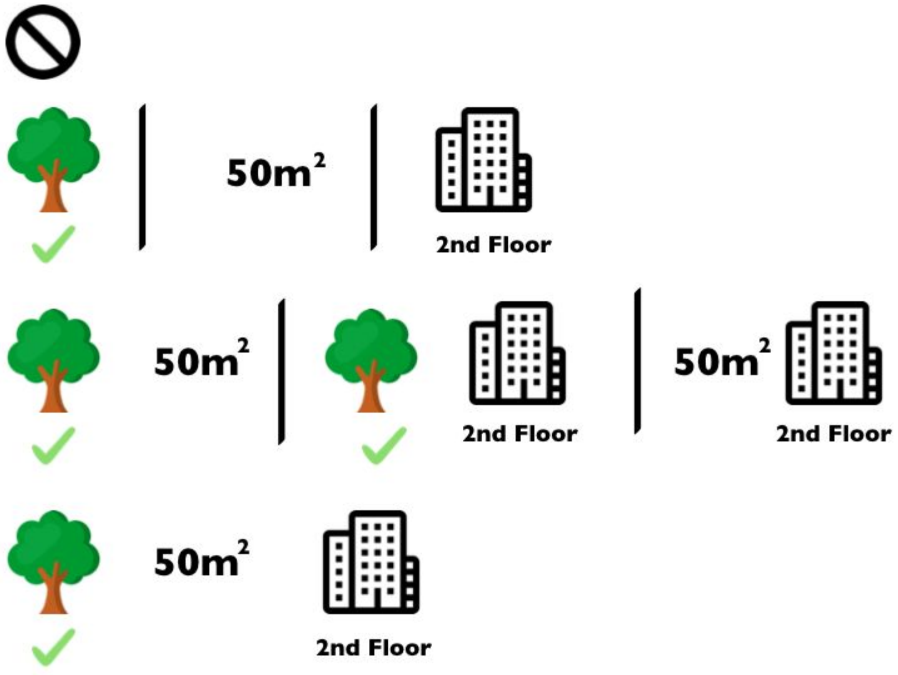
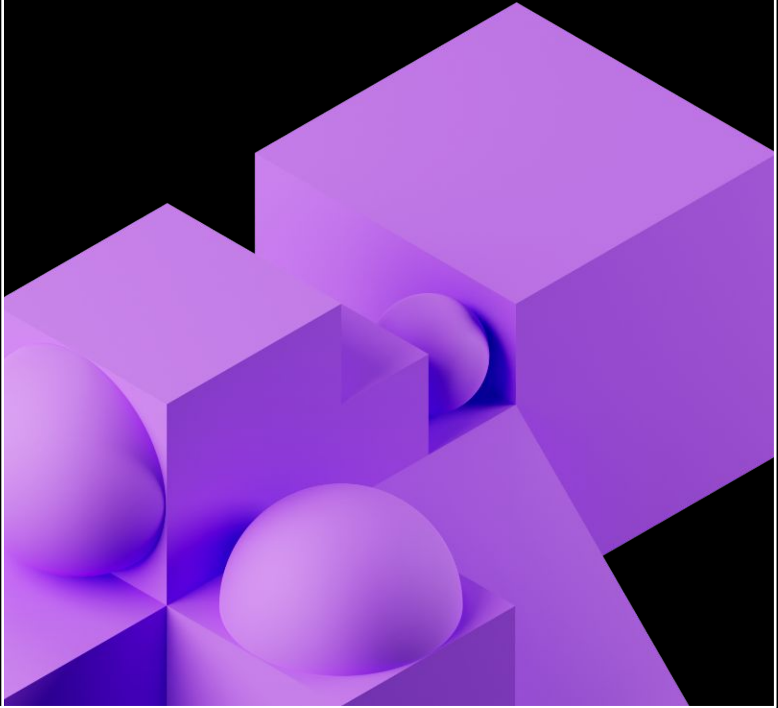


