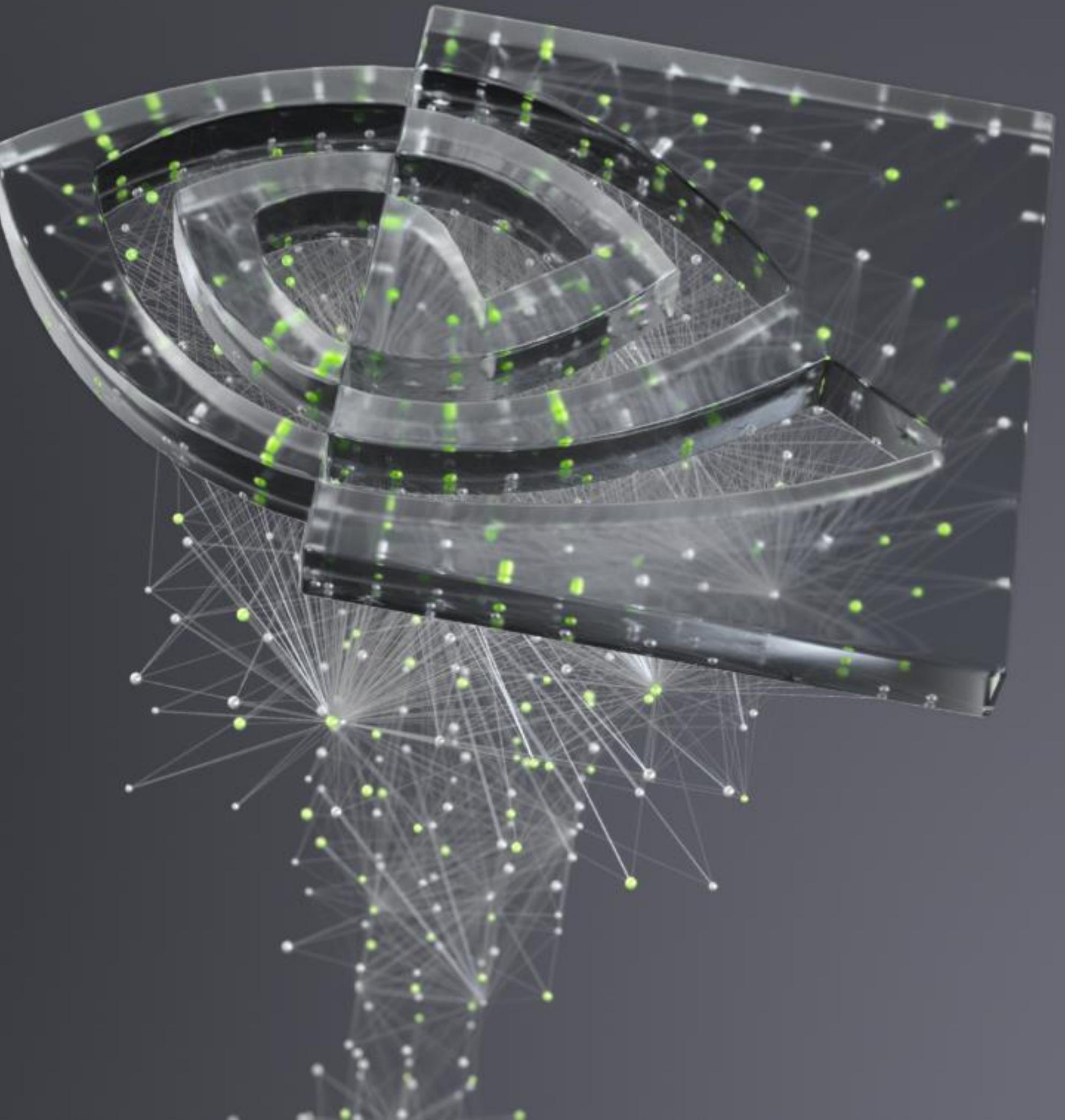




NVIDIA®

AI FOR SCIENCE A PRACTICAL INTRODUCTION TO DEEP LEARNING

WITH KERAS AND TENSORFLOW



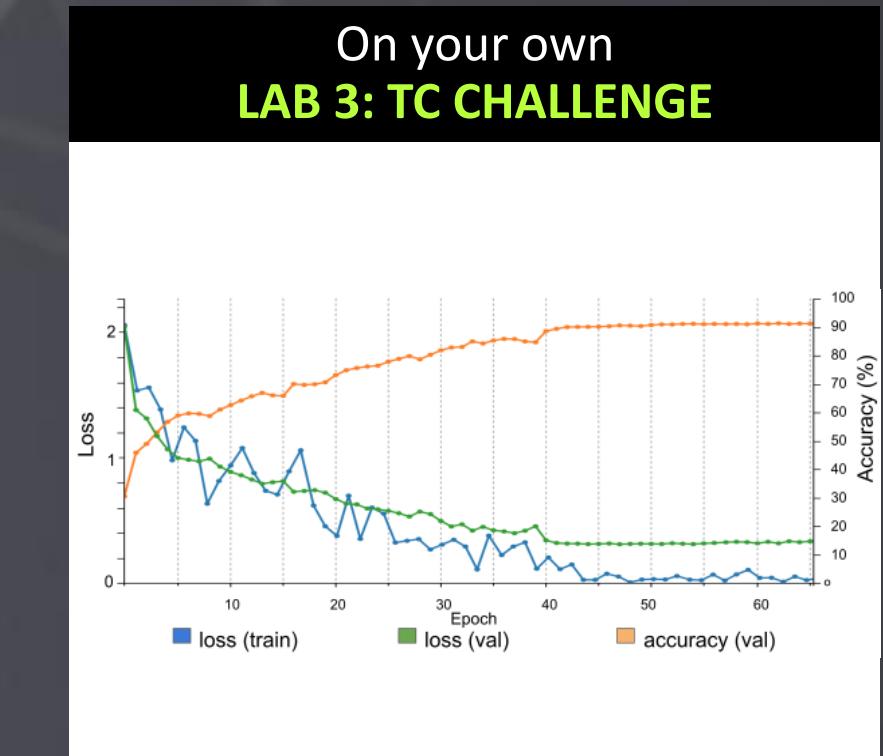
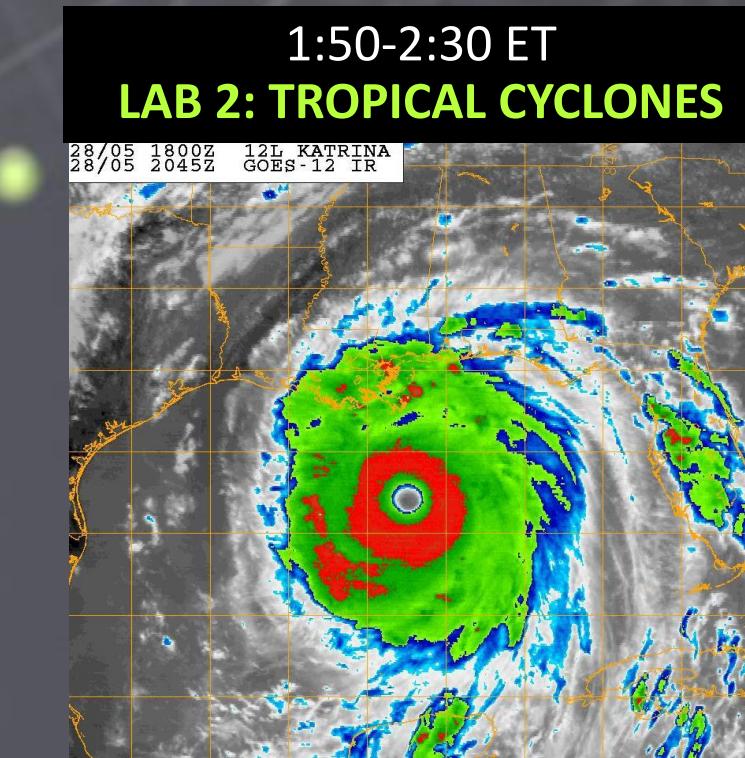
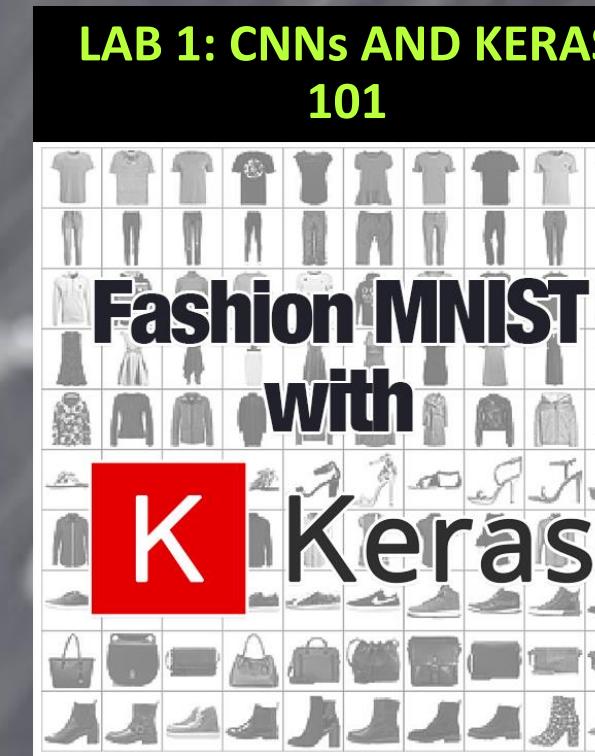
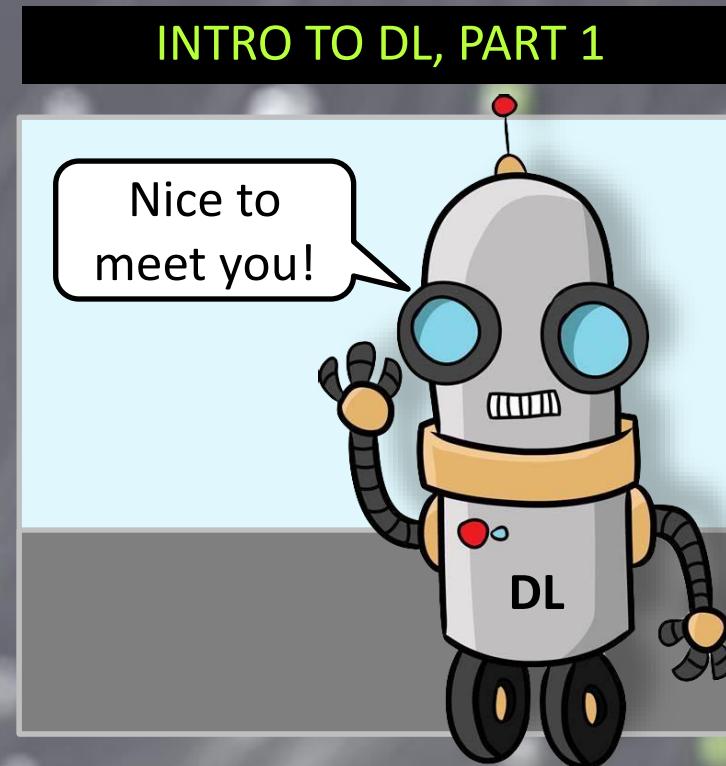
LEARNING GOALS



In couple of hours, we can only travel so far

Main goal:
Become familiar with main ideas and process

A starting point for solving your own problems

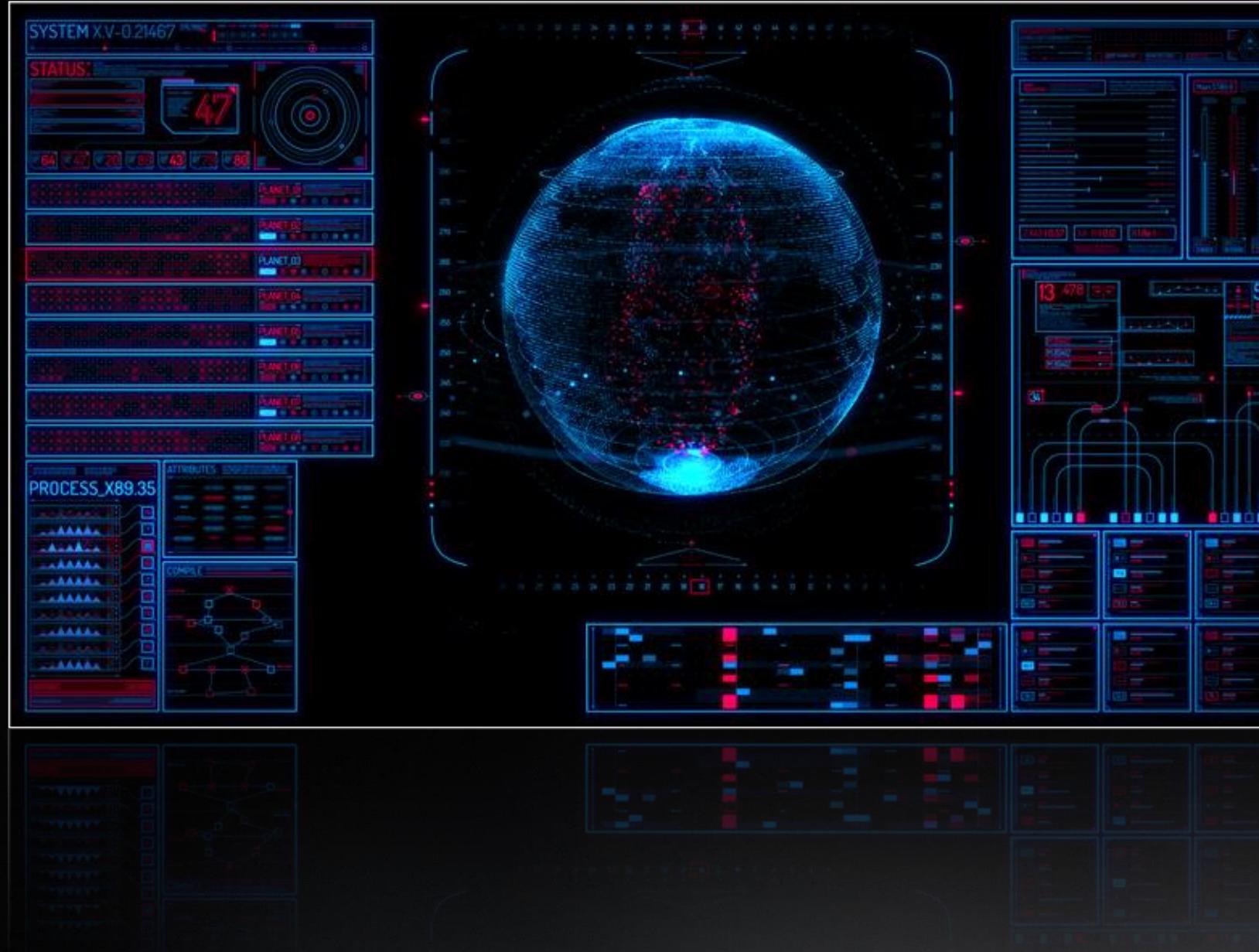


AGENDA

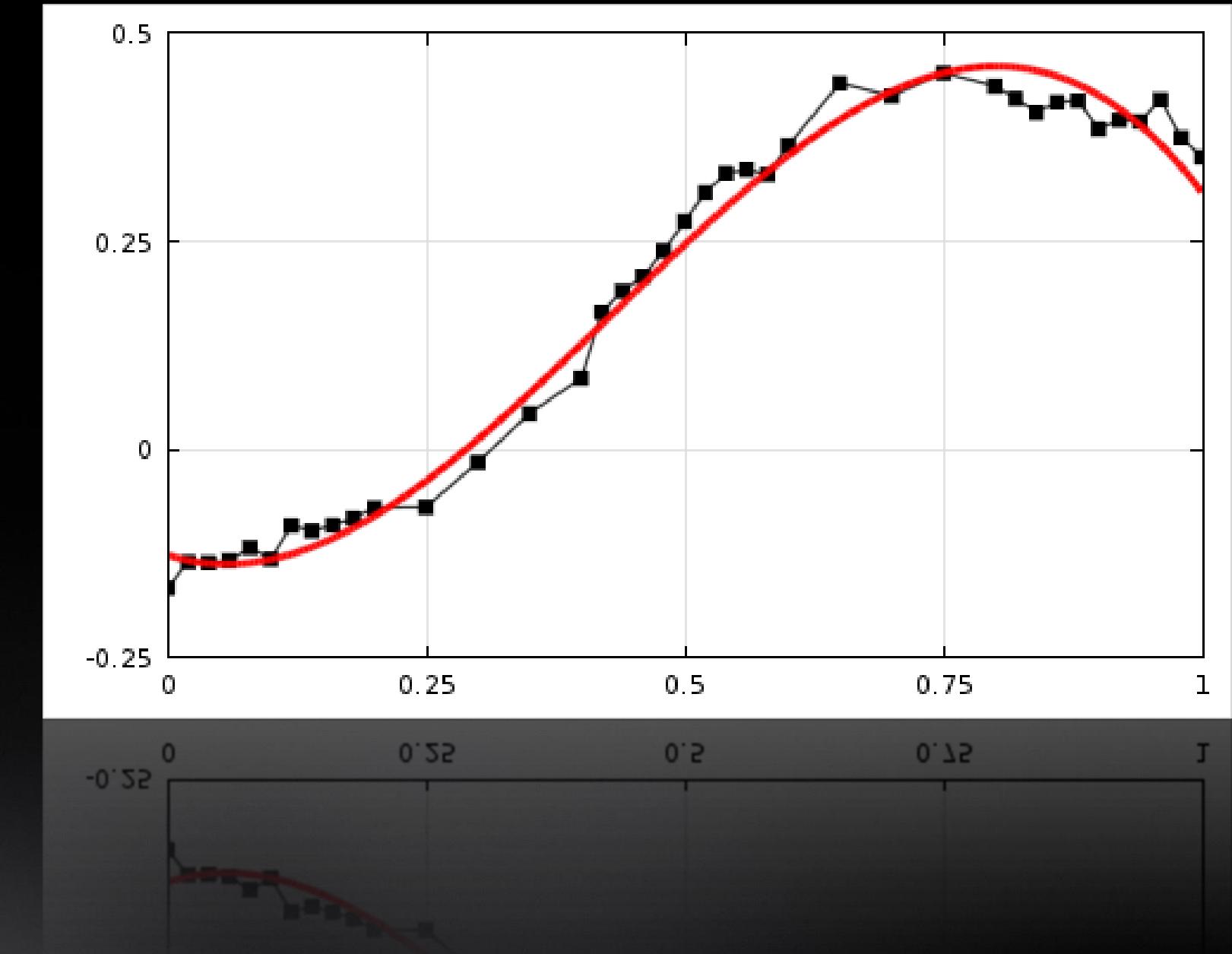
DEEP LEARNING ANALOGIES

What is this deep learning thing, anyway?

A NEW TYPE OF SOFTWARE

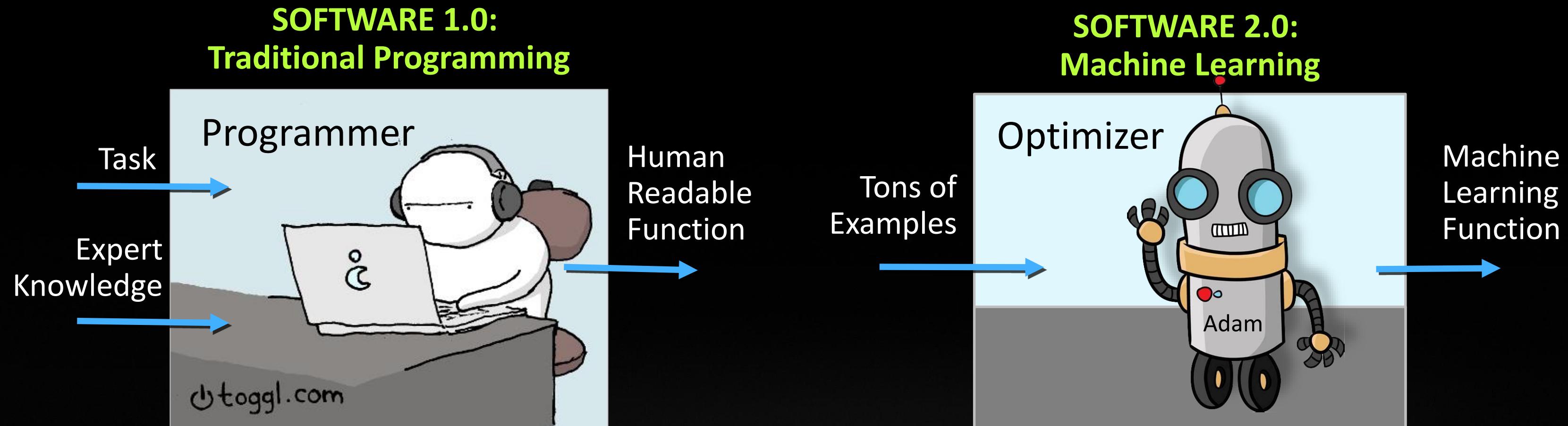


A GENERALIZATION OF CURVE FITTING



A NEW WAY TO BUILD SOFTWARE

Traditional Programming vs Machine Learning

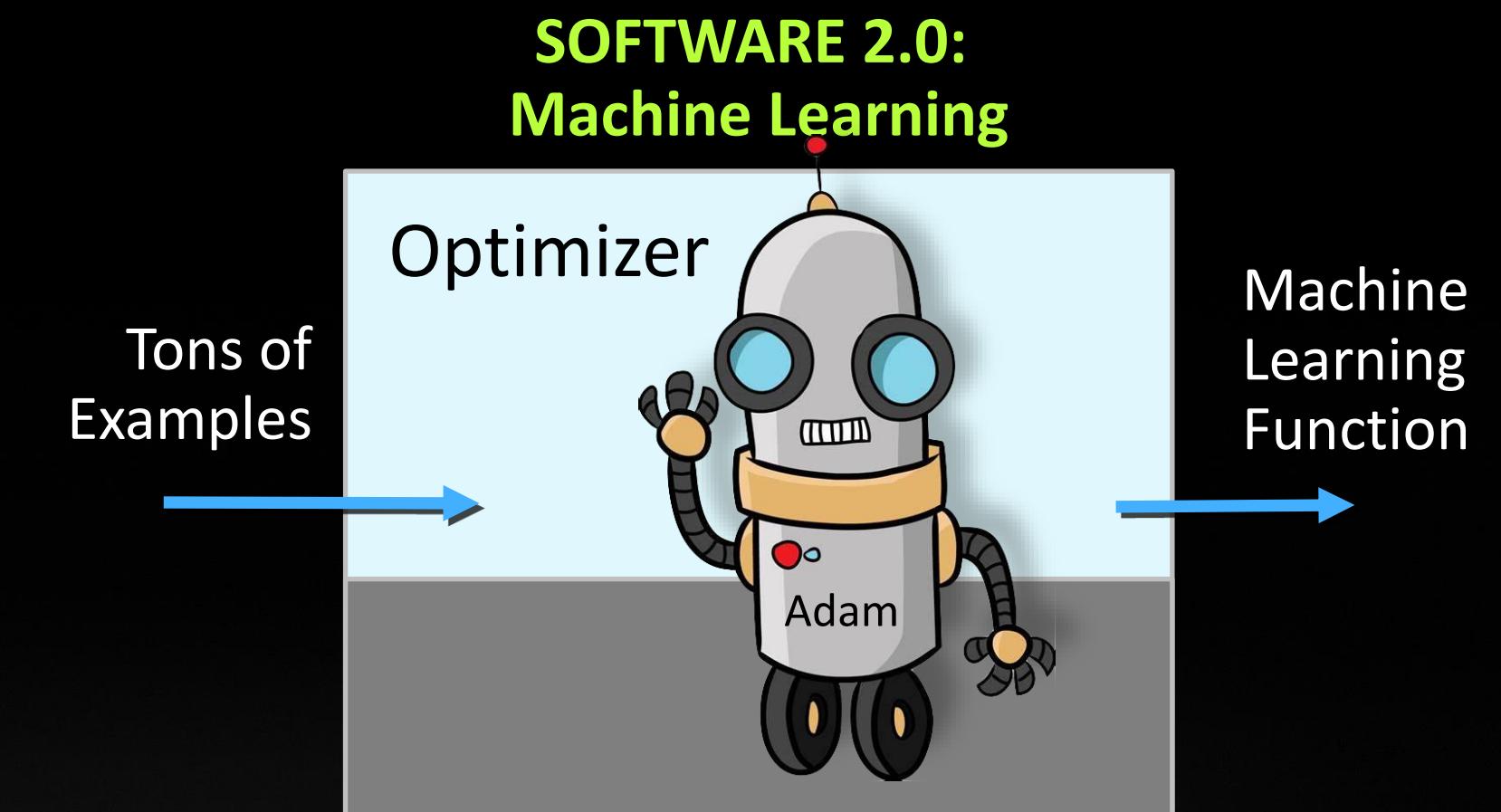


A DIFFERENT WAY TO BUILD SOFTWARE

Traditional Programming vs Machine Learning

Goals for today:

1. Learn to use this new approach
2. Revolutionize Science



A DIFFERENT WAY TO BUILD SOFTWARE

Hand written vs learned functions.

TEMP, PRESSURE, MOISTURE



PROBABILITY OF RAIN

HAND-WRITTEN FUNCTION

```
Function1(T,P,Q)  
  
update_mass()  
  
update_momentum()  
  
update_energy()  
  
do_mechanics()  
  
do_physics()  
  
y = get_precipitation()  
  
return y
```

Convert expert
knowledge into a function

LEARNED FUNCTION

```
Function1(T,P,Q)  
  
A = relu( w1 * [T,P,Q] + b1 )  
  
B = relu( w2 * A + b2 )  
  
C = relu( w3 * B + b3 )  
  
D = relu( w4 * C + b4 )  
  
E = relu( w5 * D + b5 )  
  
y = sigmoid(w6 * E + b6)  
  
return y
```

Reverse-engineer a function
from inputs / outputs

A DIFFERENT WAY TO BUILD SOFTWARE

The two approaches are complimentary



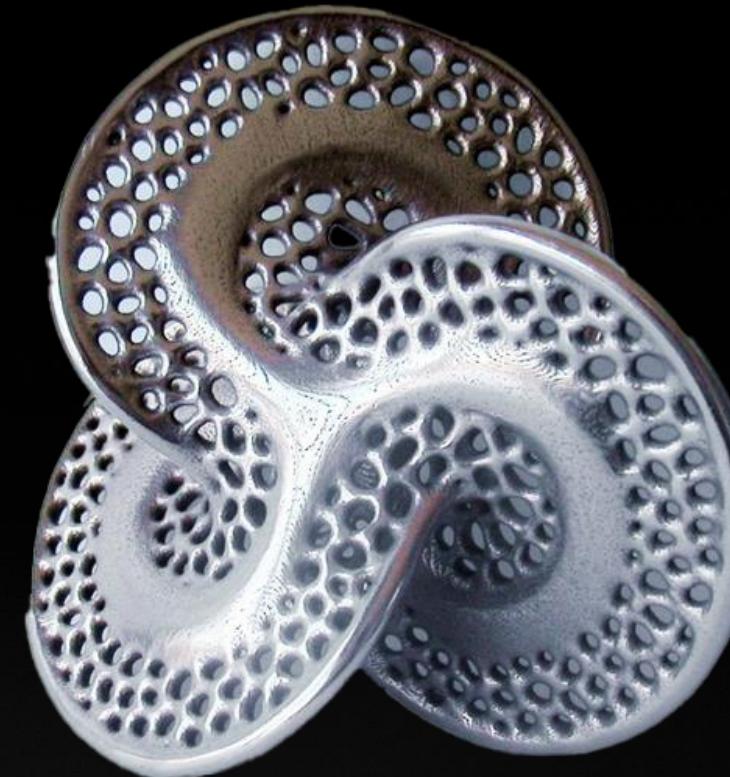
MANUAL PROGRAMMING

“SOFTWARE 1.0”
ENGINEERED
LABOR INTENSIVE
EXPLICIT
EXPLAINABLE
SIMPLE
FROM EXPERTISE

MACHINE LEARNING

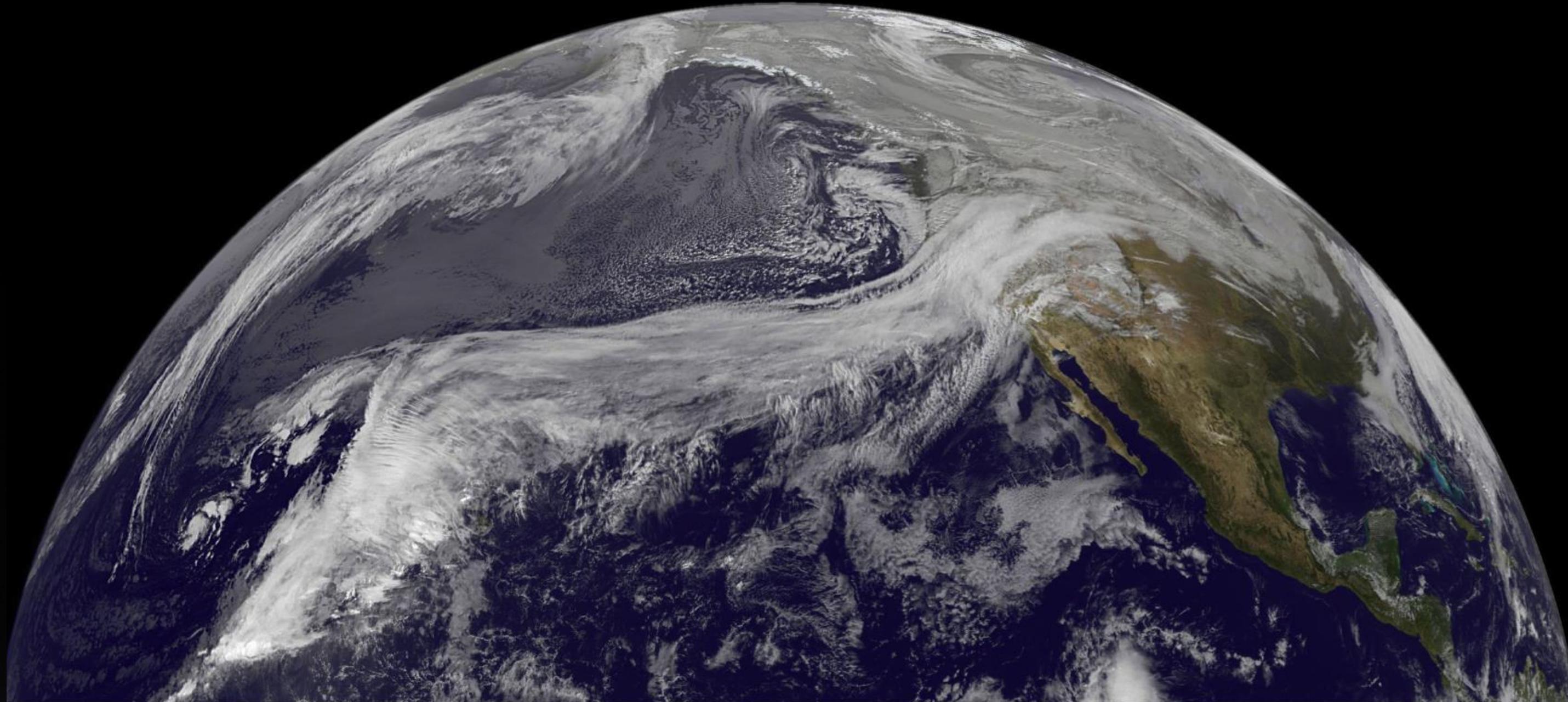
“SOFTWARE 2.0”
REVERSE-ENGINEERED
AUTOMATIC
IMPLICIT
SUBTLE
COMPLEX
FROM EXAMPLES

(For best results, combine as needed)



A DIFFERENT WAY TO BUILD SOFTWARE

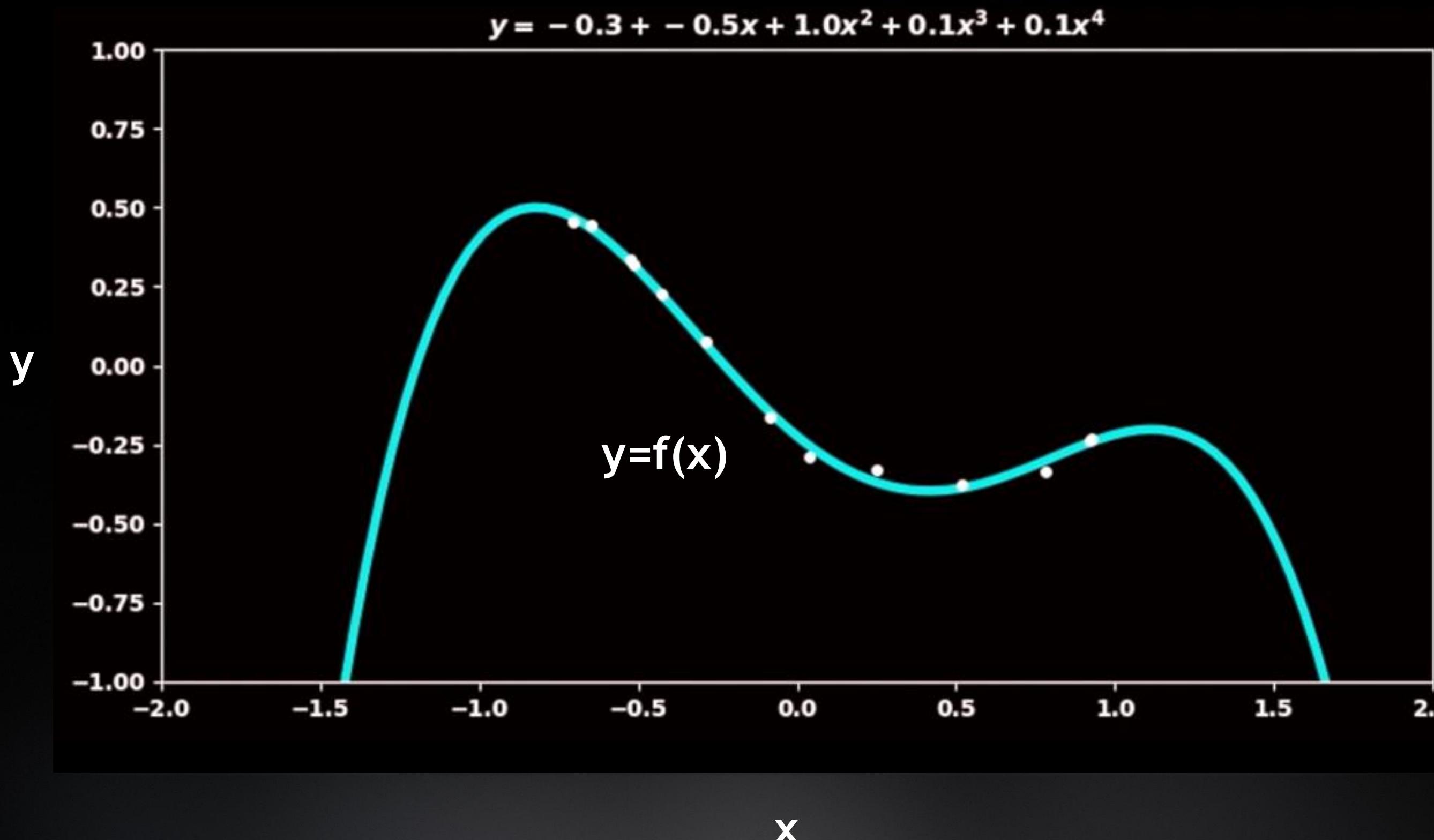
Complex phenomena are best described implicitly.



EXAMPLE: ATMOSPHERIC RIVER

A GENERALIZATION OF CURVE FITTING

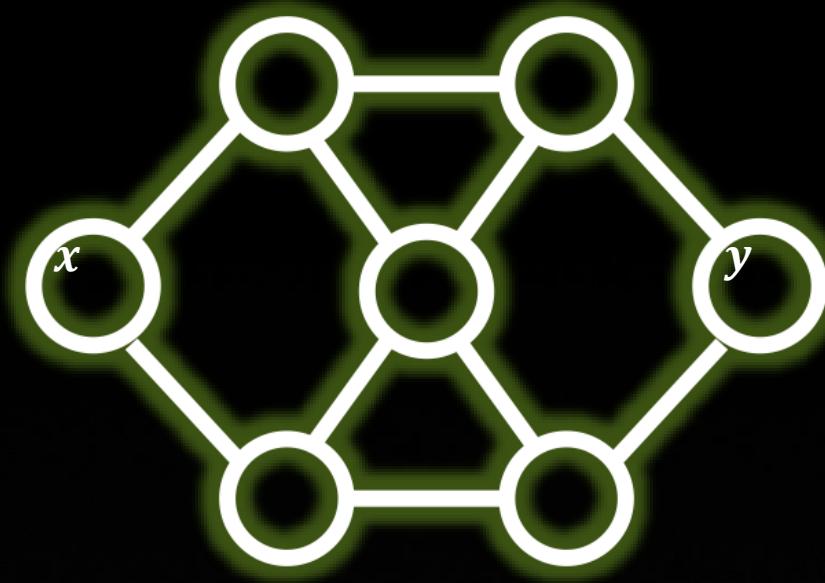
Curve fitting provides the starting intuition



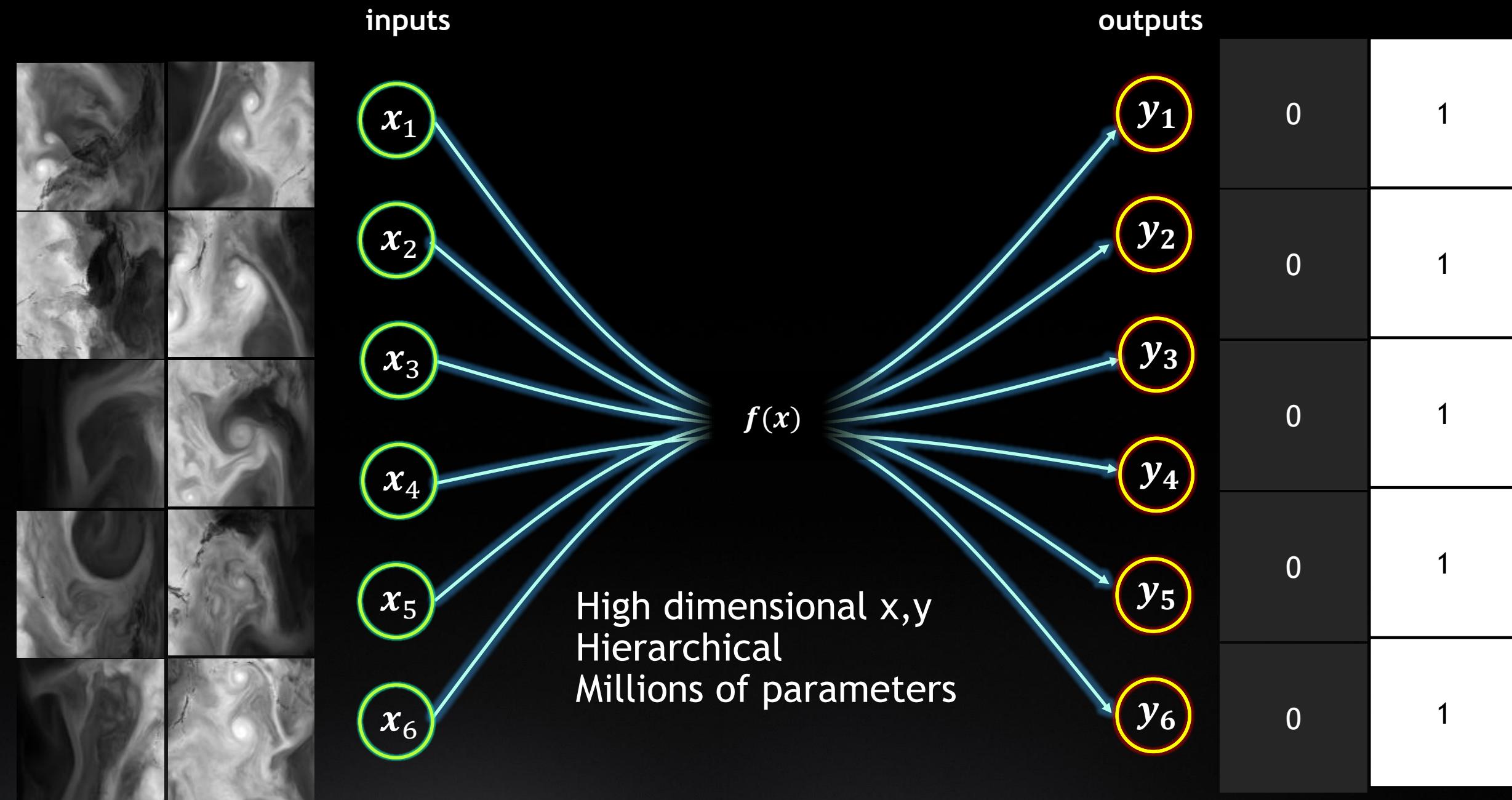
A GENERALIZATION OF CURVE FITTING

Differences from traditional curve fitting

Find f , given x and y



Supervised Deep Learning





IMPLEMENTATION BASICS

AUTO-ML

Eventually, the optimizer might be able to do everything for you

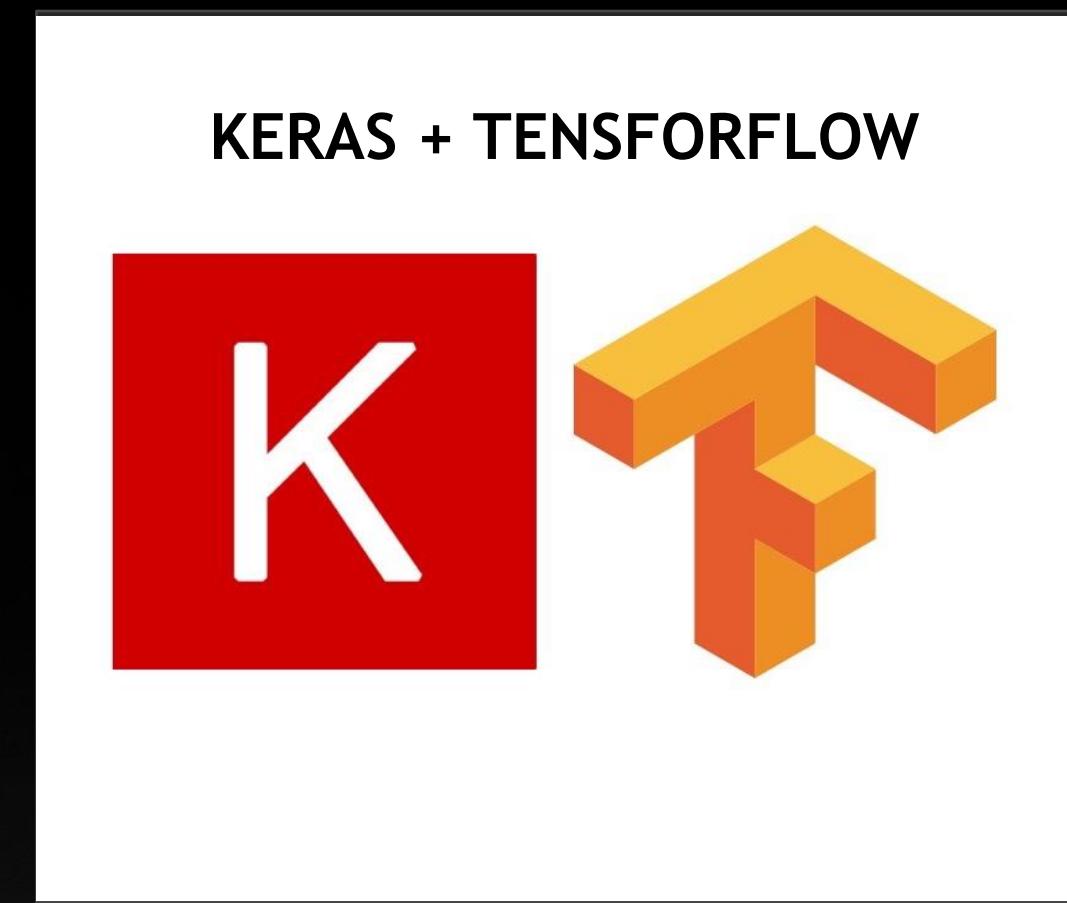


WHAT YOU NEED TO MAKE DEEP LEARNING WORK

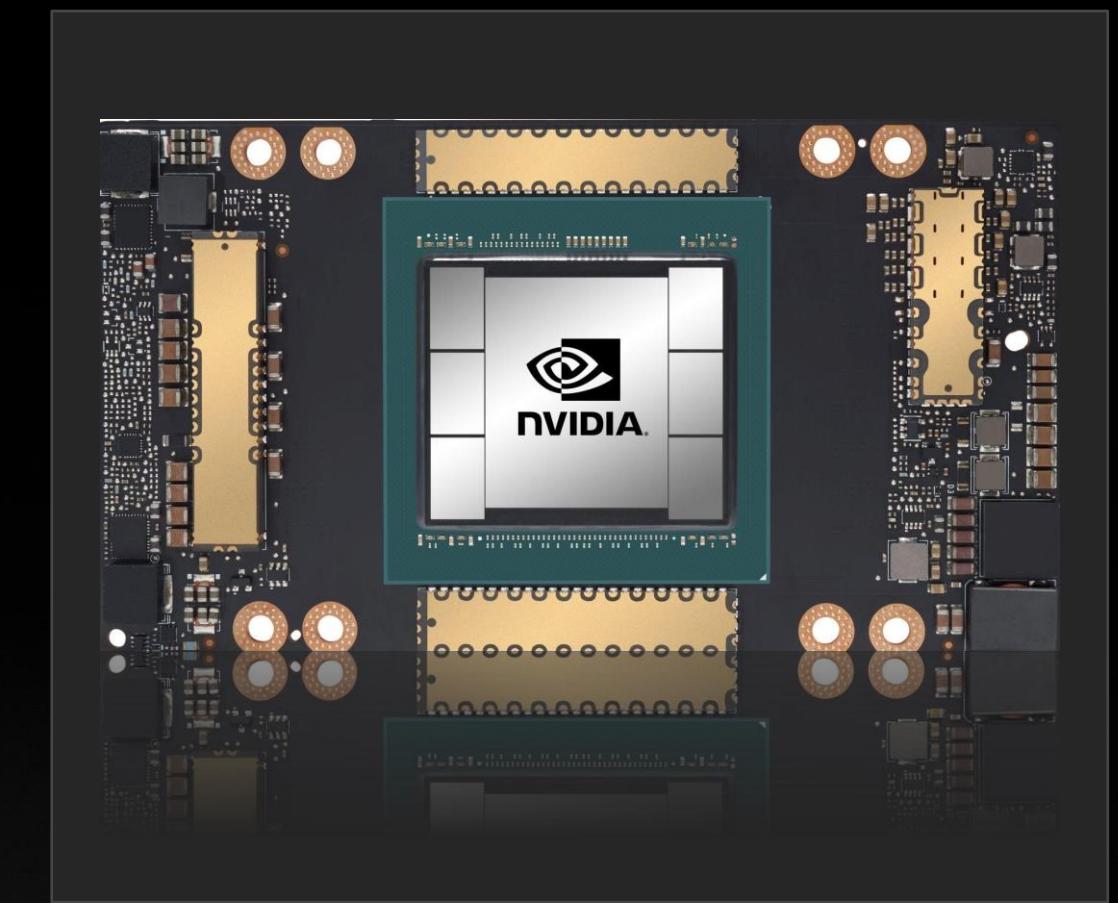
You need three main ingredients (and some skill)



LARGE QUANTITIES OF DATA



ML FRAMEWORK

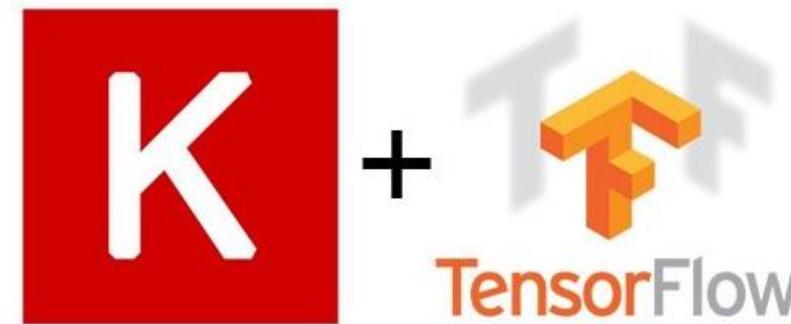


GPU ACCELERATOR

DEEP LEARNING FRAMEWORKS

Many frameworks to choose from (but not for Fortran)