Image source

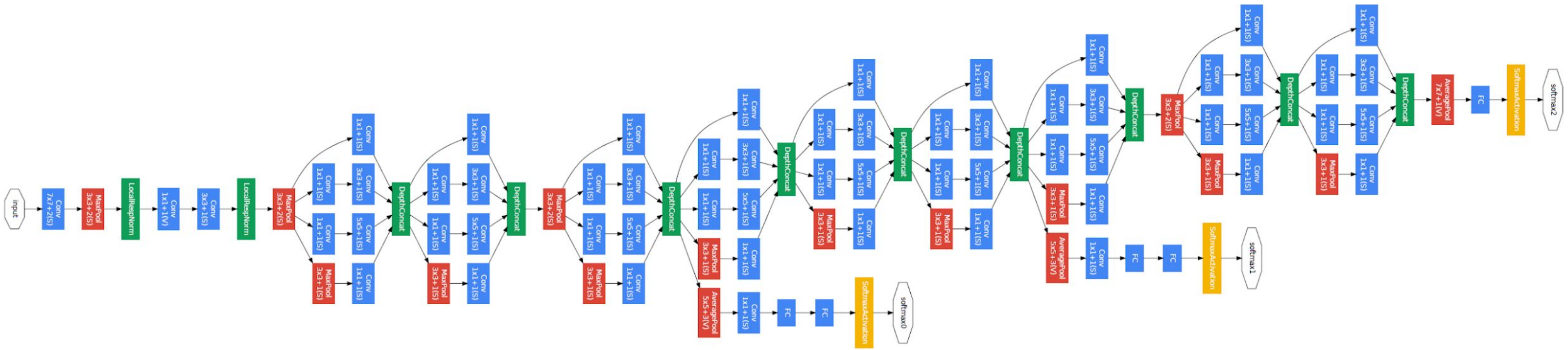
CV



What Do CNNs Learn?

- Breaking Convnets

Inception Model: 22 Layers in Total



Testing Details

- 7 models ensemble
  - 1 has a different architecture, how are the other 6 different?
- Multi-scale testing: 256, 288, 320, and 352 dimensions (4 scales)
- Multi-crop testing
- 4 scales×3 squares×6 crops×2 mirrored versions=144 crops/image

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

Inception v2/BN-Inception

- Adds Batch Norm
    - Normalizes layer inputs, allows for higher learning rate
- Replaces 5×5 conv with two 3×3 convs to reduce params

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

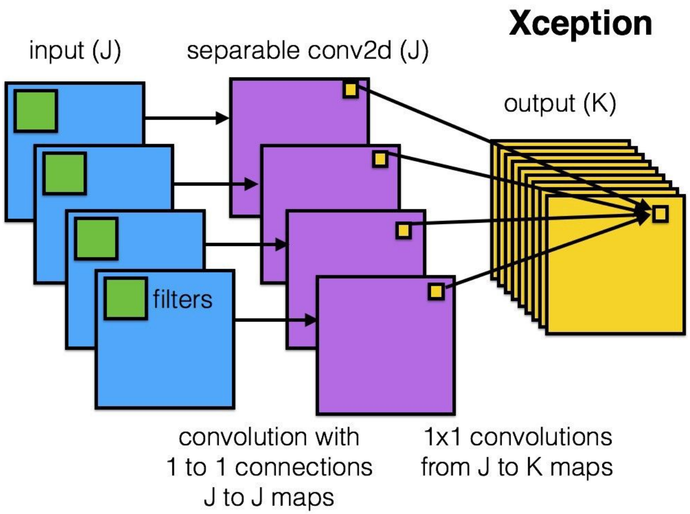
$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

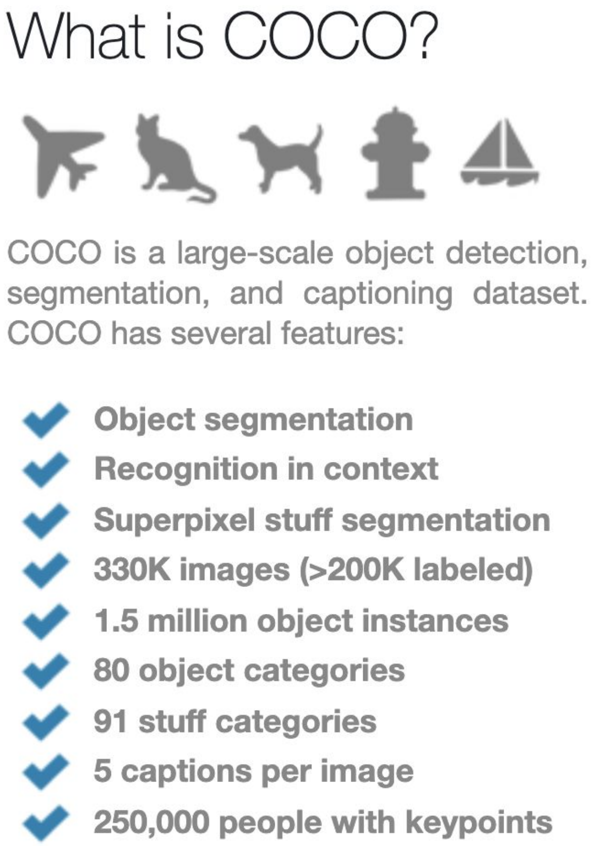
XCeption (Extreme Inception) – 2017

"Cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly."



Where is CV headed?

- We've overfit to ImageNet/MNIST/CIFAR…
  - COCO (MSFT/Facebook)
  - Open Images Dataset (Google)



databricks