PyTorch



Python

mxnet

GLUON

C++



Julia



mxnet

Caffe2

Microsoft
CNTK

TensorFlow

Keras

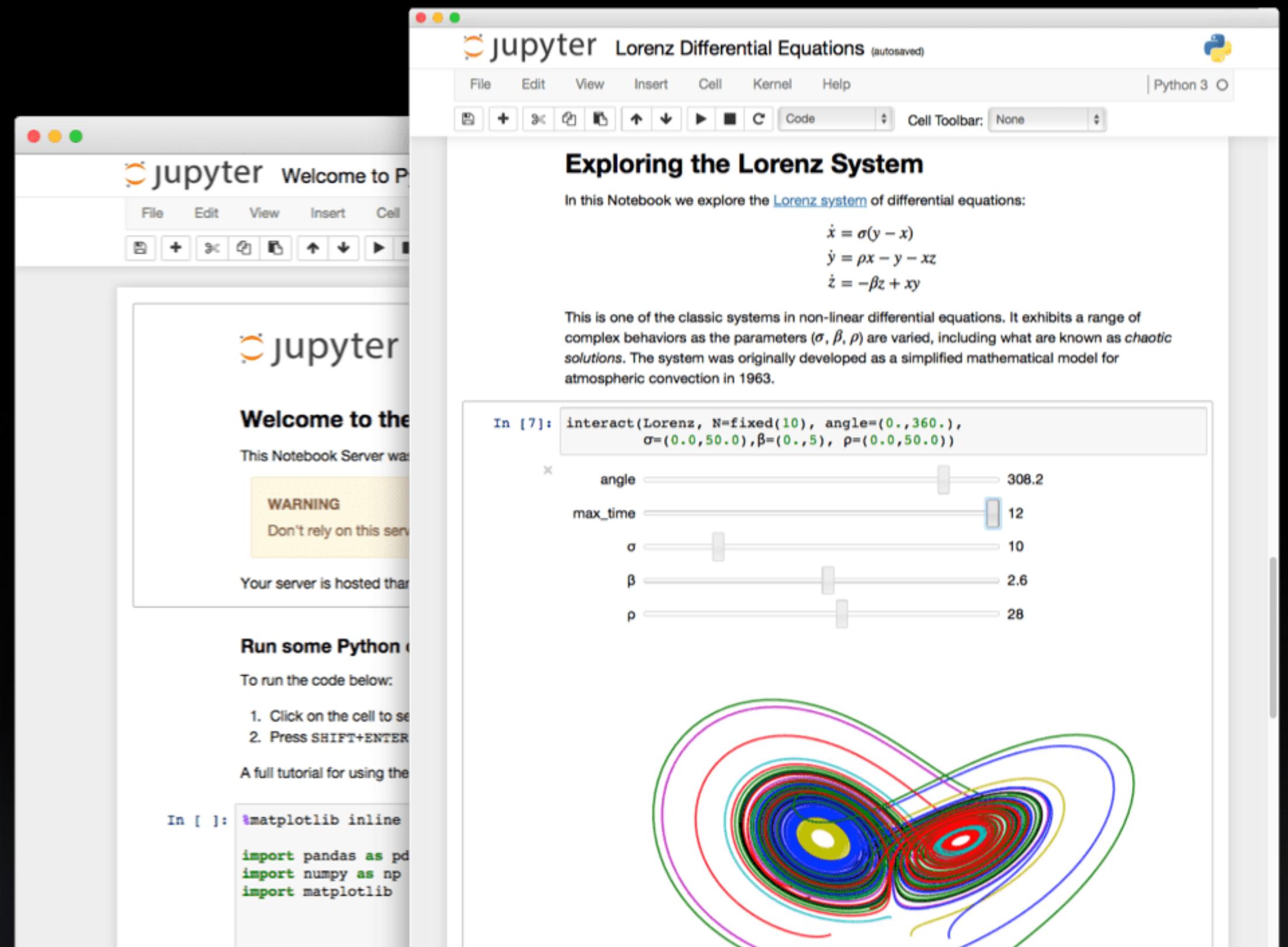
GLUON

PYTORCH

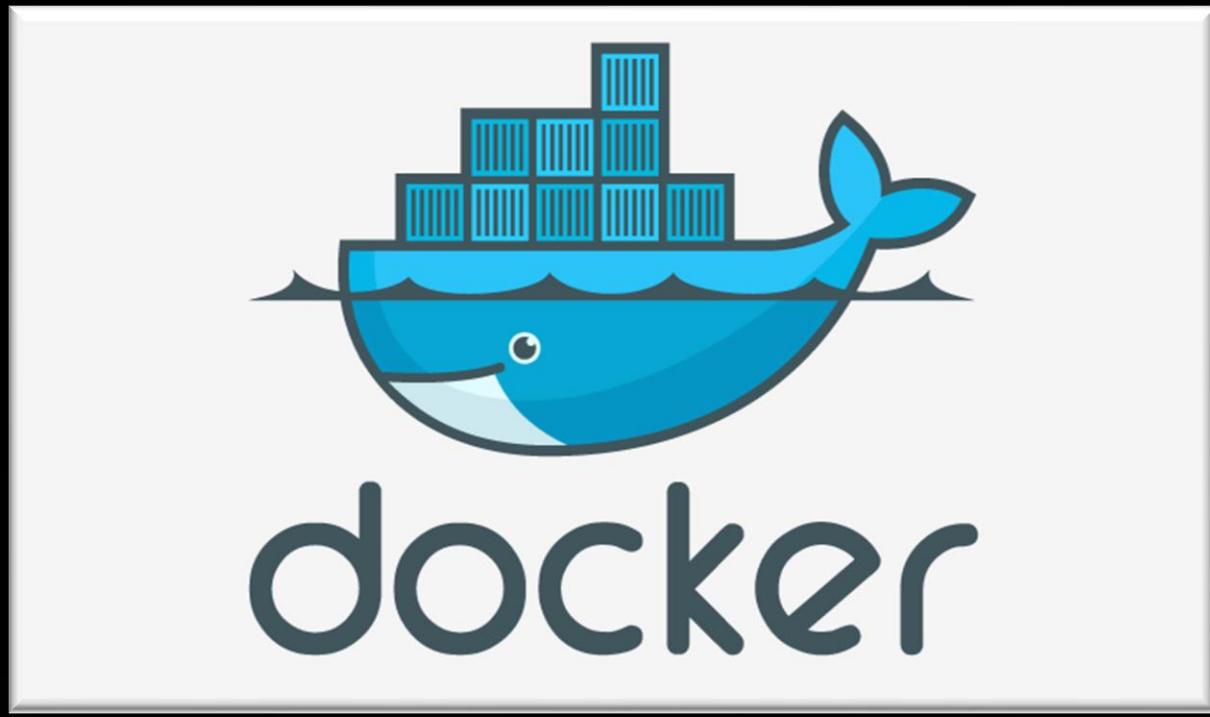
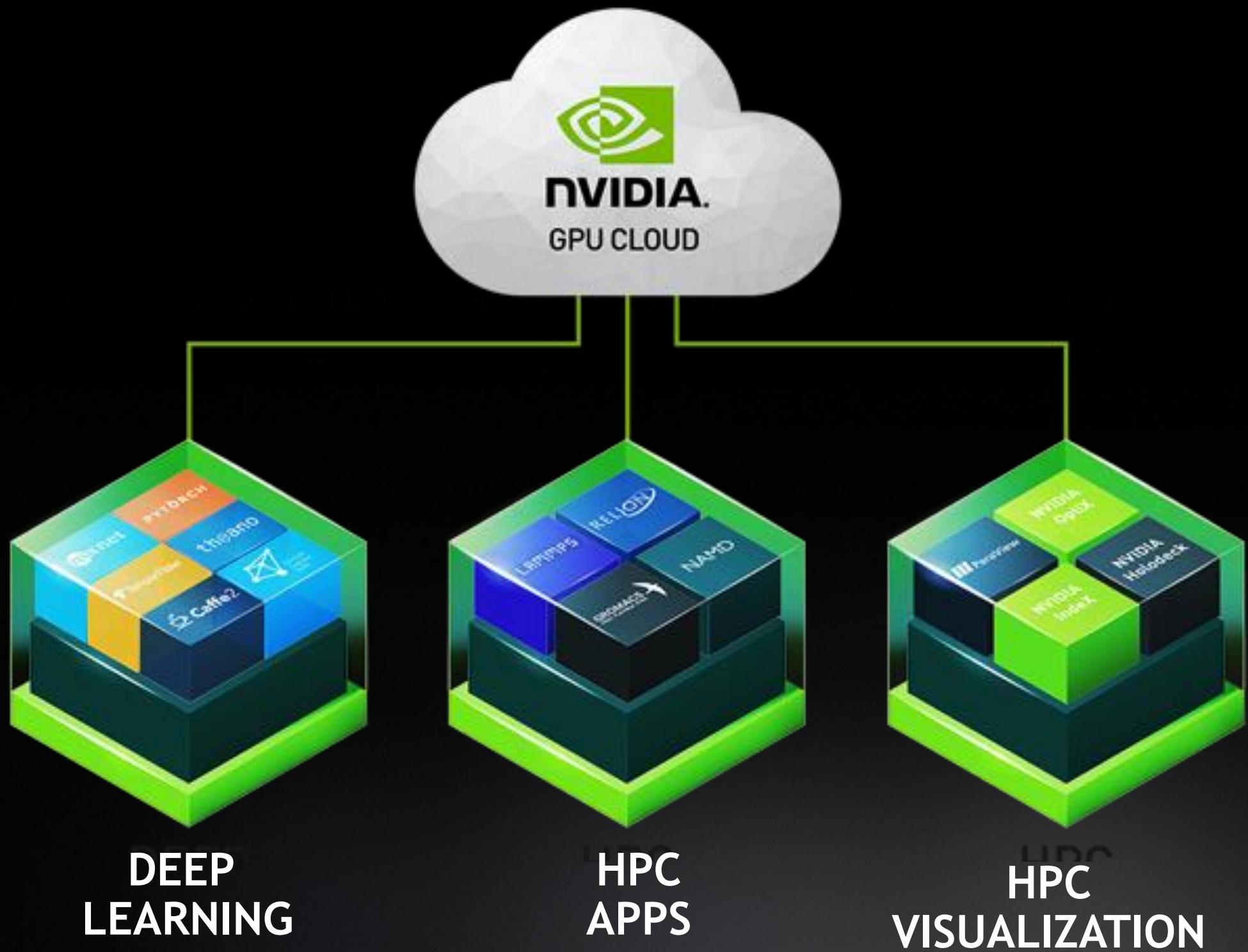
theano

DEVELOPMENT ENVIRONMENT

JUPYTER NOTEBOOKS



NVIDIA GPU CLOUD REGISTRY CONTAINERIZED SOFTWARE



LINEAR REGRESSION

With Scikit Learn

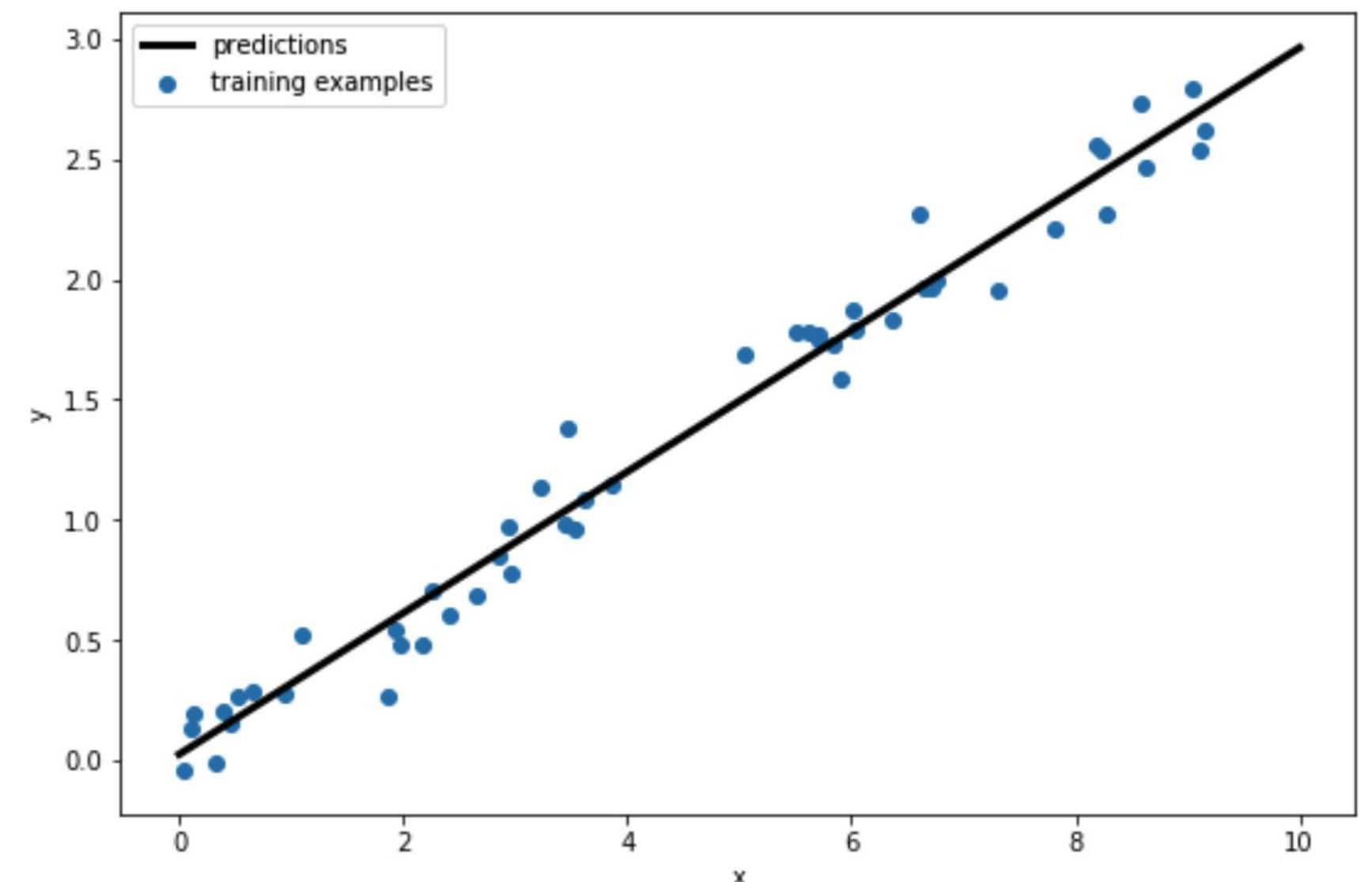
```
from sklearn.linear_model import LinearRegression
from numpy.random import rand
import numpy as np

# DATA
npts      = 50
x_train  = 10*rand(npts,1)
y_train  = 0.3*x_train + 0.5 * rand(npts,1)

# MODEL
model = LinearRegression()

# TRAIN
model.fit(x_train, y_train)

# TEST
x_test   = np.linspace(0,10,npts).reshape(-1,1)
y_predict = model.predict(x_test)
```



LINEAR REGRESSION

With Tensorflow and Keras

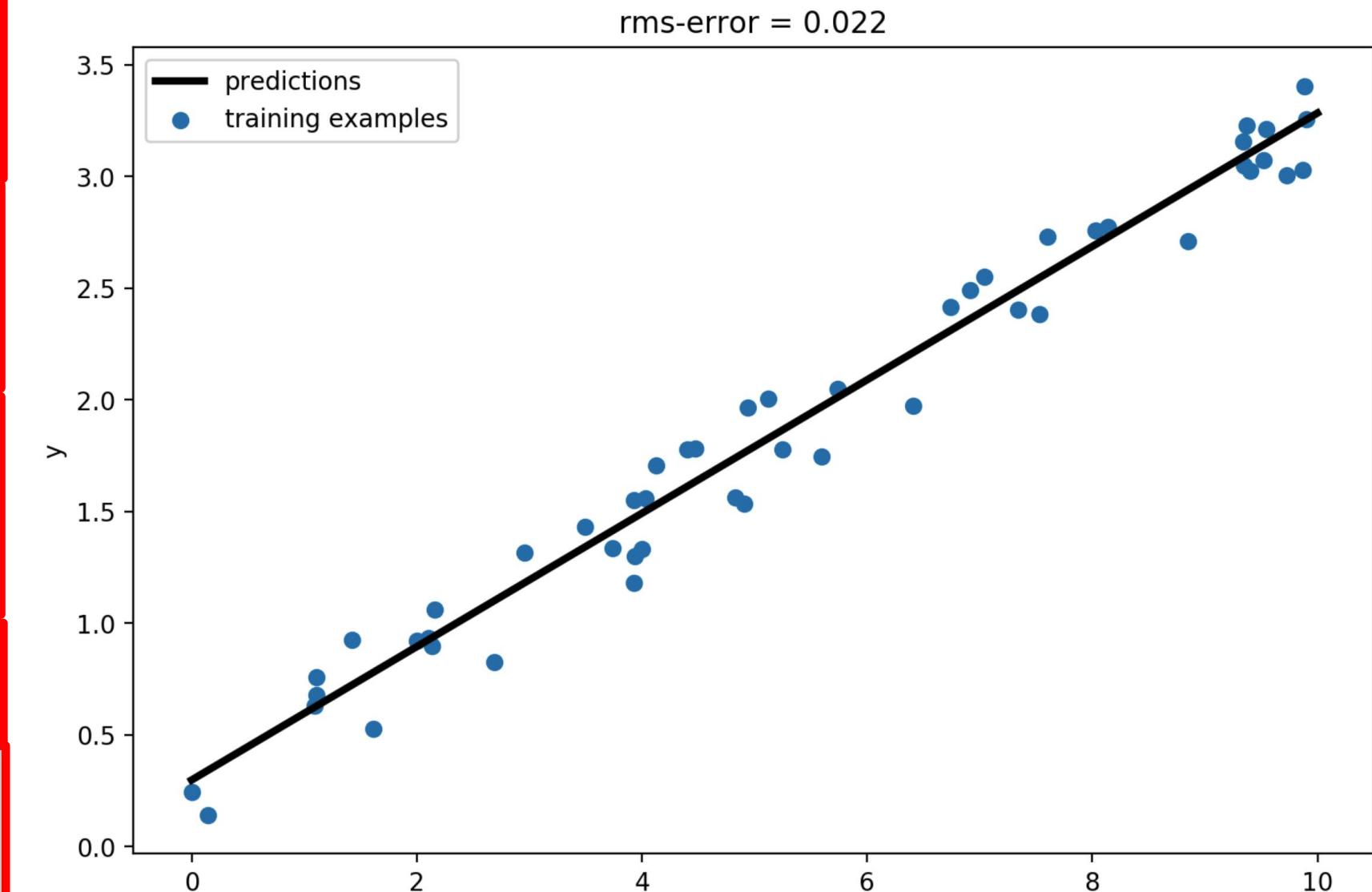
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.random import uniform
K.clear_session()

# DATA
npts      = 50
x_train   = 10*uniform(shape=[npts,1])
y_train   = 0.3*x_train + 0.5 * uniform(shape=[npts,1])

# MODEL
model = keras.models.Sequential()
model.add(keras.layers.Dense(1, input_shape=[1]))
optimizer = keras.optimizers.Adam(lr=1e-1)
model.compile(loss='mean_squared_error',optimizer=optimizer)

# TRAIN
hist = model.fit(x_train,y_train,epochs=100,verbose=0)

# TEST
x_test    = tf.linspace(0,10,npts)
x_test    = tf.reshape(x_test,[npts,1])
y_predict = model.predict(x_test)
```

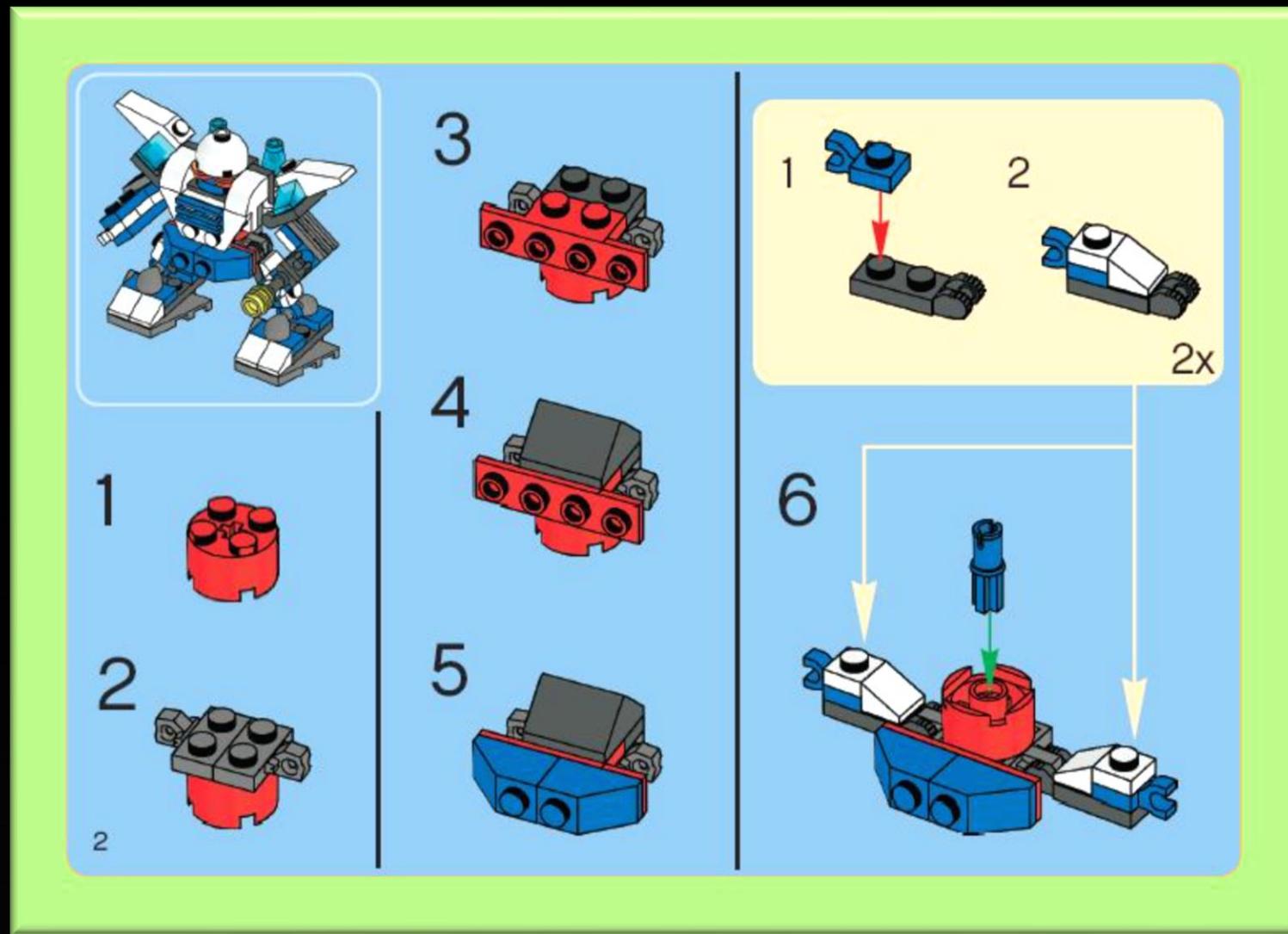




TRAINING

TRAINING VS INFERENCE

TRAINING PHASE



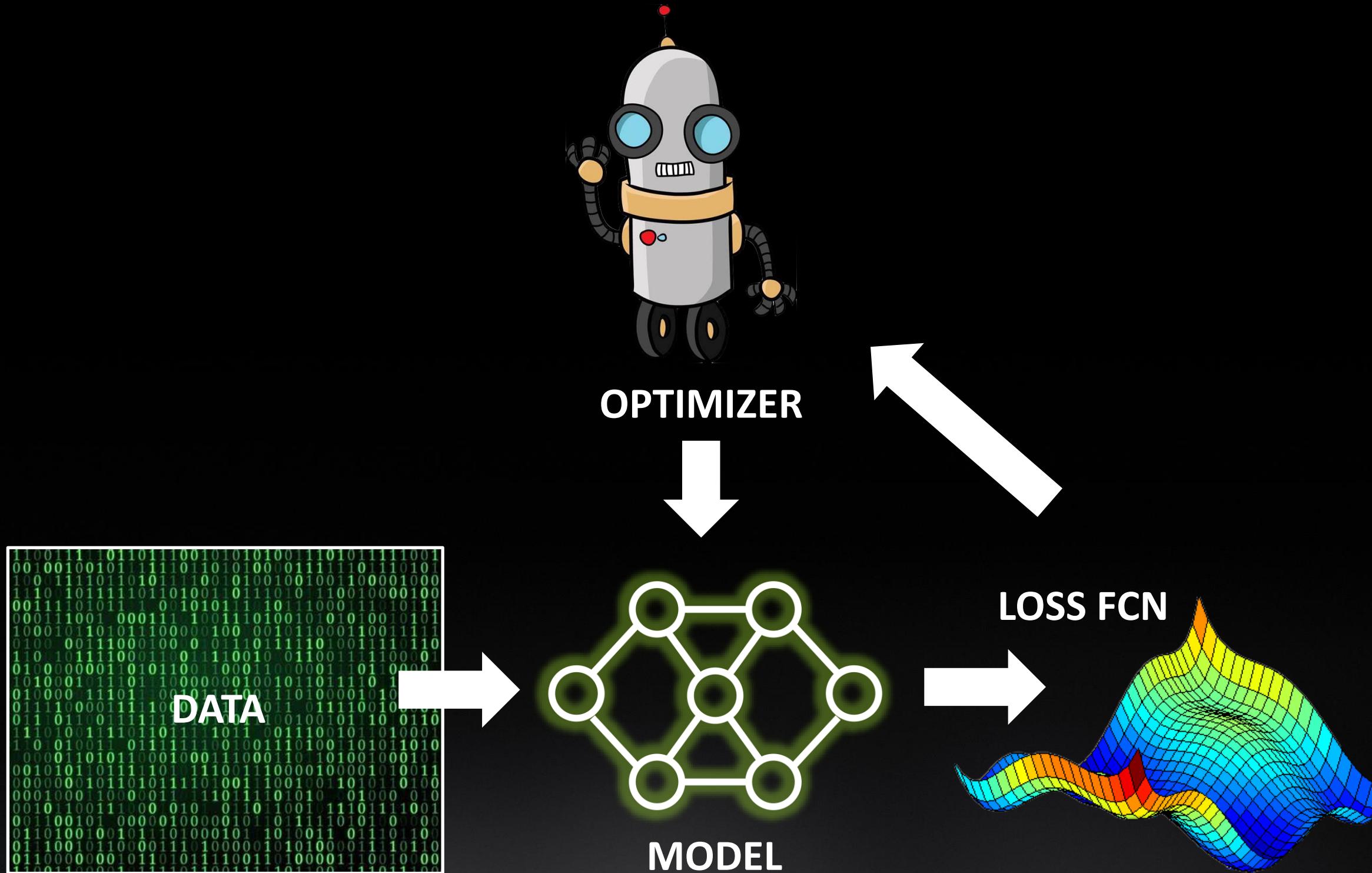
SEARCH FOR THE RIGHT PIECES



APPLY THE COMPLETED MODEL

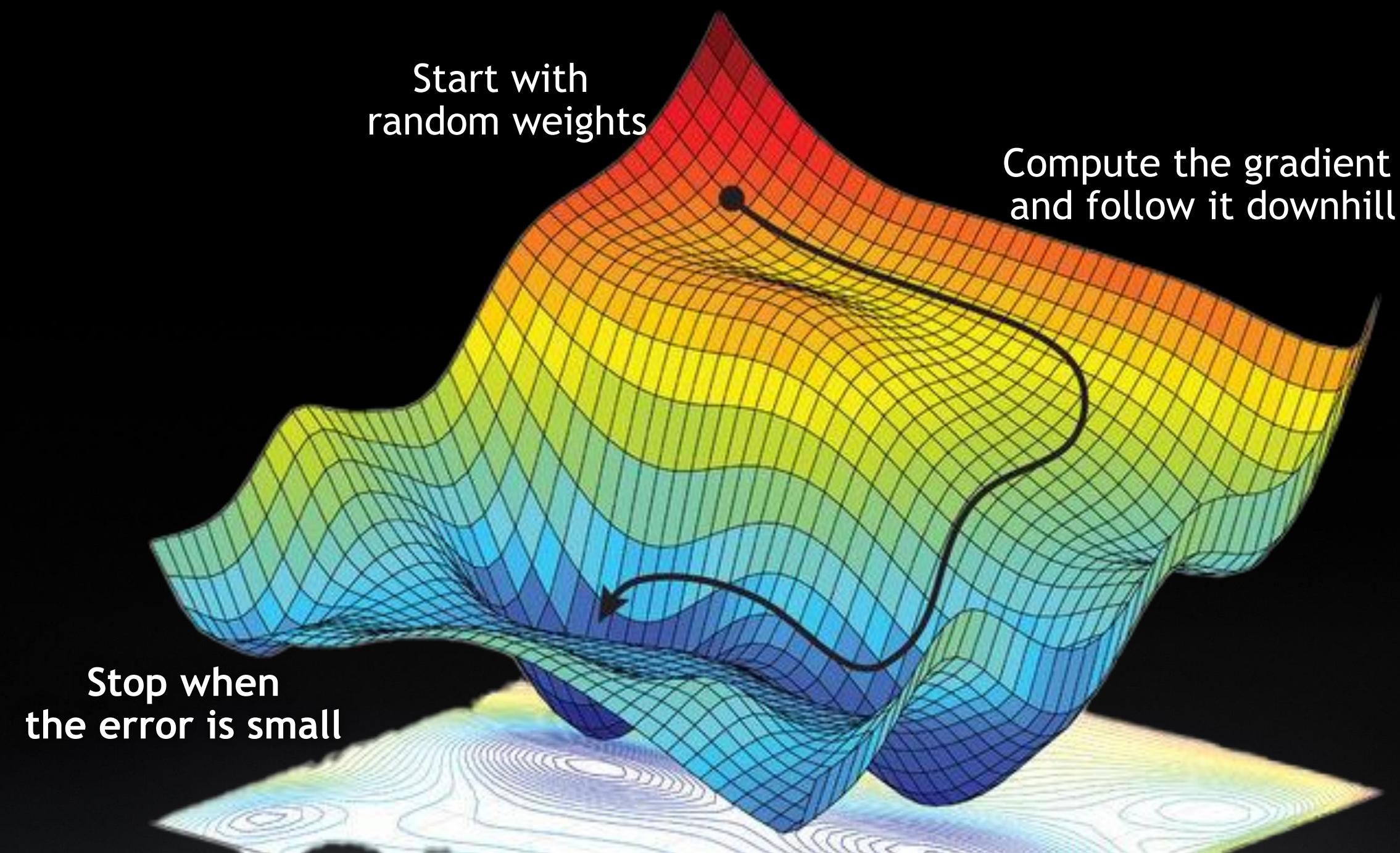
TRAINING: THE PLAYERS

DATA, MODEL, LOSS, AND OPTIMIZER



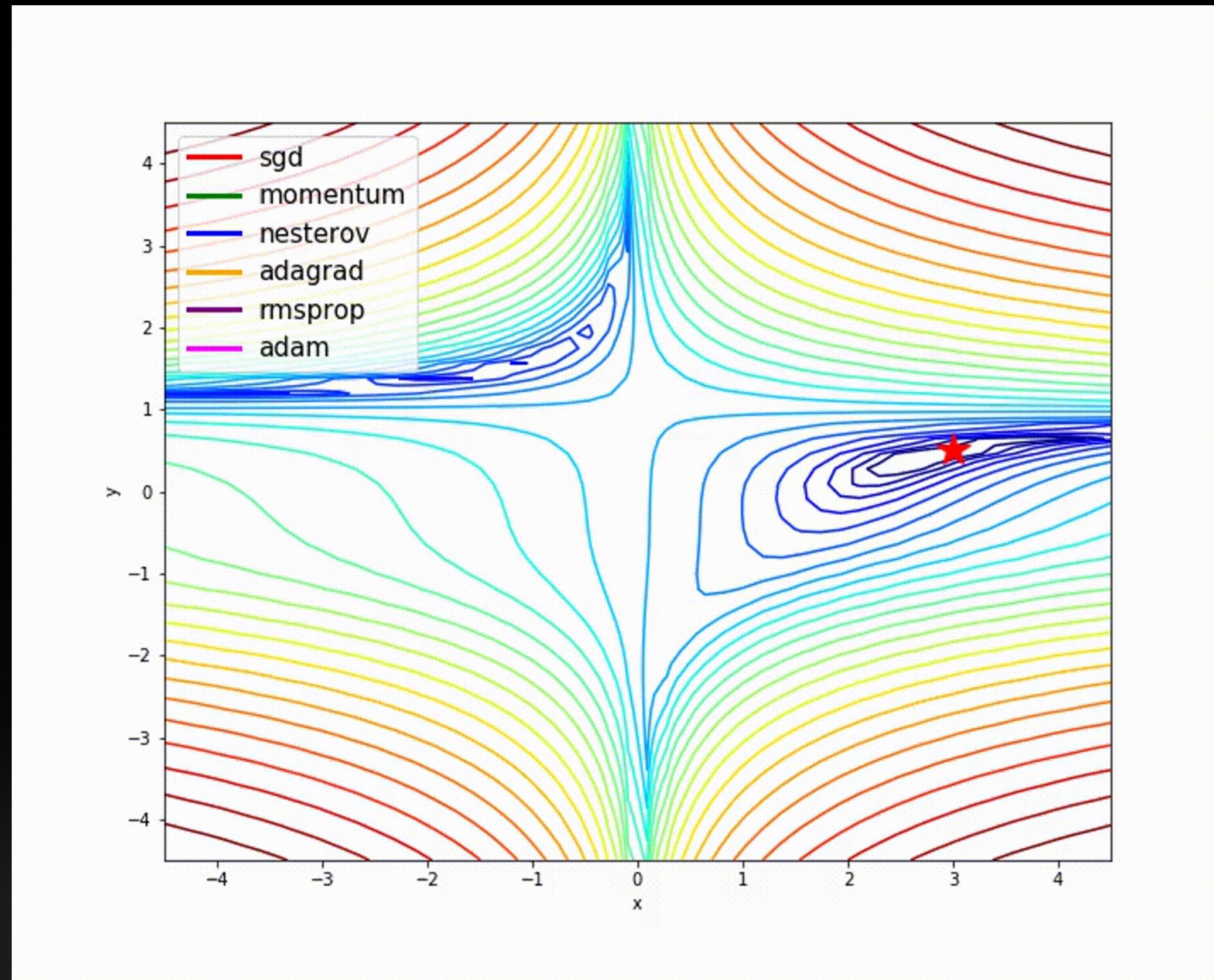
TRAINING: GRADIENT DESCENT

Finding a solution is as easy as falling down a hill



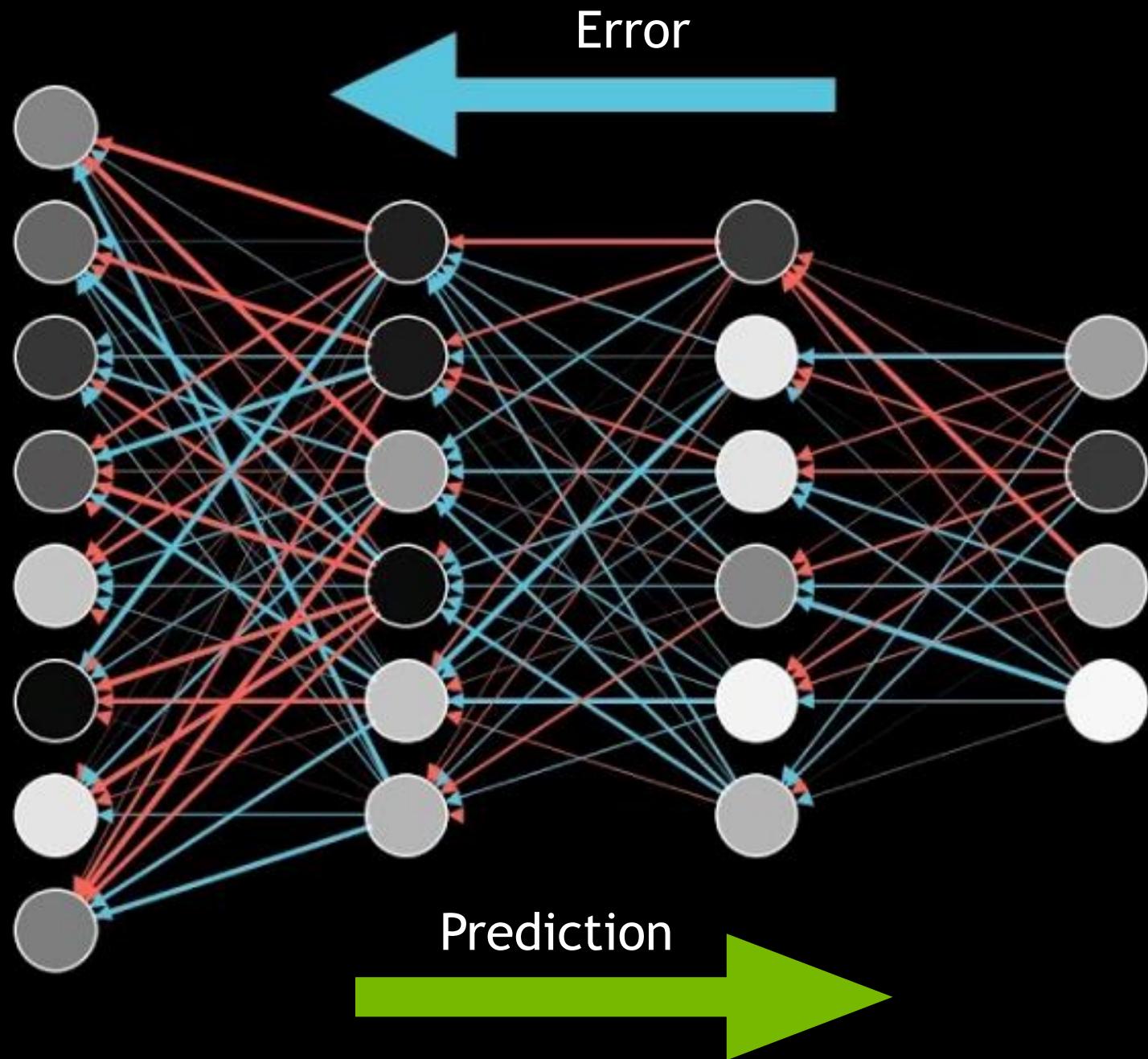
OPTIMIZERS

Many variations on stochastic gradient descent



TRAINING: BACKPROPAGATION

Compute the gradient, by efficiently assigning blame

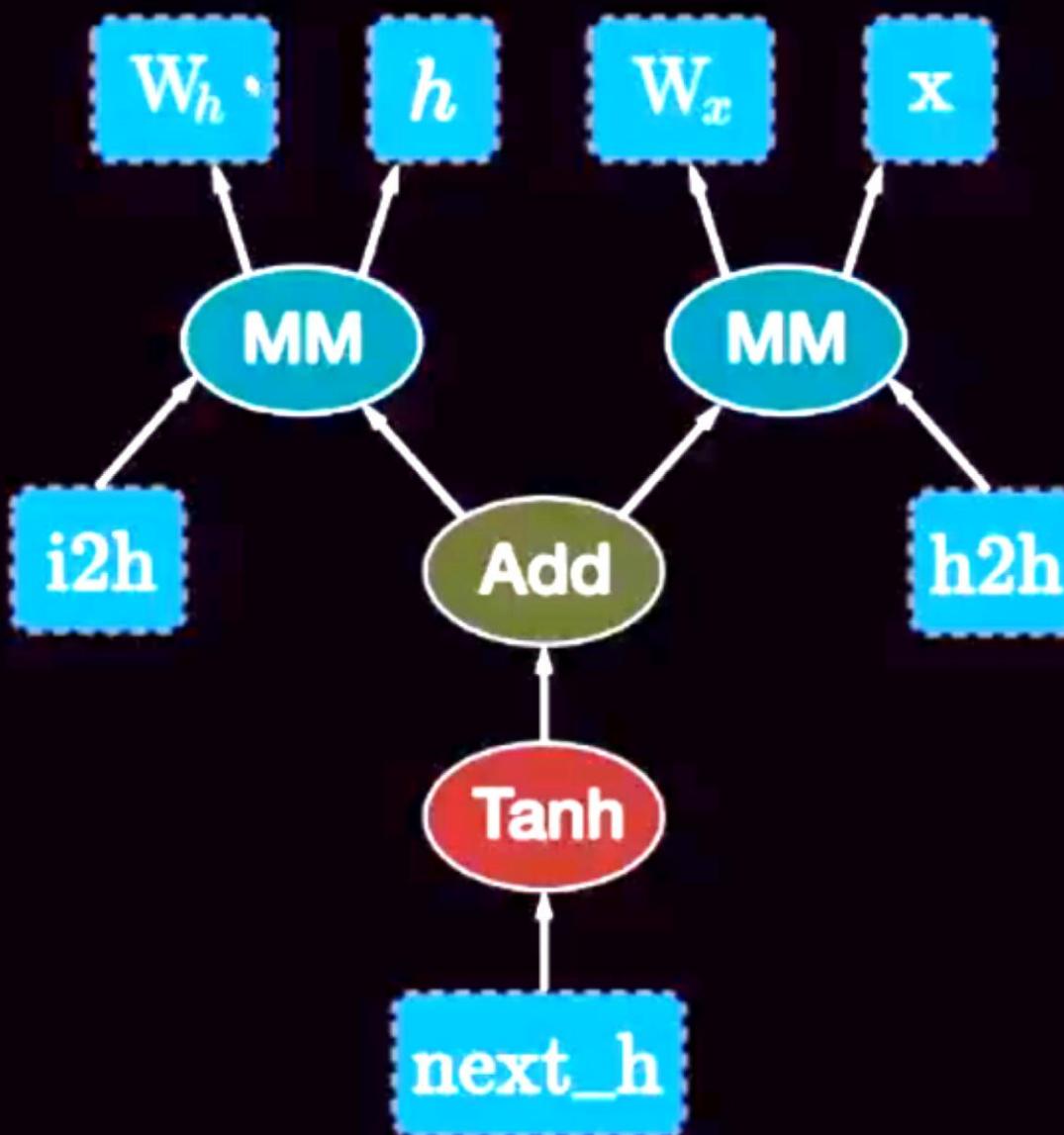


AUTOGRAD

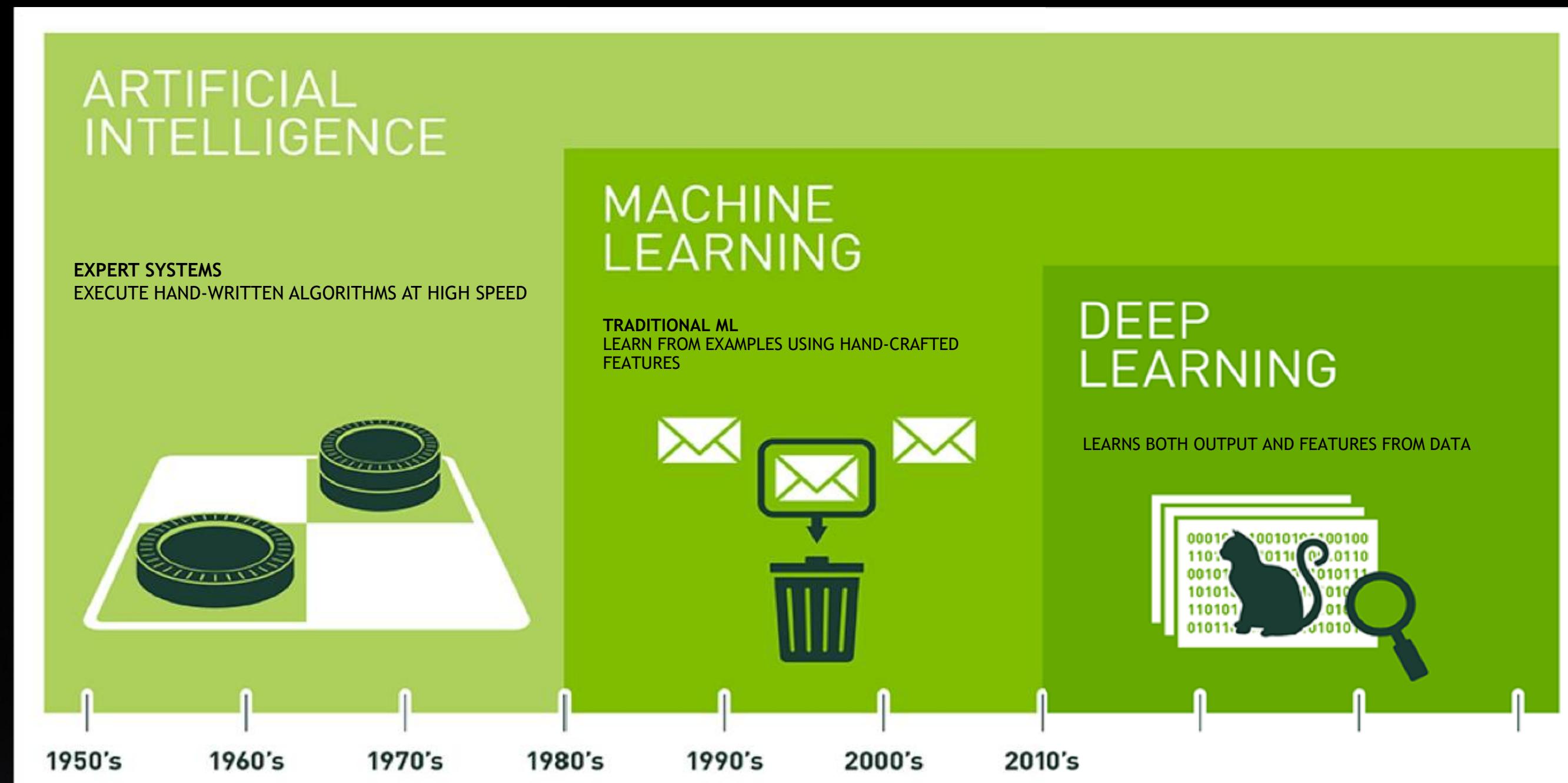
Let a framework keep track of your gradient, so you don't have to

PyTorch Autograd

```
from torch.autograd import Variable  
  
x = Variable(torch.randn(1, 10))  
prev_h = Variable(torch.randn(1, 20))  
W_h = Variable(torch.randn(20, 20))  
W_x = Variable(torch.randn(20, 10))  
  
i2h = torch.mm(W_x, x.t())  
h2h = torch.mm(W_h, prev_h.t())  
next_h = i2h + h2h  
next_h = next_h.tanh()  
  
next_h.backward(torch.ones(1, 20))
```

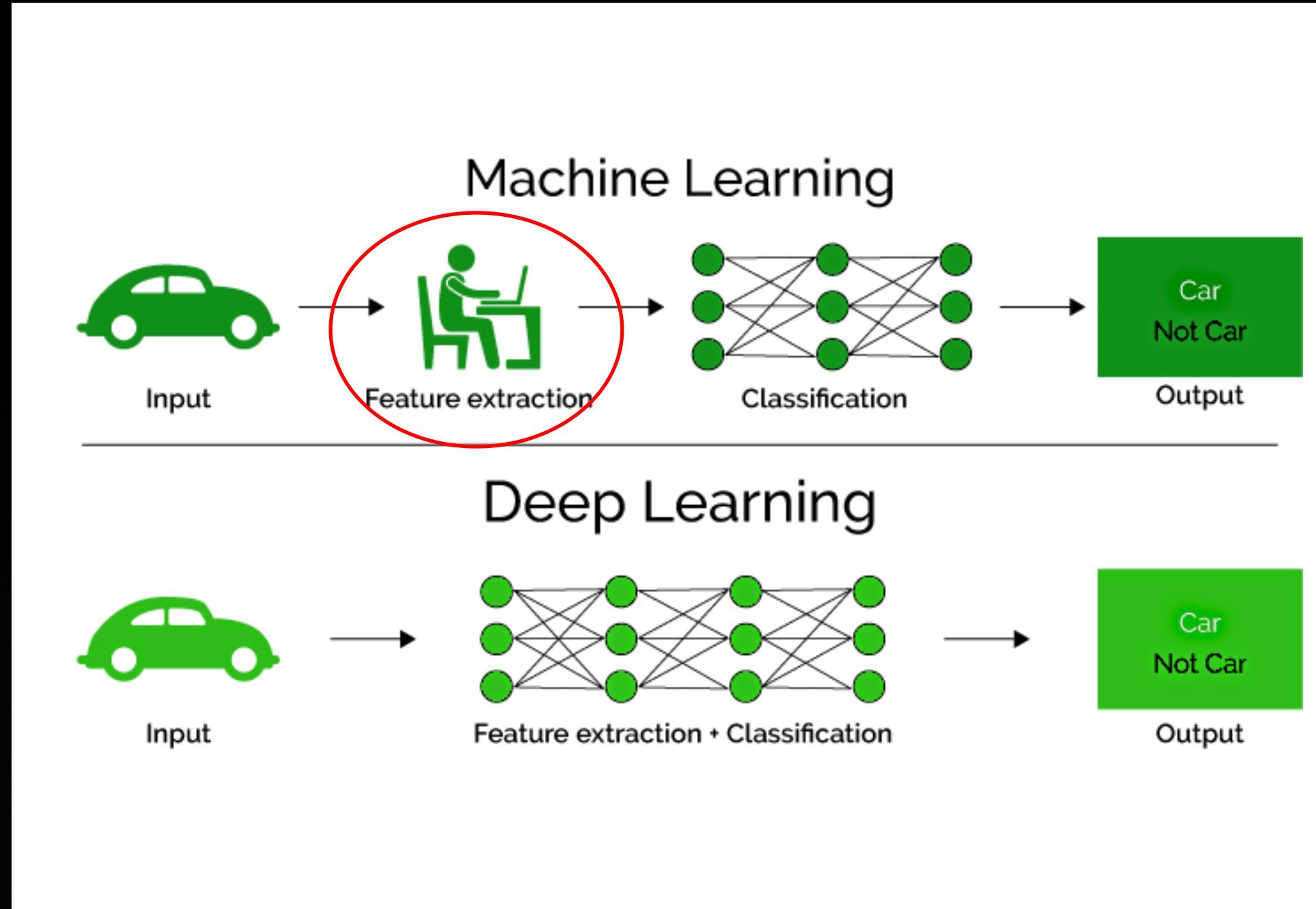


AI, MACHINE LEARNING, DEEP LEARNING



DEEP LEARNING VS. MACHINE LEARNING

When should I use deep learning vs traditional machine learning?



TRADITIONAL MACHINE LEARNING

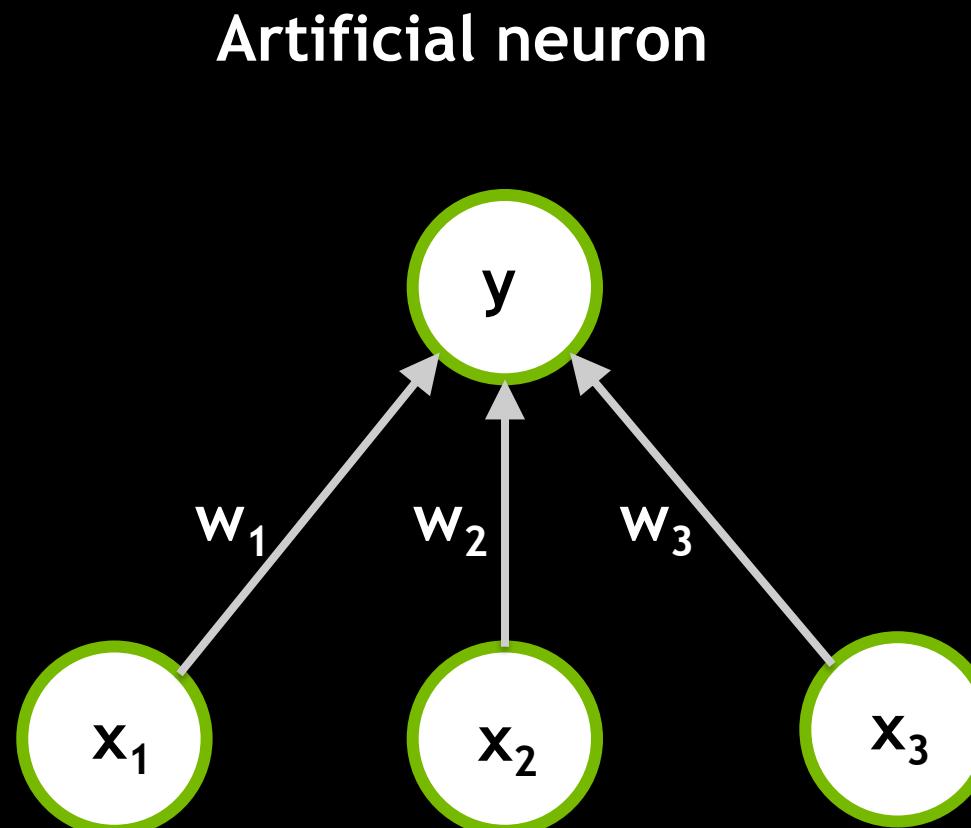
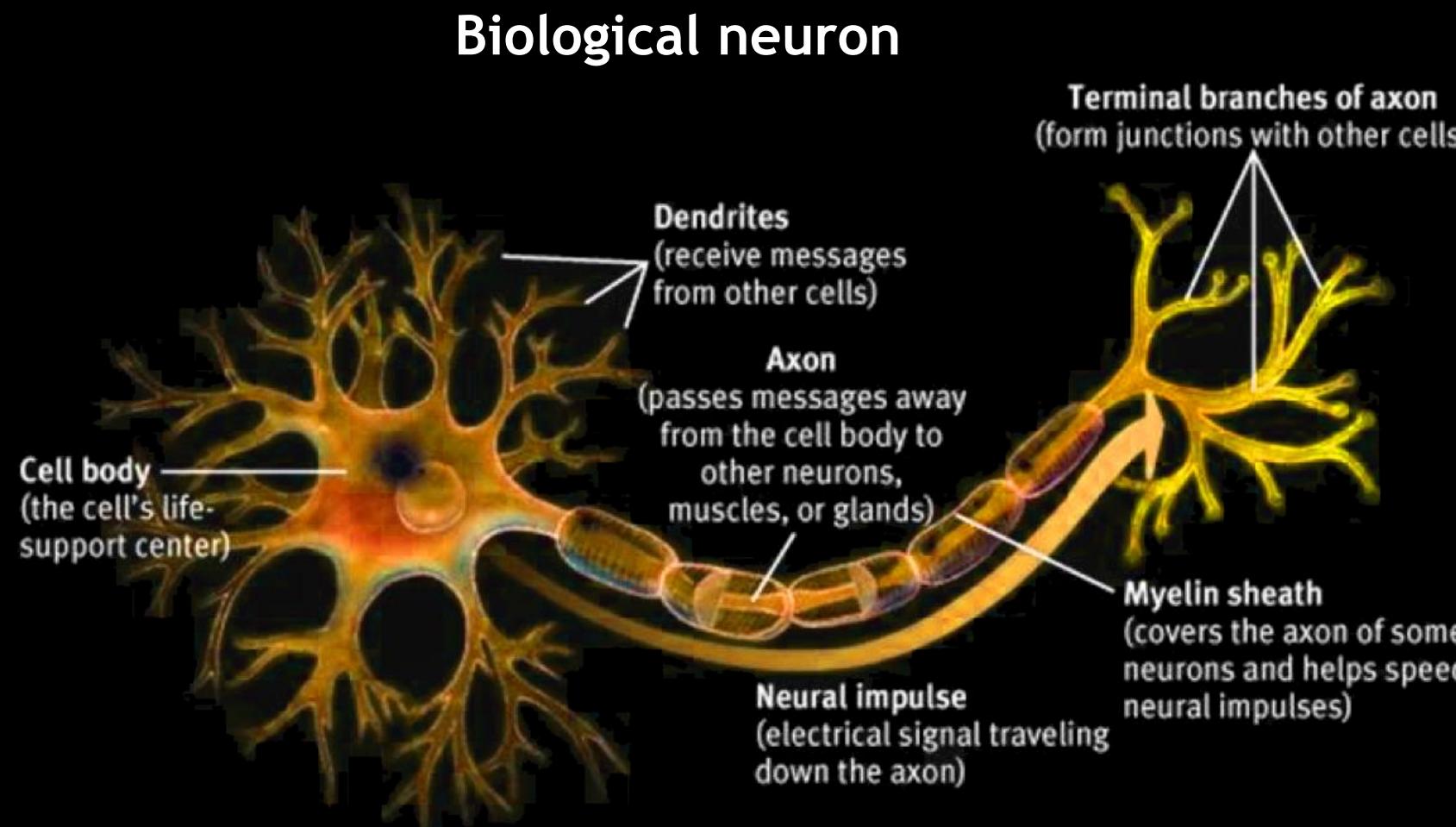
Random forests, SVM, K-means, Logistic Regression
Features hand-crafted by experts
Small set of features: 10s or 100s
NVIDIA RAPIDS: orders of magnitude speedup

SUPERVISED DEEP LEARNING

CNN, RNN, LSTM, GAN, Variational Auto-encoders
Finds features automatically
High dimensional data: images, sounds, speech
Large set of labelled data (10k+ examples)
NVIDIA CU-DNN: accelerates DL frameworks

ARTIFICIAL NEURONS

Simple equations with adjustable parameters



$$y = f(w_1x_1 + w_2x_2 + w_3x_3)$$

<https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>

CURVE FIT WITH SINGLE LAYER NEURAL NETWORK

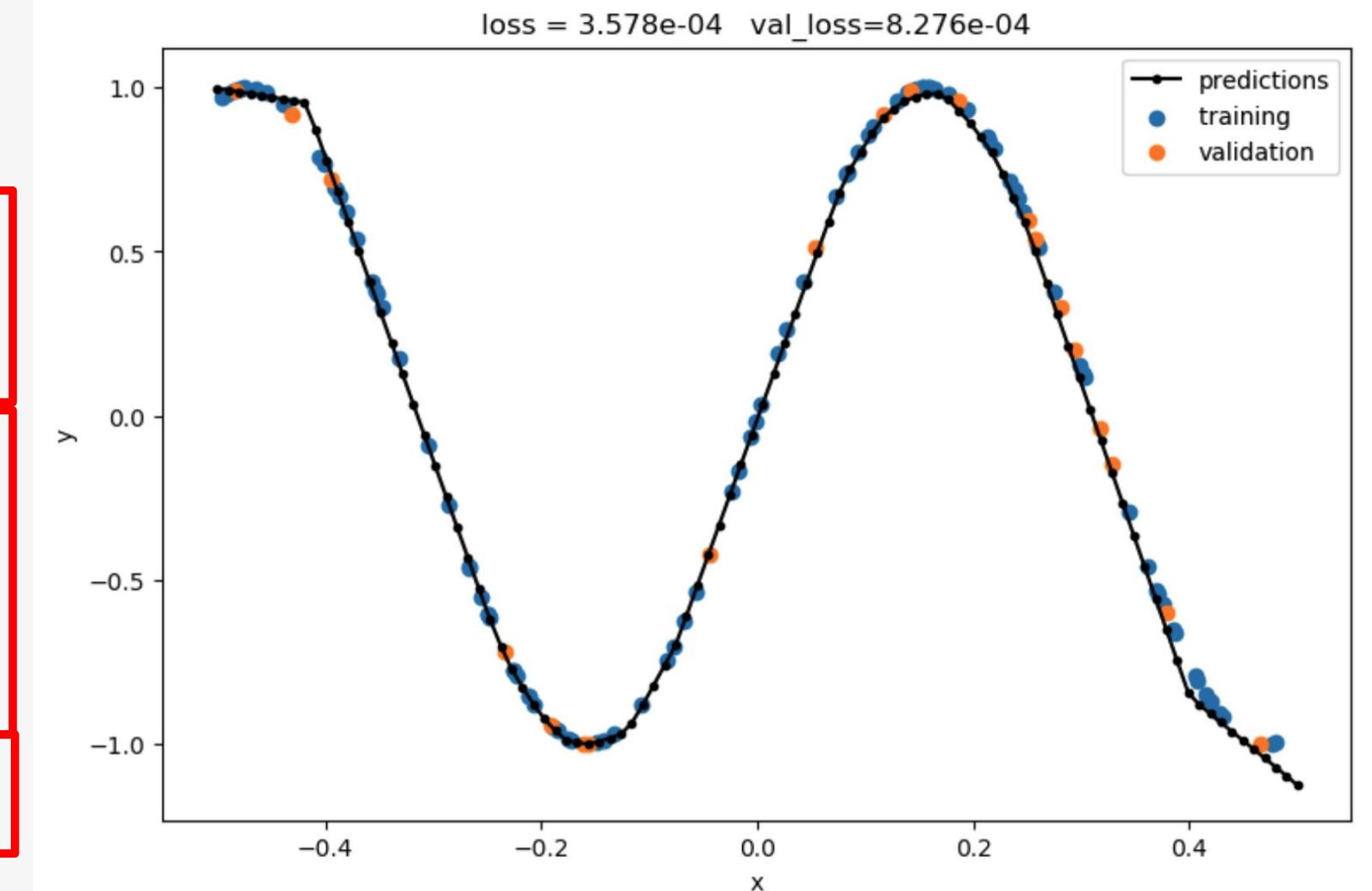
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.random import uniform
K.clear_session()

# DATA
def f(x): return tf.math.sin(10*x)
x_train = uniform(shape=[100,1]) - 0.5
x_val   = uniform(shape=[ 20,1]) - 0.5
y_train = f(x_train)
y_val   = f(x_val)

# MODEL (using keras functional API)
inputs  = keras.Input(shape=[1])
x       = keras.layers.Dense(units=100,activation="relu")(inputs)
x       = keras.layers.Dense(1)(x)
outputs = keras.layers.Add()([inputs,x])
model   = keras.Model(inputs=inputs, outputs=outputs)
optimizer= keras.optimizers.Adam(lr=1e-2)
model.compile(loss='mean_squared_error',optimizer=optimizer)

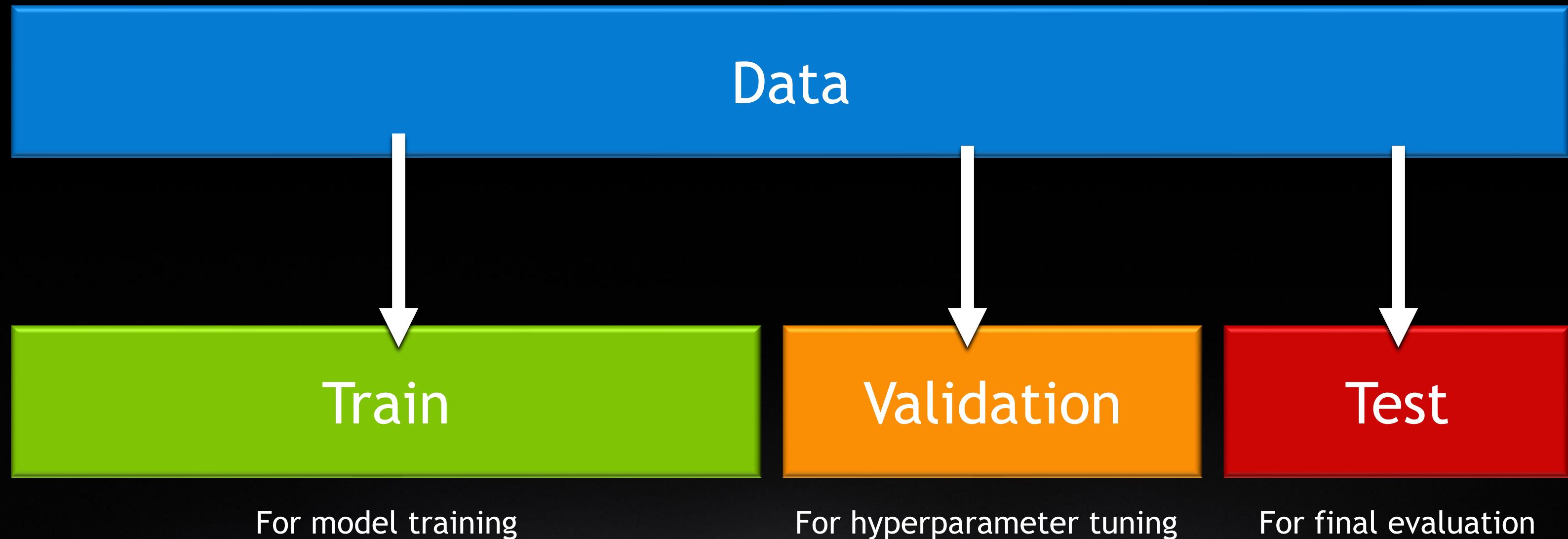
# TRAIN
hist = model.fit(x_train,y_train,validation_data=(x_val,y_val),
                  epochs=5000,verbose=0, batch_size=1)

# TEST
x_test   = tf.linspace(-.5,.5,100)
y_predict = model.predict(x_test)
```



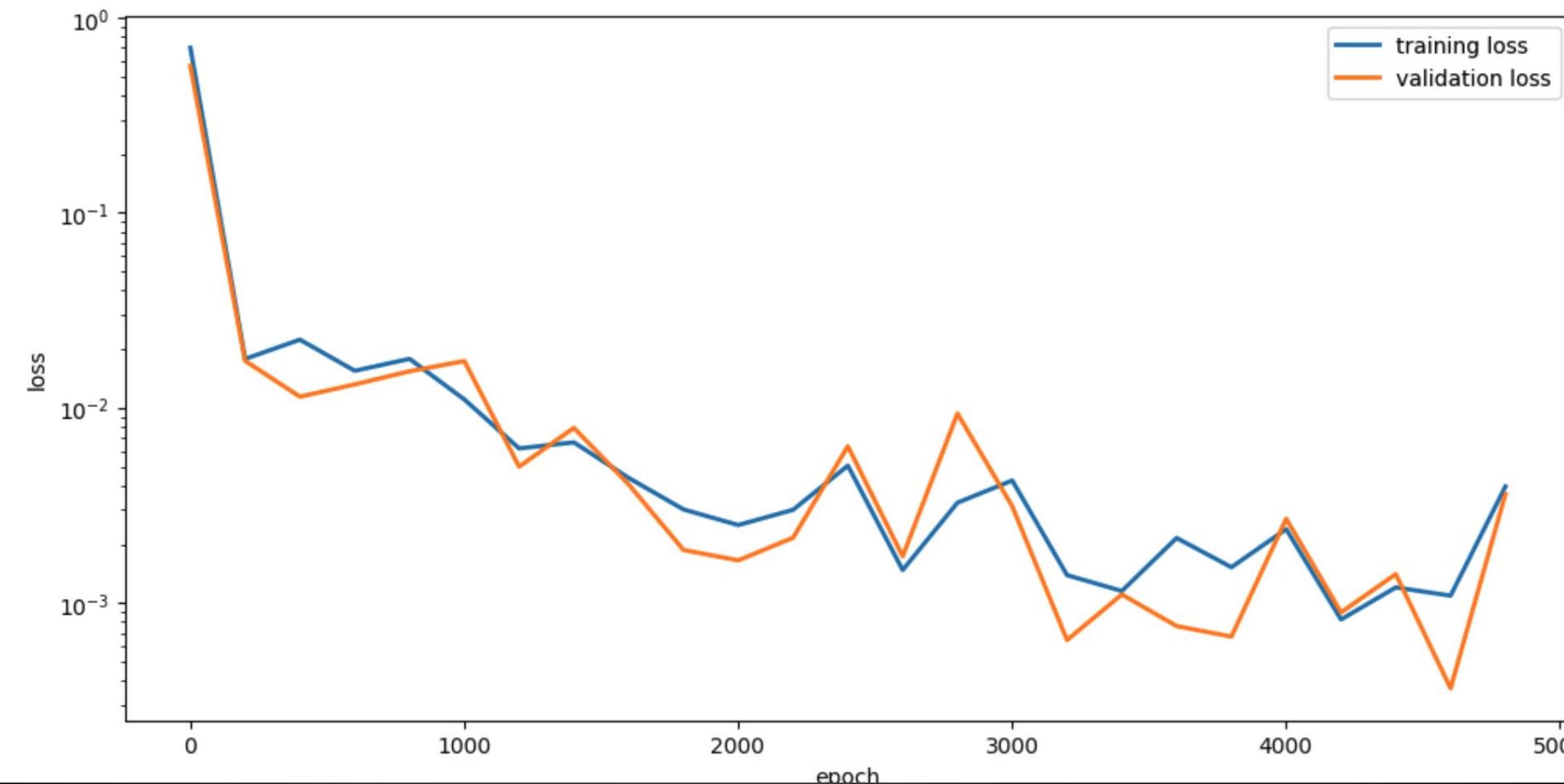
DATA SPLITTING

KEEP TEST, TRAINING, AND VALIDATION DATA SEPERATE

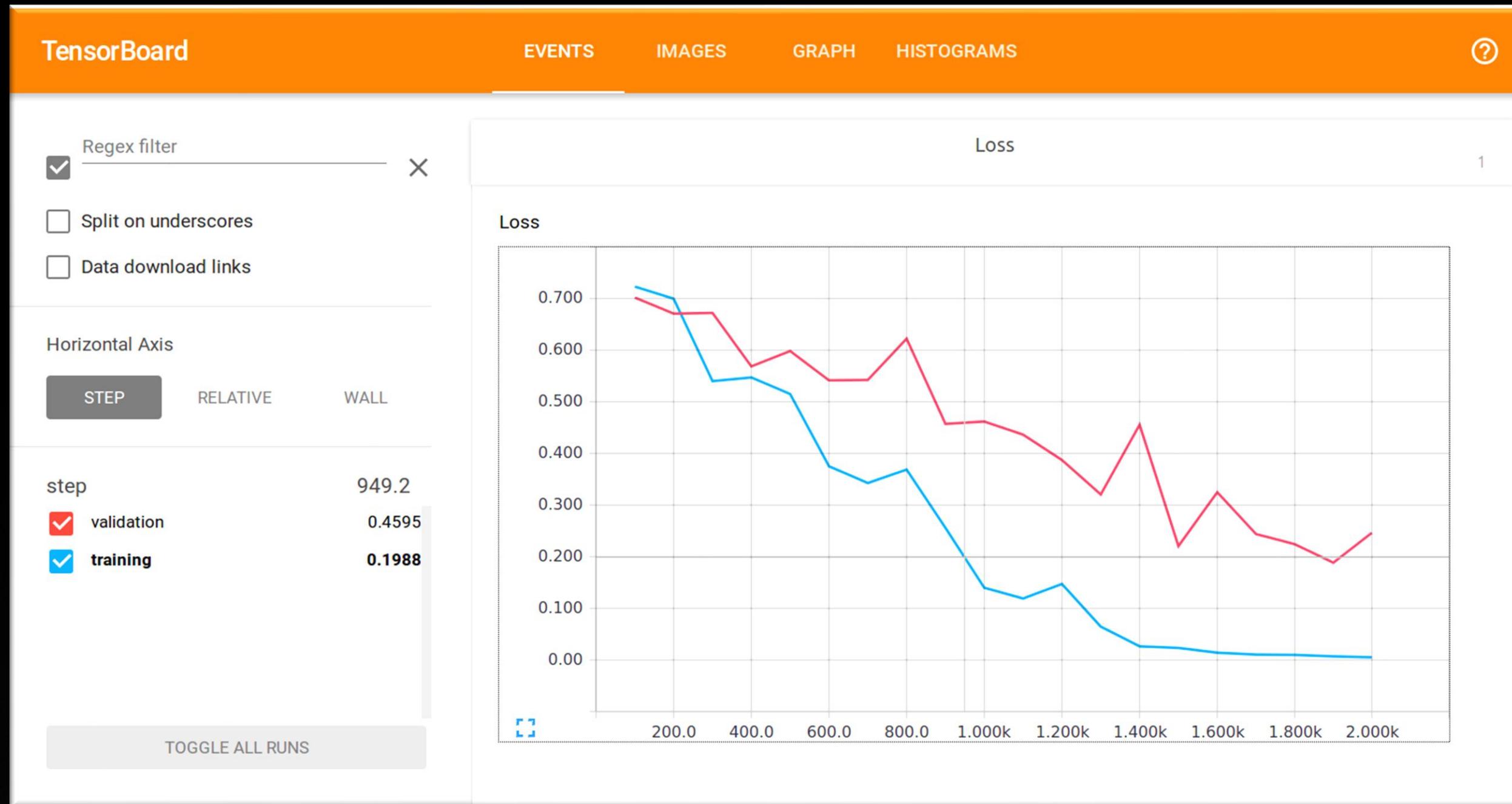


PLOTTING TRAINING AND VALIDATION LOSS

```
1 %matplotlib inline
2 import matplotlib.pyplot as plt
3
4 t_loss = hist.history['loss']
5 v_loss = hist.history['val_loss']
6 epoch = tf.range(1,len(t_loss)+1)
7 step = 200
|
9 plt.figure(figsize=(12,6),dpi=100)
10 plt.semilogy(epoch[::step],t_loss[::step],linewidth=2,label='training loss')
11 plt.semilogy(epoch[::step],v_loss[::step],linewidth=2,label='validation loss')
12 plt.xlabel('epoch'); plt.ylabel('loss'); plt.legend()
```



VISUALIZATION TOOLS: TENSORBOARD



```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=".logs")
model.fit(x_train, y_train, epochs=2, callbacks=[tensorboard_callback])
# run the tensorboard command to view the visualizations.
```

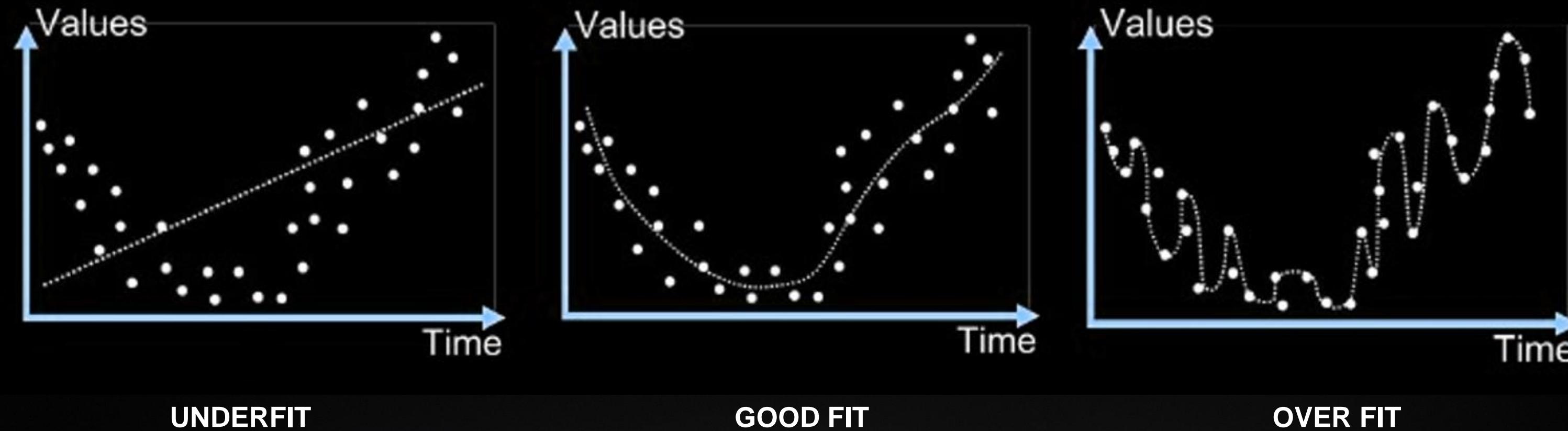
```
tensorboard --logdir=path_to_your_logs
```



MODEL CAPACITY AND REGULARIZATION

MODEL CAPACITY

A good model is one that generalizes to new data



GOOD-FIT

Checking for Generalization

Training Loop

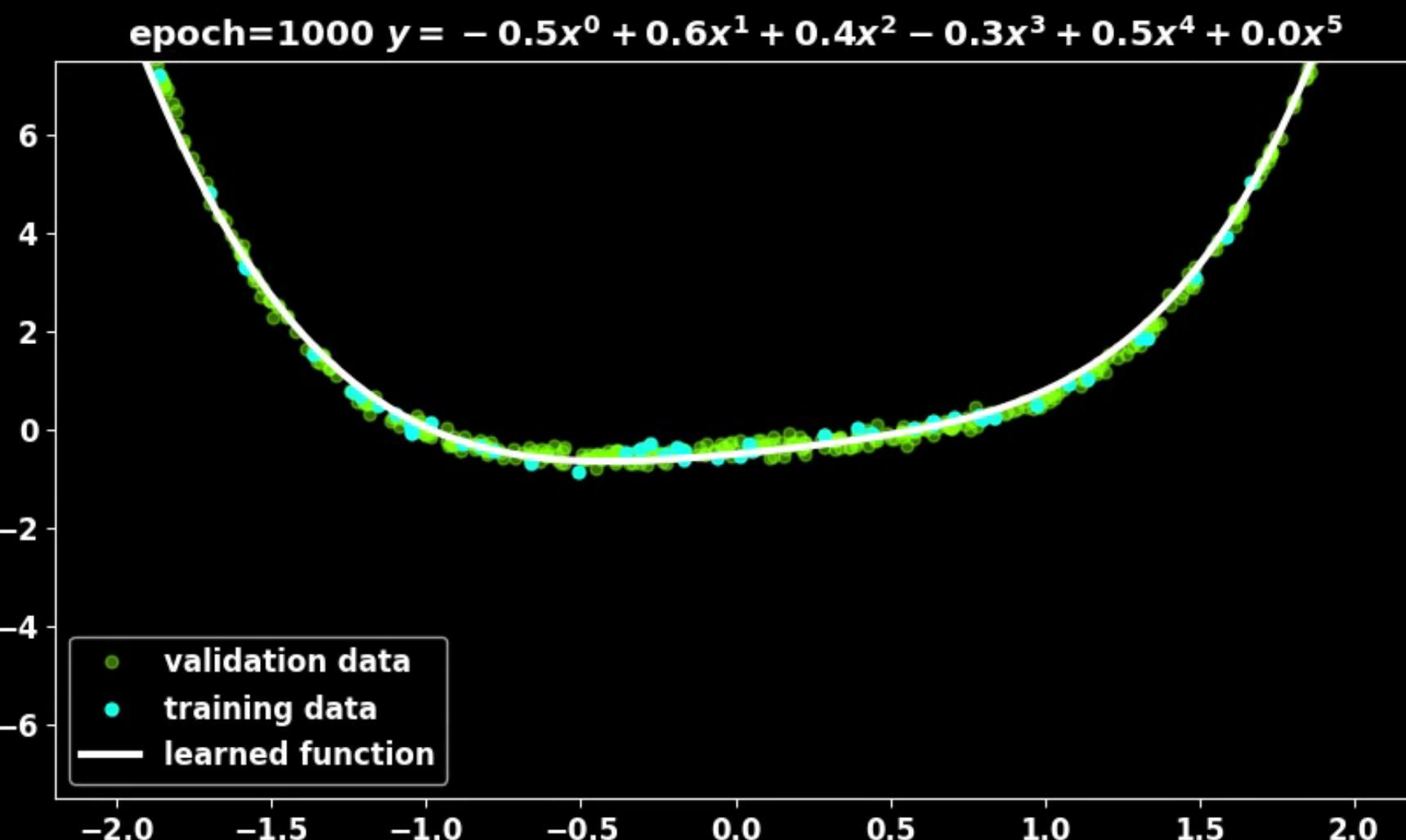
```
xtrain, ytrain, xval, yval = load_data(npts=500, train_fraction = 0.10)

model      = taylor_series(order=5)
optimizer = torch.optim.Adam(model.params, lr=1.0e-2)
epochs     = 1000

for i in range(epochs+1):

    # training
    optimizer.zero_grad()
    yhat = model(xtrain)
    loss = (yhat - ytrain).pow(2).mean()
    loss.backward()
    optimizer.step()

    # validation
    yval_hat = model(xval)
    loss_val = (yval_hat - yval).pow(2).mean()
```



OVER-FITTING

Captures training data, but generalizes poorly

Training Loop

```
xtrain, ytrain, xval, yval = load_data(npts=500, train_fraction = 0.02)

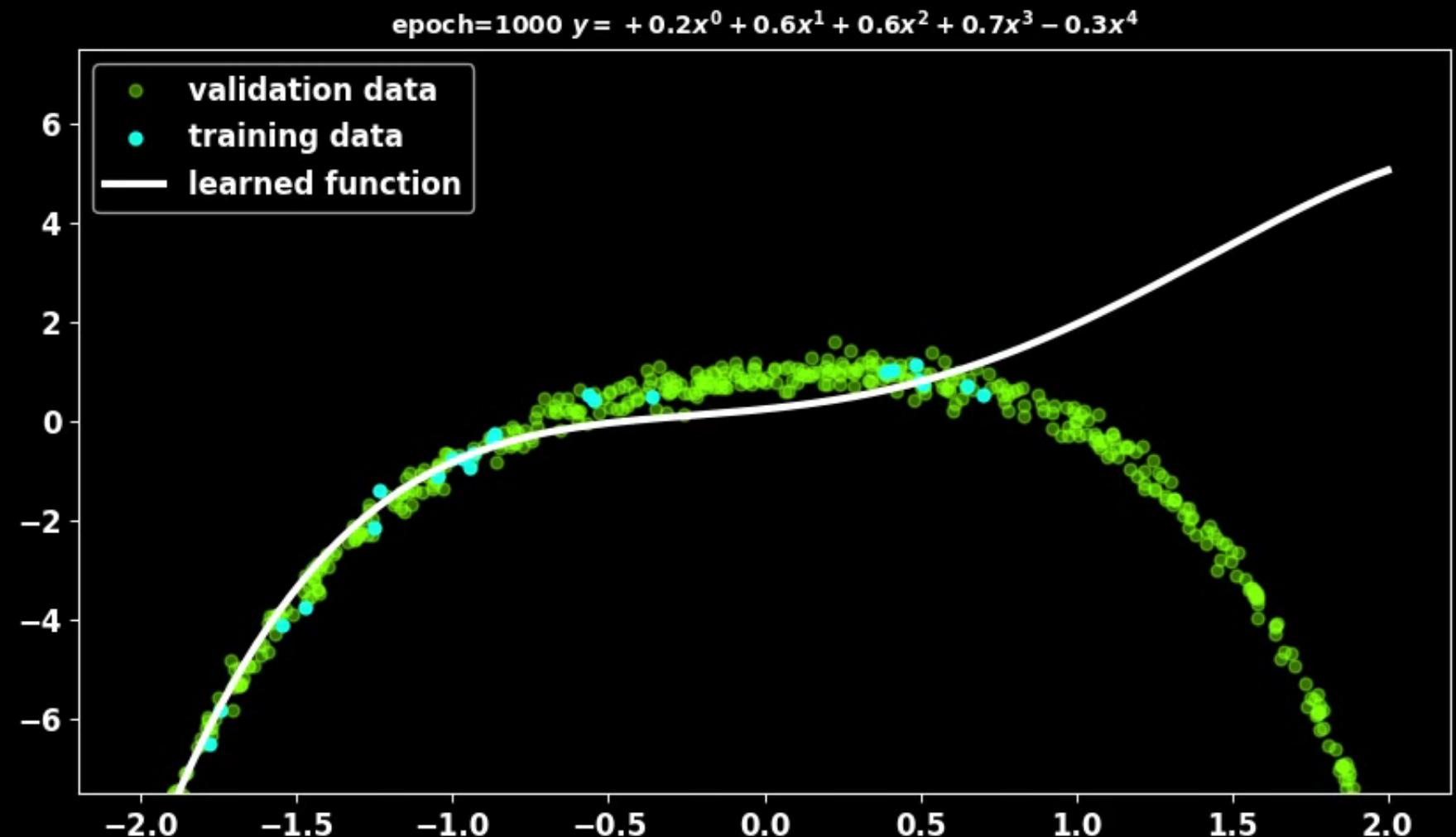
model      = taylor_series(order=8)
optimizer = torch.optim.Adam(model.params, lr=5.0e-3)
epochs     = 1000

for i in range(epochs+1):

    # training
    optimizer.zero_grad()
    yhat = model(xtrain)
    loss = (yhat - ytrain).pow(2).mean()
    loss.backward()
    optimizer.step()

    # validation
    yval_hat = model(xval)
    loss_val = (yval_hat - yval).pow(2).mean()
```

Use more data points
Reduce model capacity



UNDER-FITTING

Model is too simple to fit the curve

Training Loop

```
xtrain, ytrain, xval, yval = load_data(npts=500, train_fraction = 0.10)

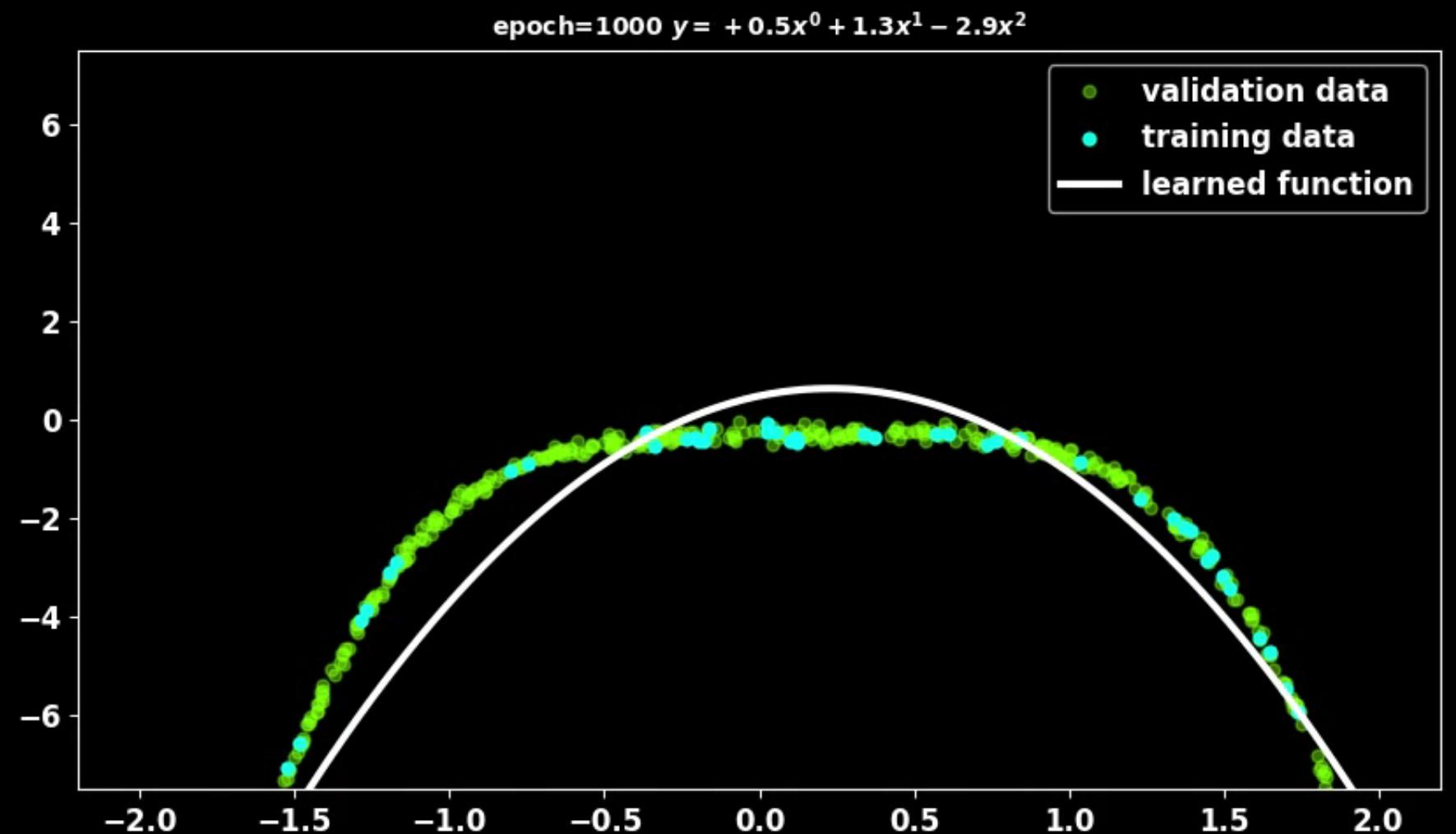
model      = taylor_series order=2)
optimizer = torch.optim.Adam(model.params, lr=1.0e-2)
epochs     = 1000

for i in range(epochs+1):

    # training
    optimizer.zero_grad()
    yhat = model(xtrain)
    loss = (yhat - ytrain).pow(2).mean()
    loss.backward()
    optimizer.step()

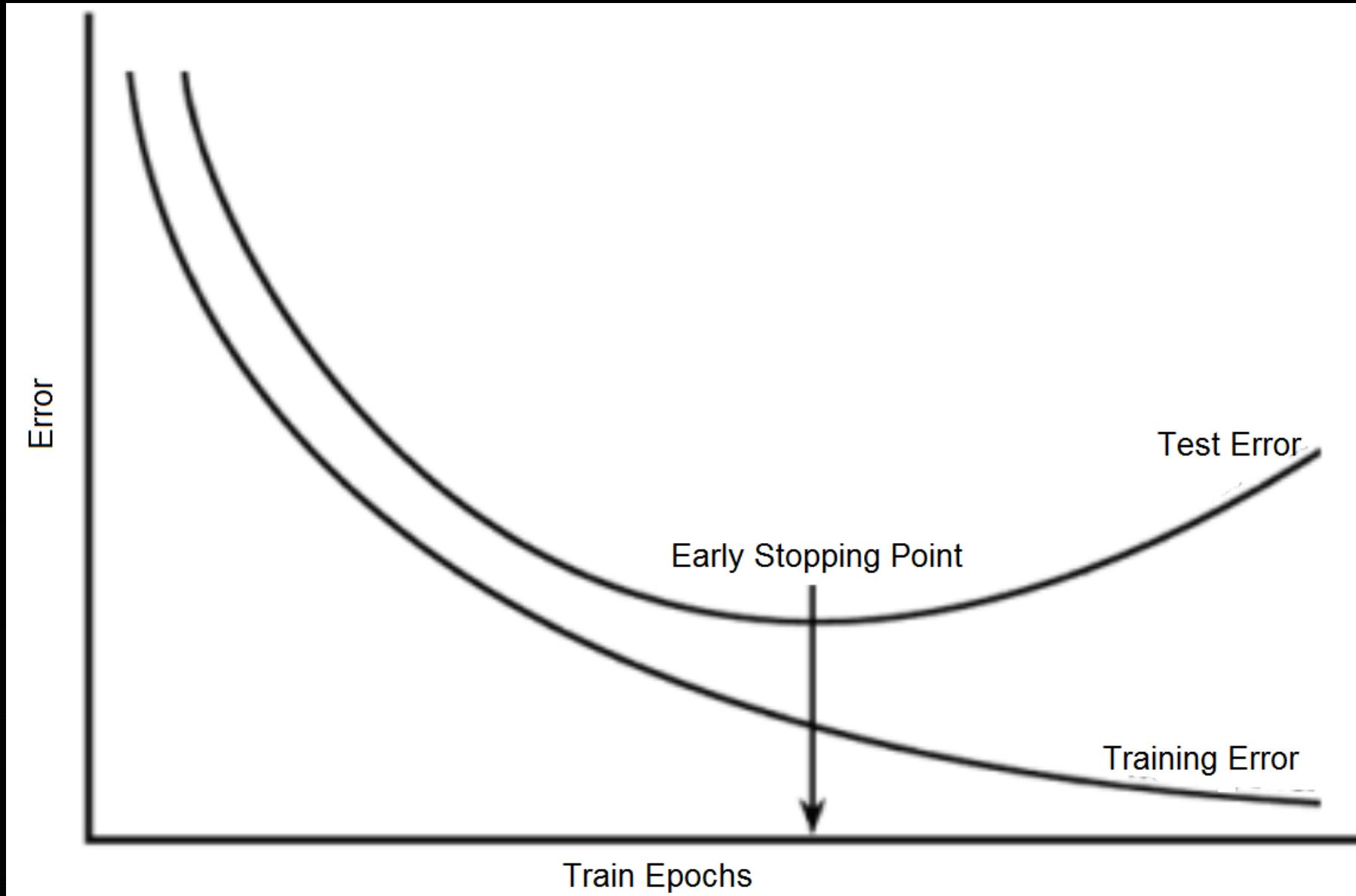
    # validation
    yval_hat = model(xval)
    loss_val = (yval_hat - yval).pow(2).mean()
```

Increase model capacity
Use a different model



REGULARIZATION

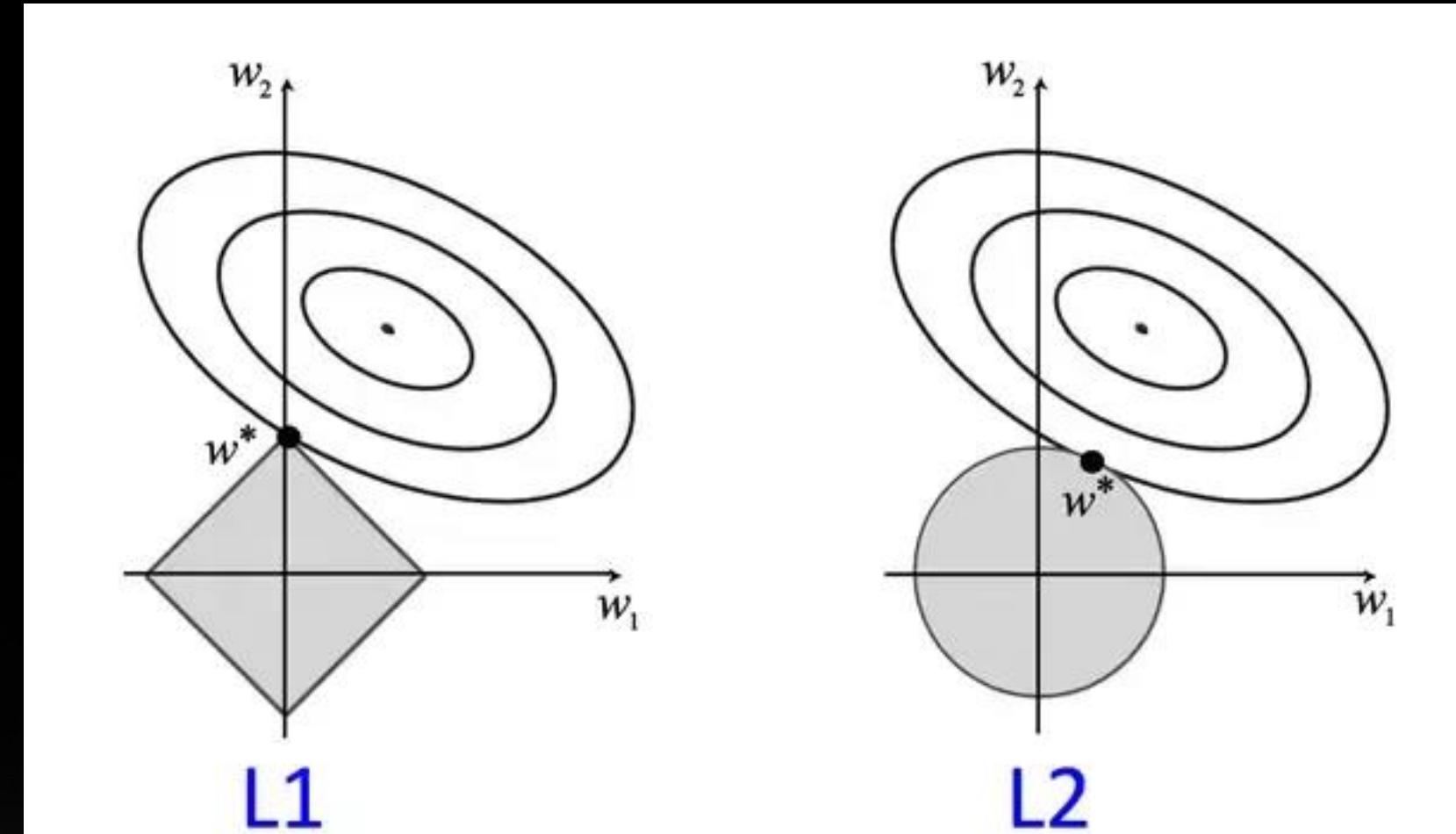
Early Stopping and Layer Regularization



```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

Train Epochs

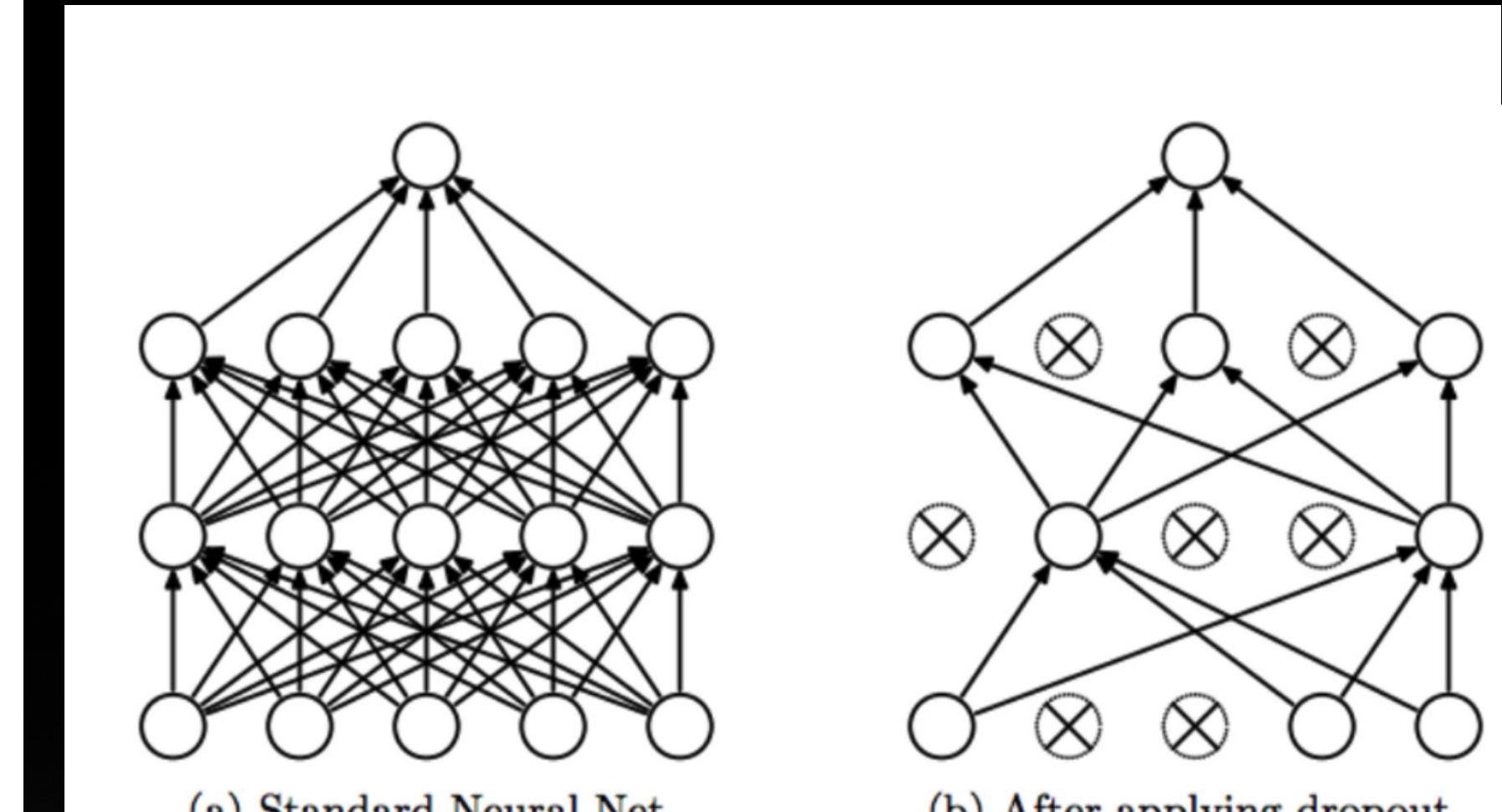
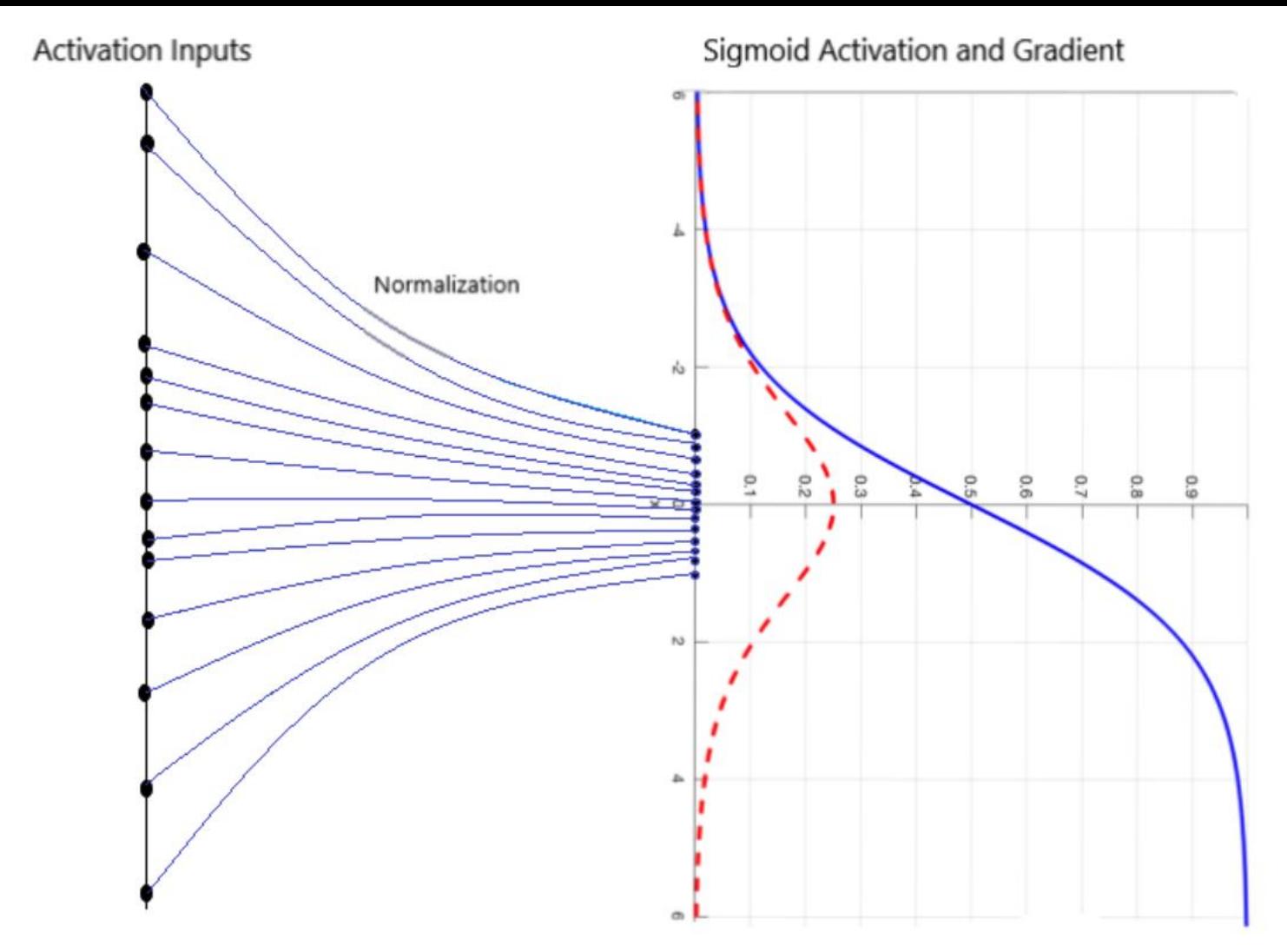


```
layer = tf.keras.layers.Dense(  
    5, input_dim=5,  
    kernel_initializer='ones',  
    kernel_regularizer=tf.keras.regularizers.L1(0.01),  
    activity_regularizer=tf.keras.regularizers.L2(0.01))
```

```
activity_regularizer=tf.keras.regularizers.L1(0.01),  
kernel_regularizer=tf.keras.regularizers.L1(0.01),  
kernel_initializer='ones',  
activity_regularizer='l1')
```

REGULARIZATION

BatchNorm and Dropout



```
model.add(Conv2D(60,3, padding = "same"))
model.add(BatchNormalization())
model.add(Activation("relu"))
```

```
model=keras.models.Sequential()
model.add(keras.layers.Dense(150, activation="relu"))
model.add(keras.layers.Dropout(0.5))
```

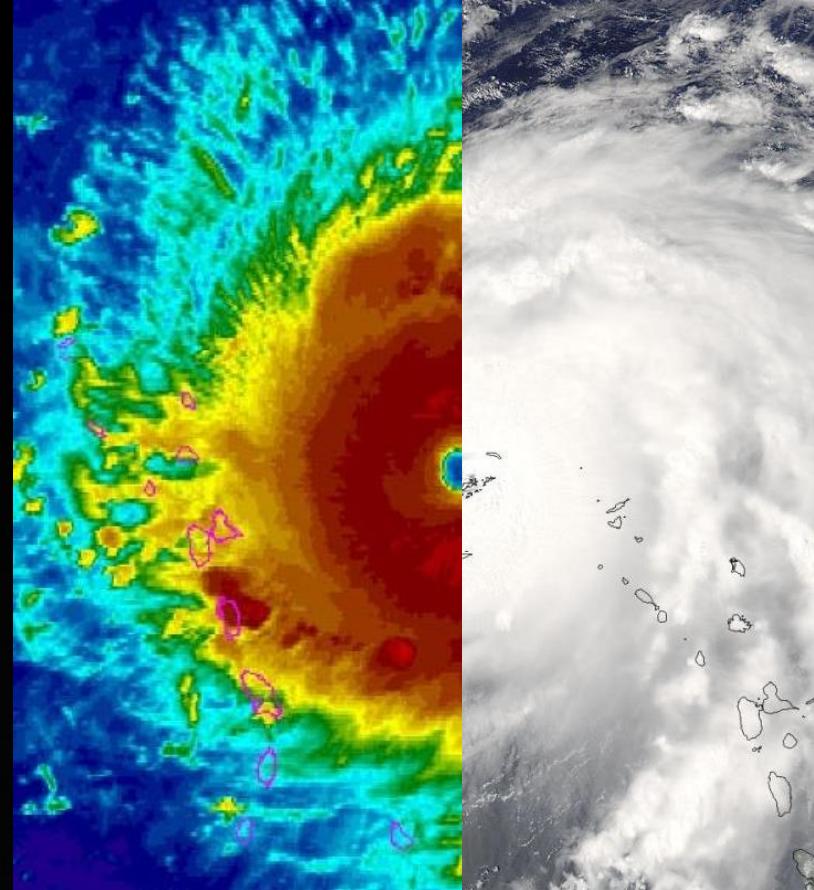
```
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(120, activation="relu"))
model=keras.models.Sequential()
```



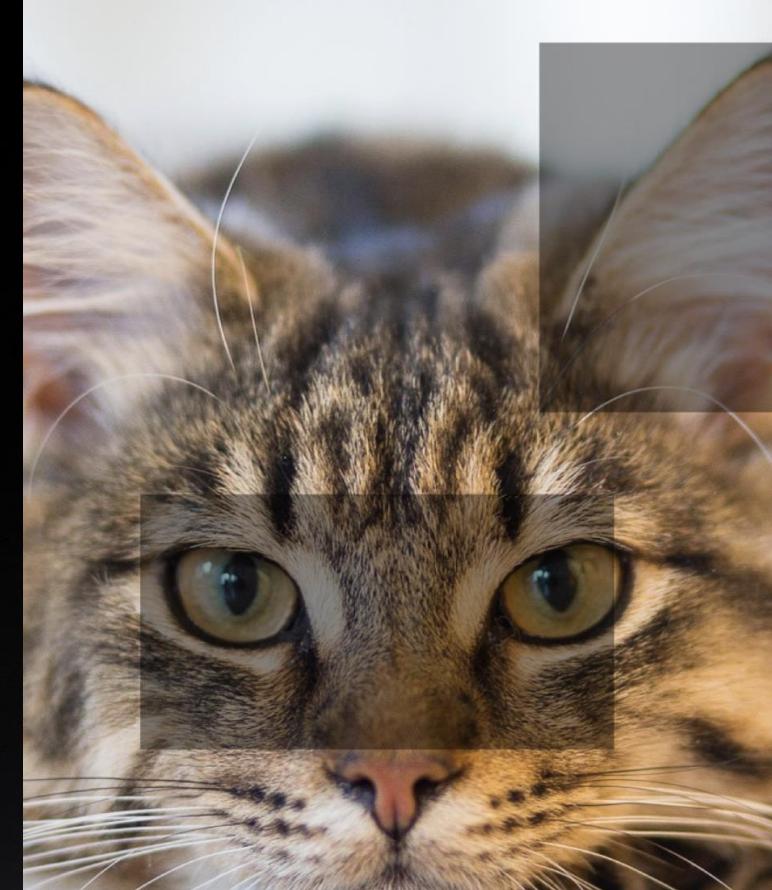
CHALLENGES AND POTENTIAL SOLUTIONS

LABELLING LARGE QUANTITIES OF DATA

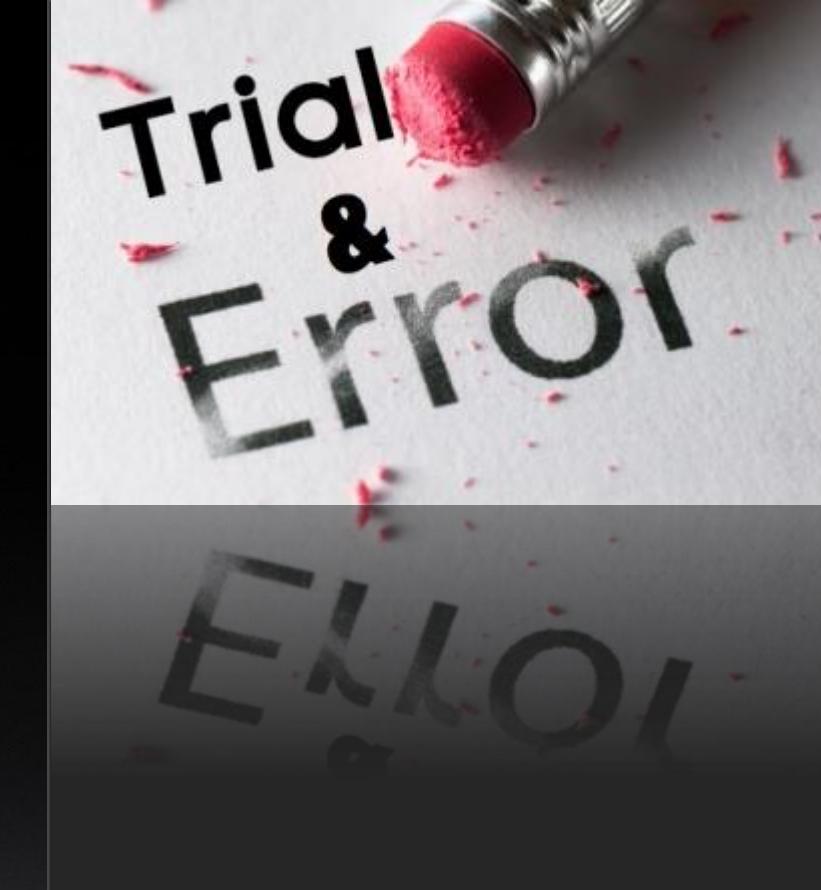
How can we overcome the need for manual labelling?



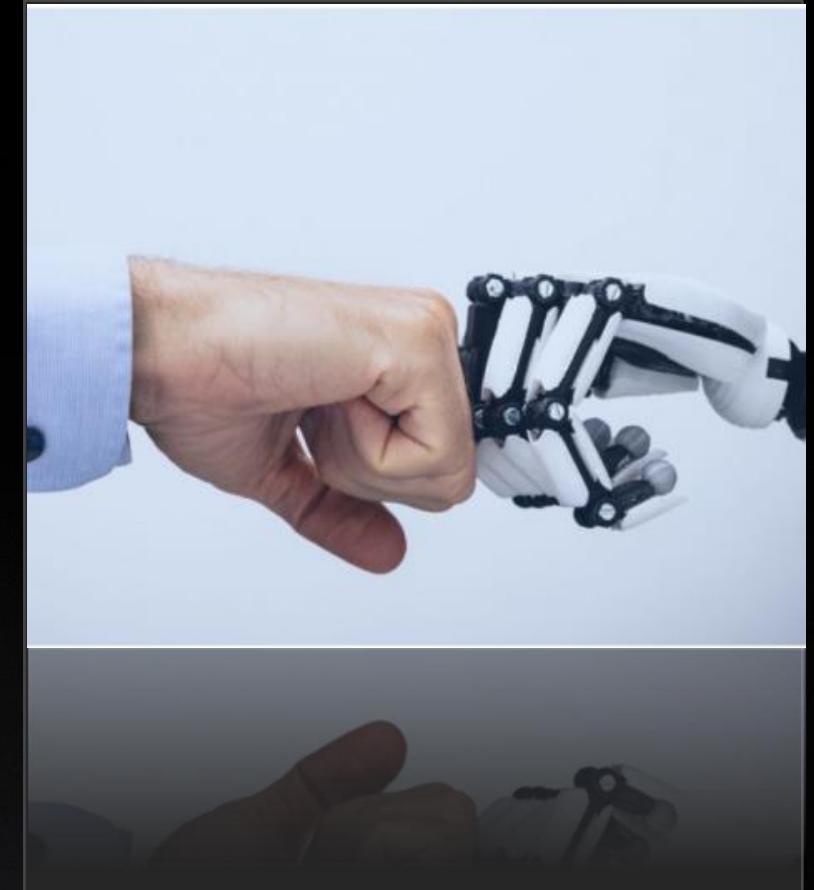
Data Fusion
Using one data source
as the label for another



Self-Supervised Learning
Predicting input B from input A

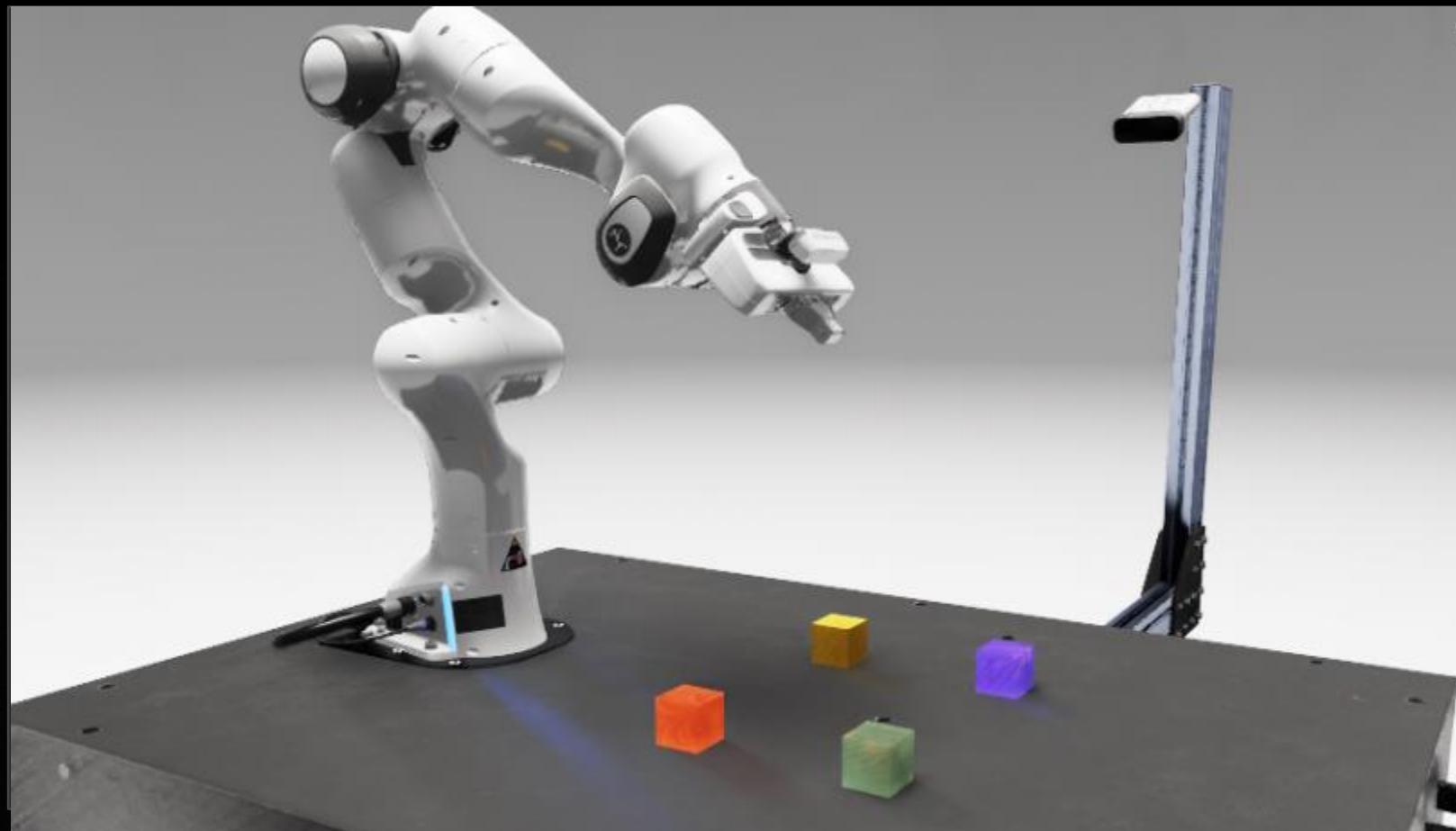


Reinforcement Learning
Obtaining labels directly from the
environment or simulation

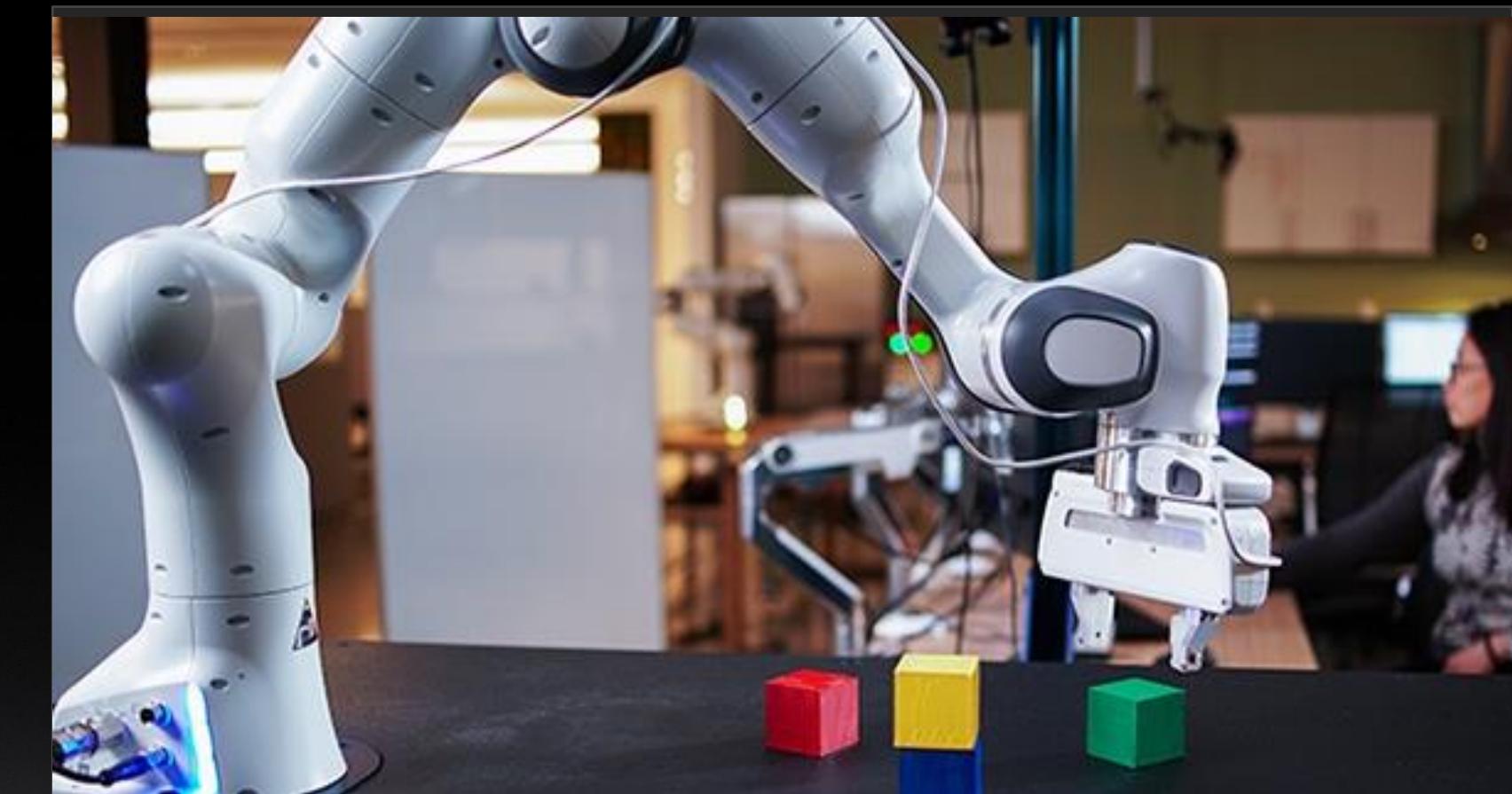


Human-in-the-loop
Using human machine iteration to
make labelling easier

TRANSFER LEARNING: DON'T START FROM SCRATCH



Train on simulated or related data

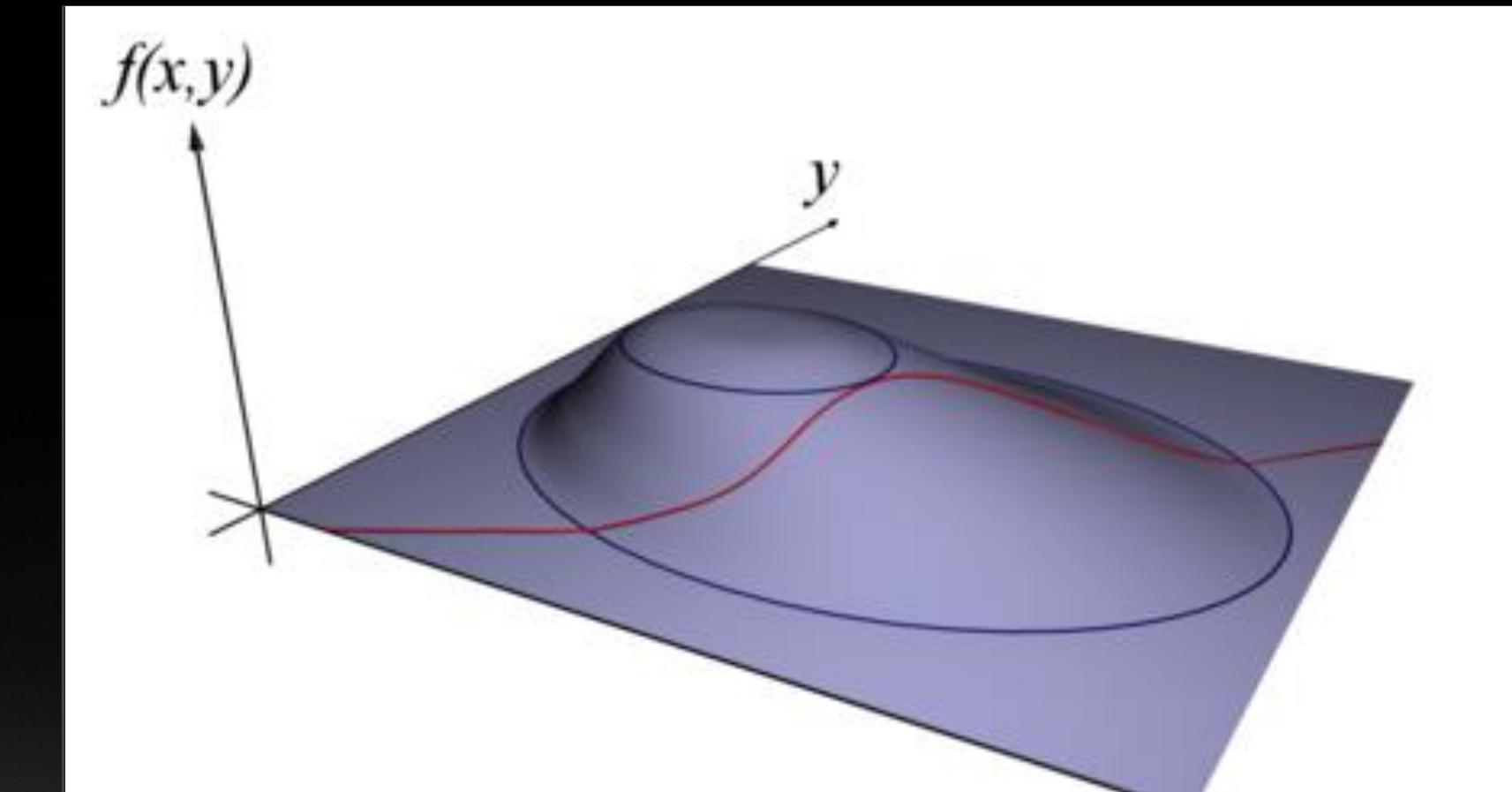


Fine-tune on the real data

ENFORCING PHYSICAL CONSTRAINTS



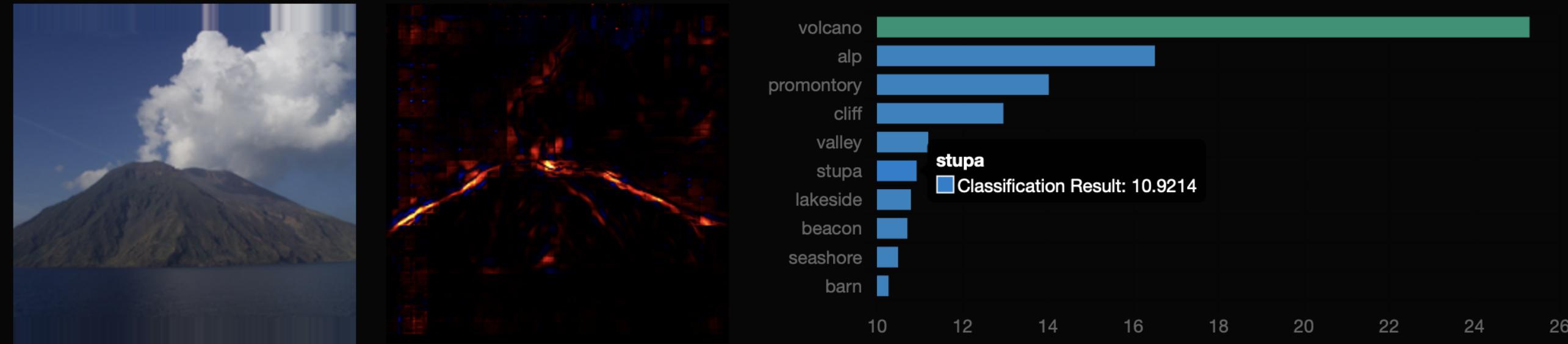
Conservation of Mass, Momentum, Energy, Incompressibility,
Turbulent Energy Spectra, Translational Invariance



Lagrange multipliers (penalization), Hard Constraints,
Projective Methods, Differentiable Programming

INTERPRETABILITY: EXPLAINABLE AI

The image was classified as **volcano**
with a classification score of **25.29**
the true class is **volcano**



The heatmap was rendered for the class **volcano**.

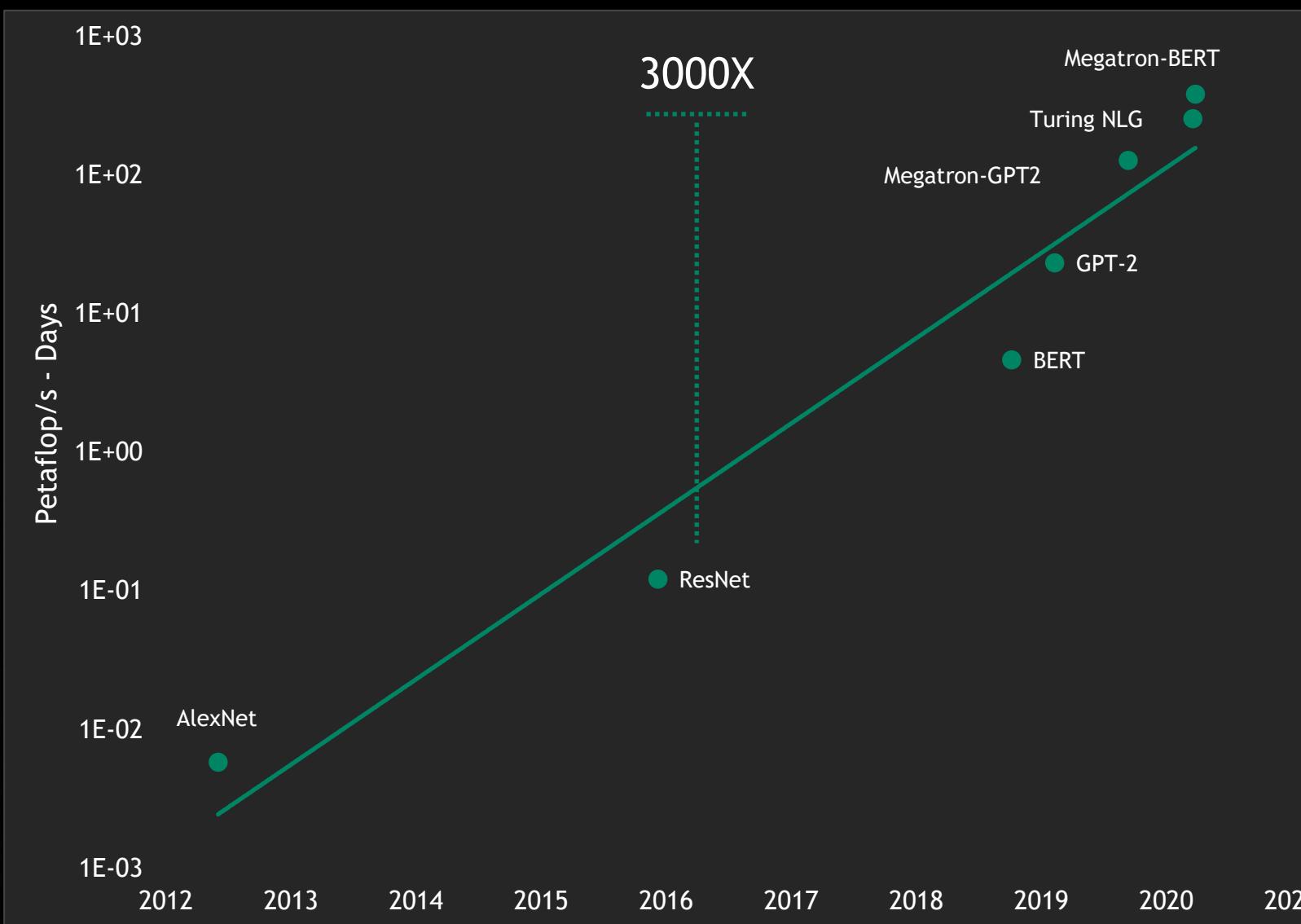
Layer-wise Relevance Propagation



USING YOUR GPU

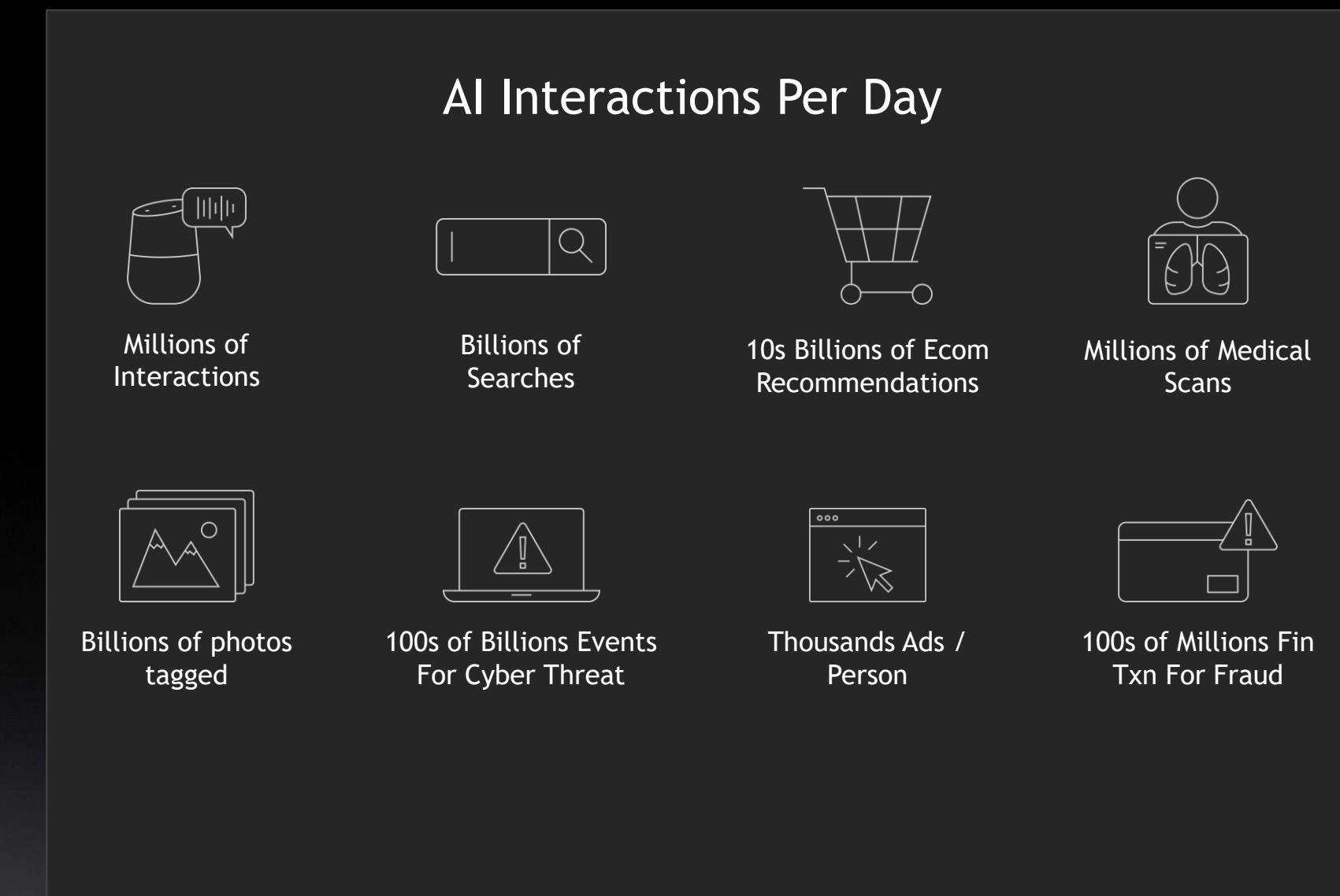
CHALLENGES: ACCELERATING BIG AND SMALL

AI Advances Demand Exponentially Higher Compute



3000X Higher Compute Required to Train Largest Models Since Volta

AI Applications Demand Distributed Pervasive Acceleration



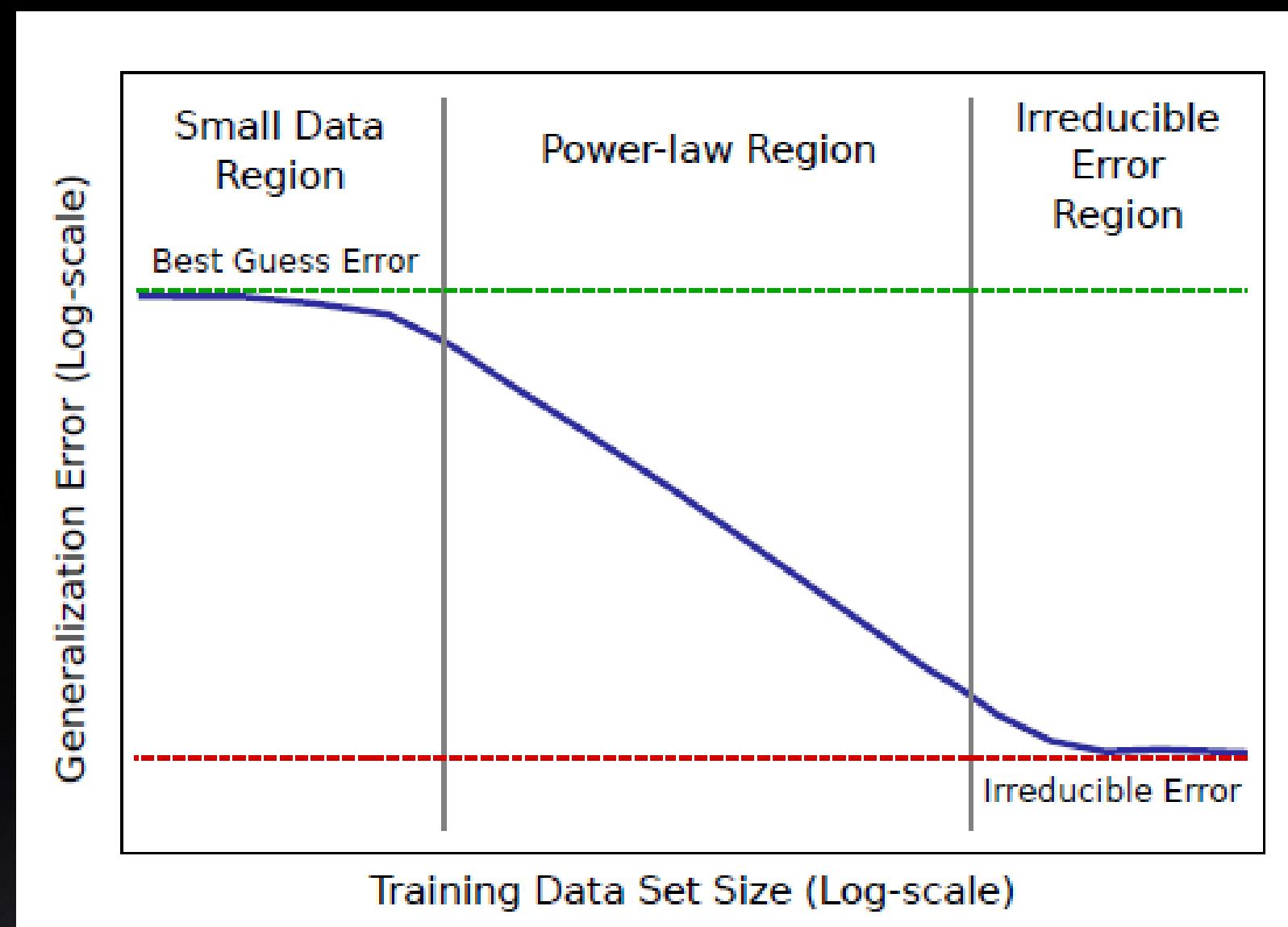
Every AI Powered Interaction Needs Varying Amount of Compute



EMPIRICAL EVIDENCE
BIGGER IS BETTER

EXPLODING DATASETS

Logarithmic relationship between the dataset size and accuracy



THE SCALING LAWS

As you increase the dataset size you must increase the model size

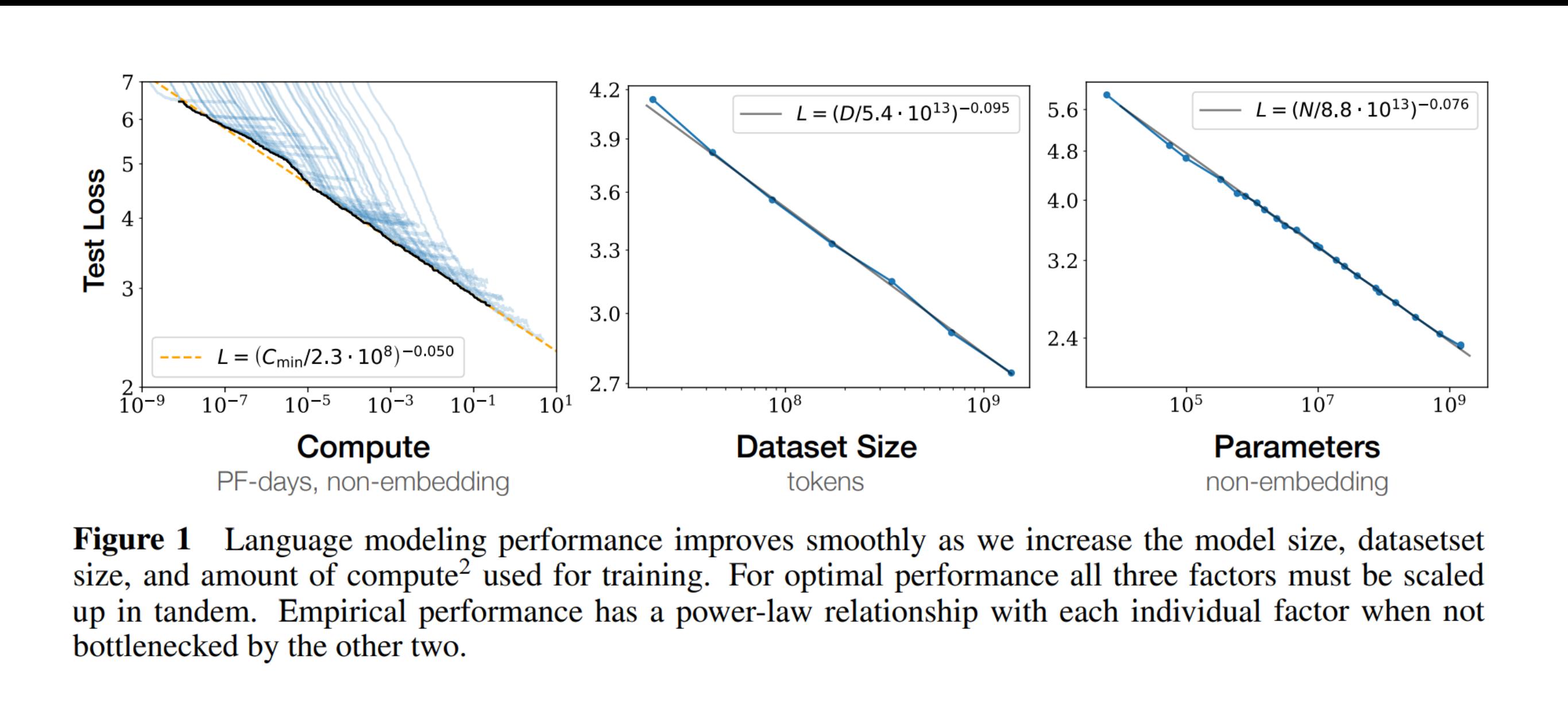
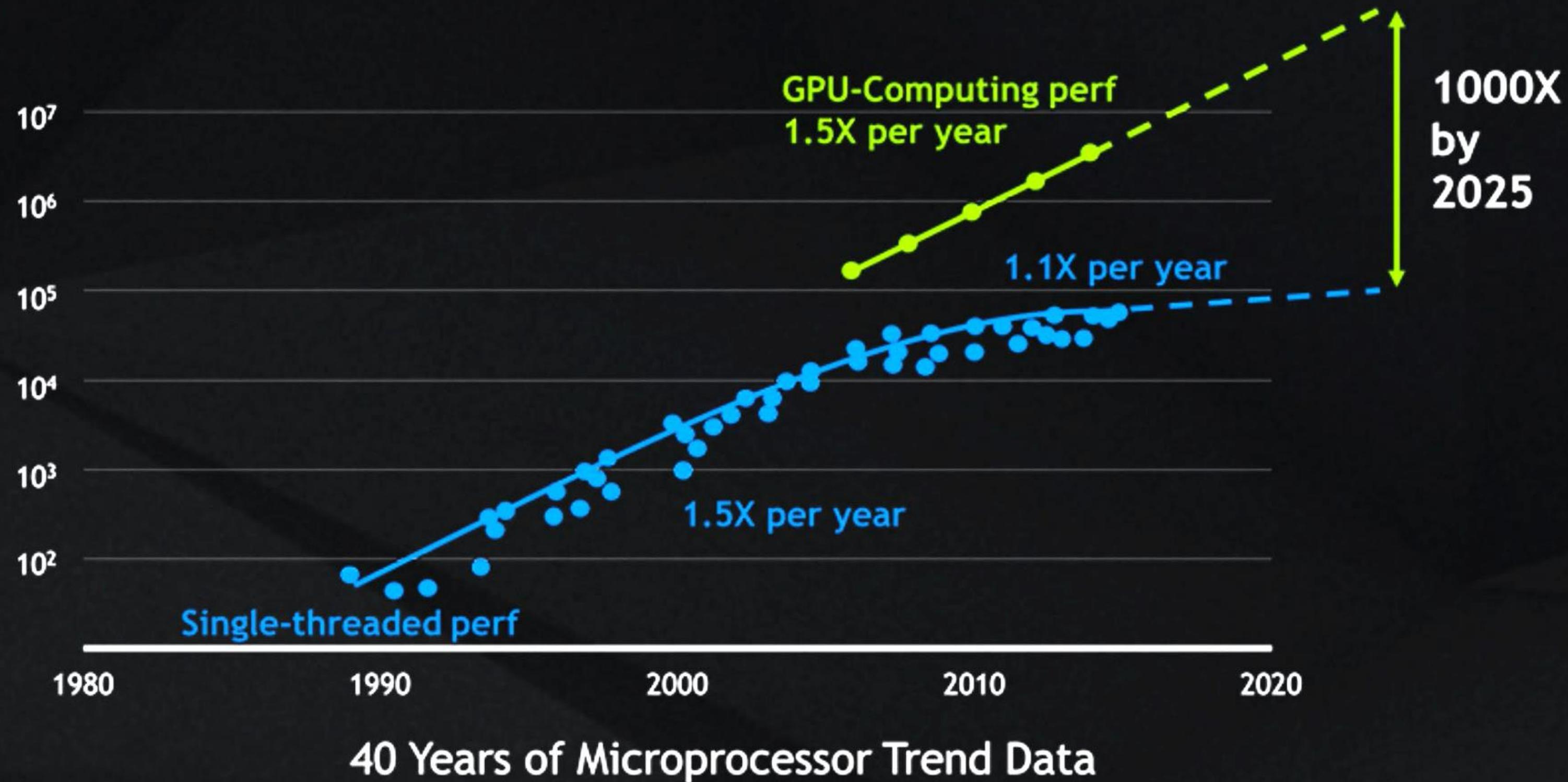


Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

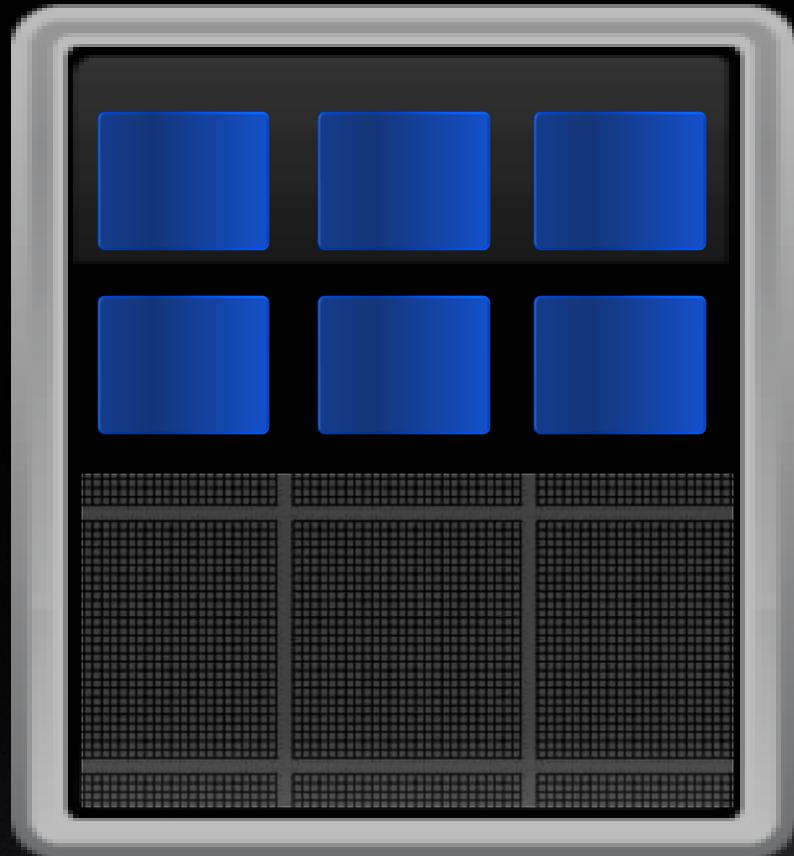
GPUS MAKE MACHINE LEARNING PRACTICAL

Train in a day, or a month?

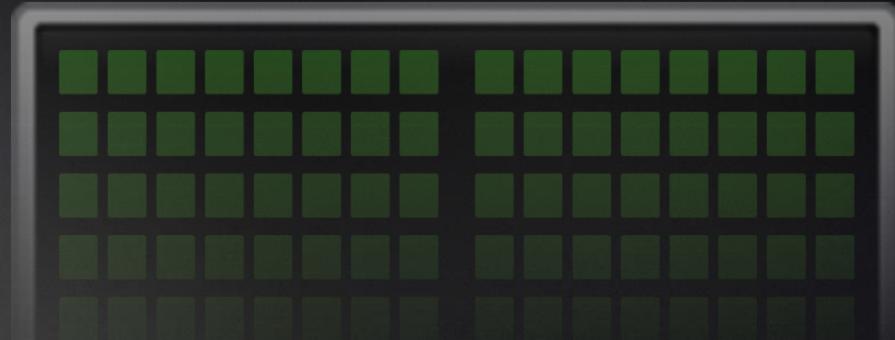
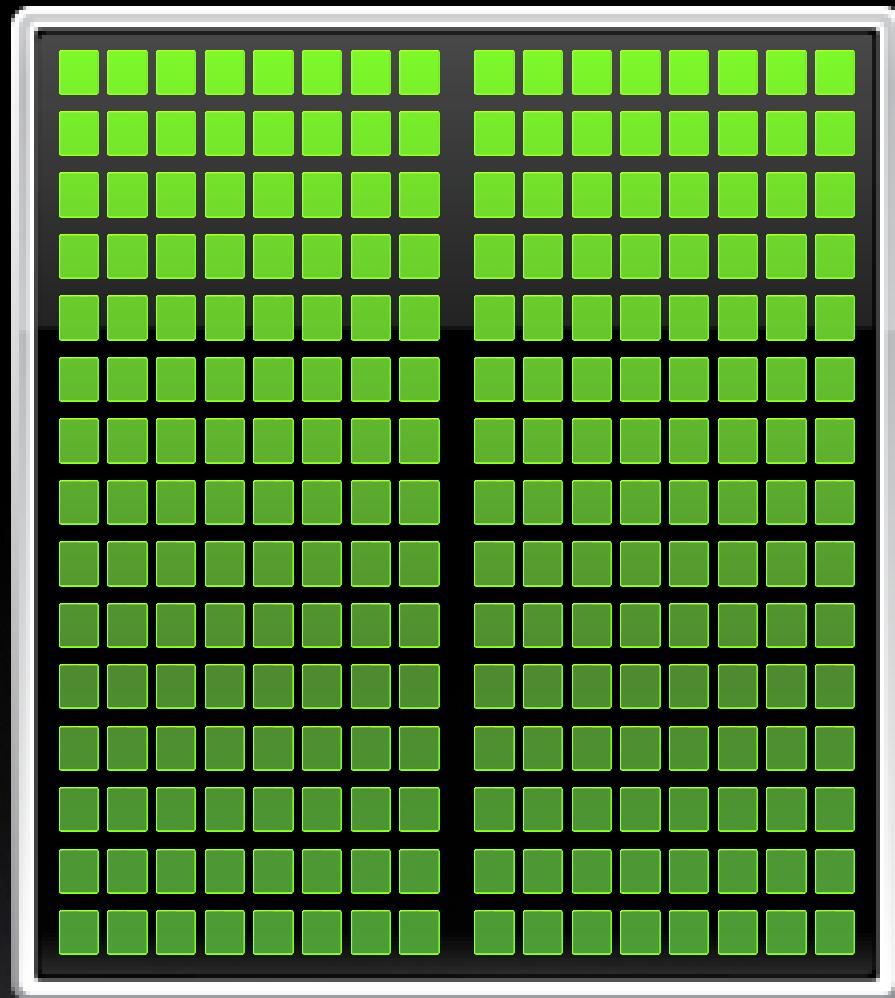


ACCELERATED COMPUTING

CPU
Optimized for
Serial Tasks



GPU Accelerator
Optimized for
Parallel Tasks



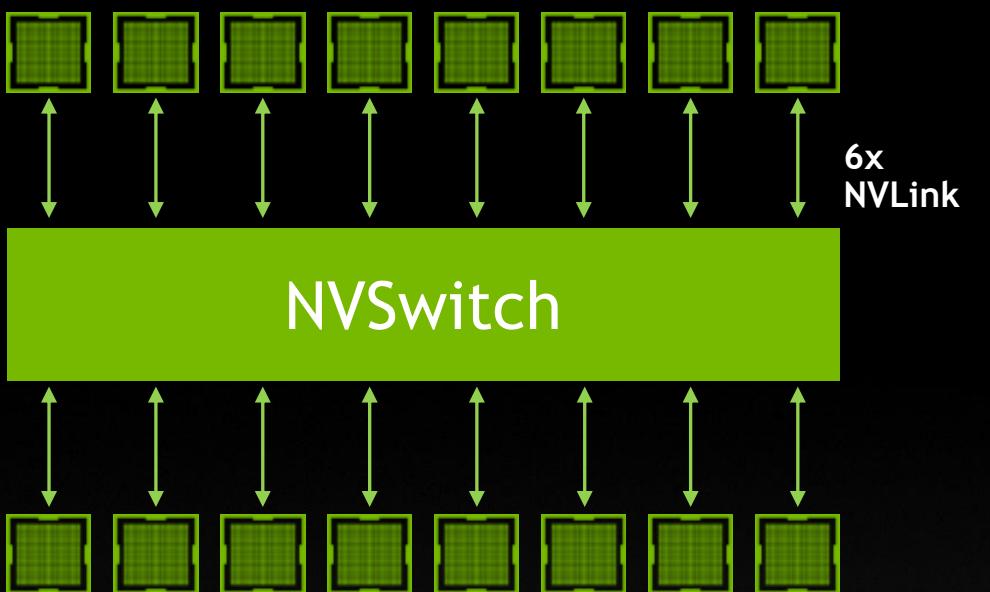
PILLARS OF DATA SCIENCE PERFORMANCE

CUDA Architecture



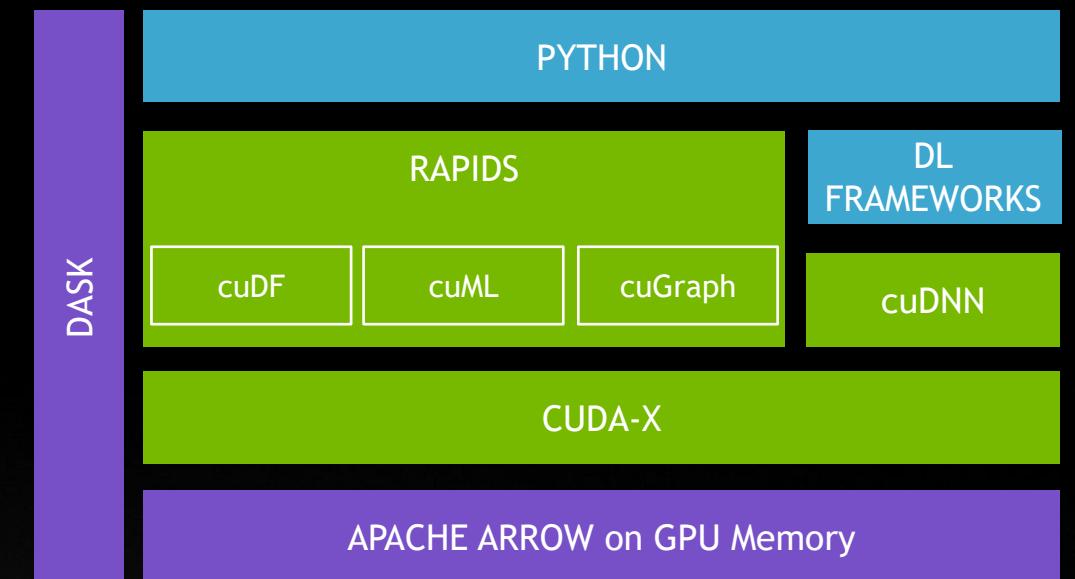
Massively Parallel Processing

NVLink/NVSwitch



High Speed Connecting between
GPUs for Distributed Algorithms

CUDA-X AI



NVIDIA GPU Acceleration Libraries
for Data Science and AI

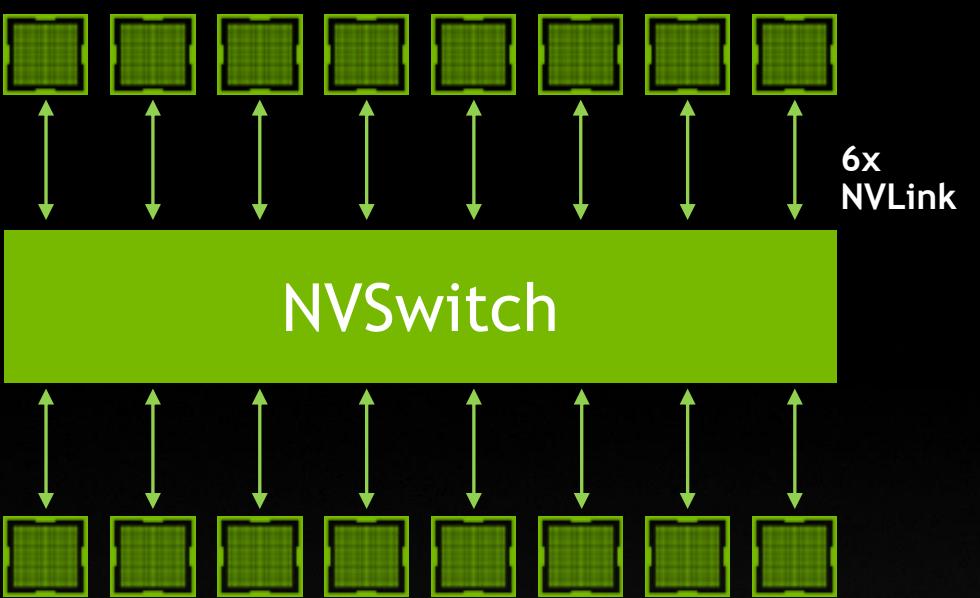
PILLARS OF DATA SCIENCE PERFORMANCE

CUDA Architecture



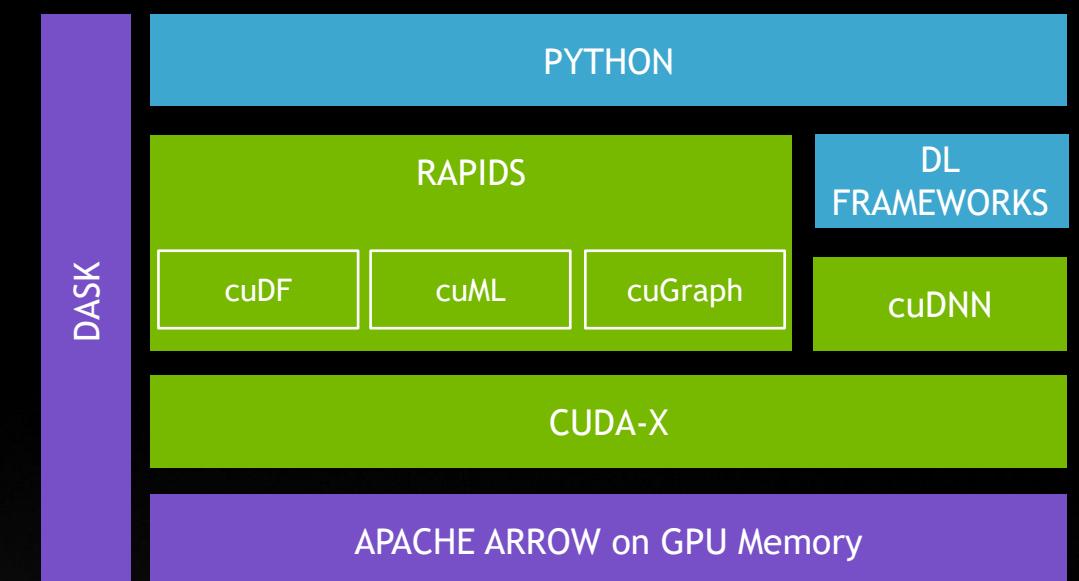
Massively Parallel Processing

NVLink/NVSwitch



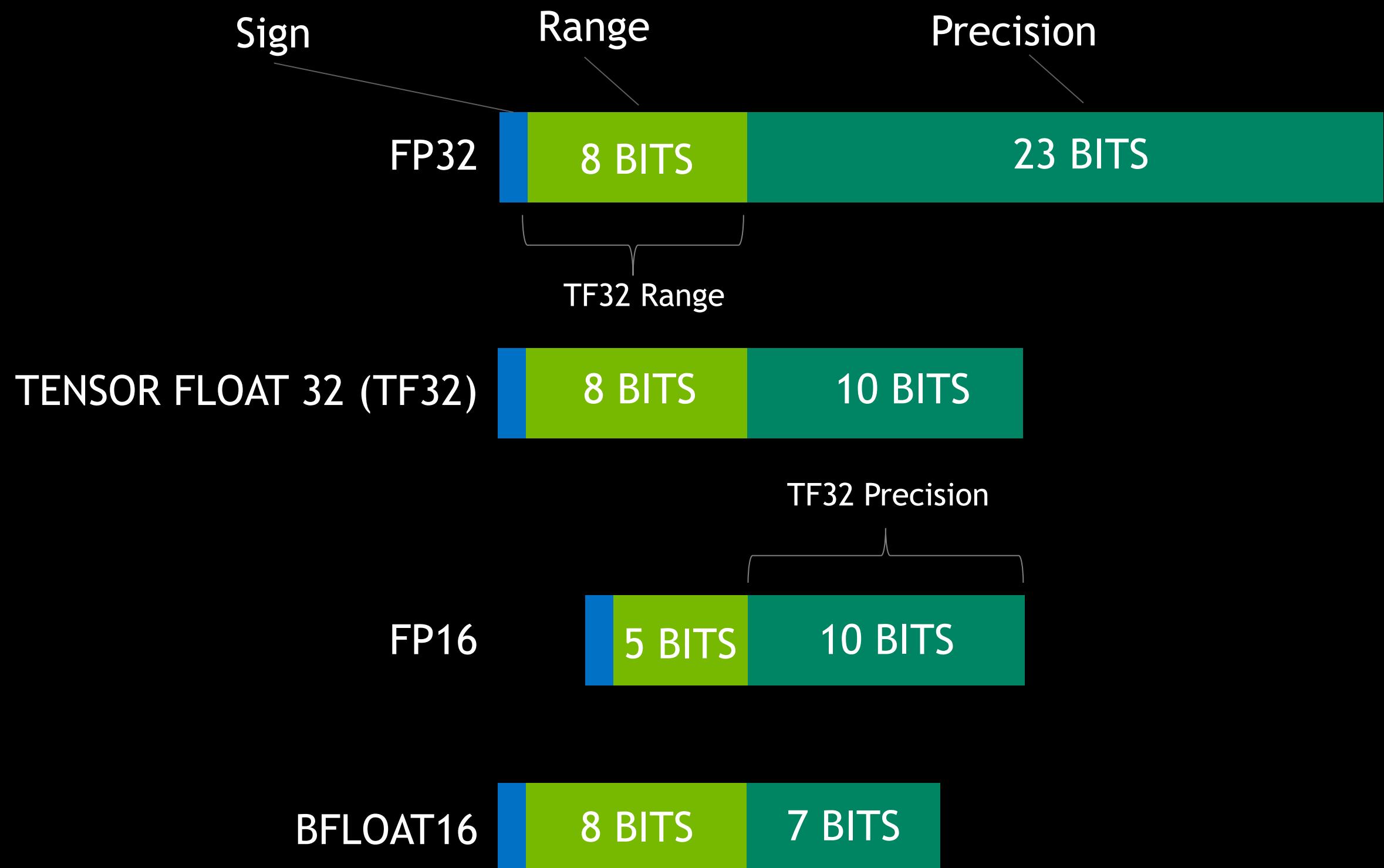
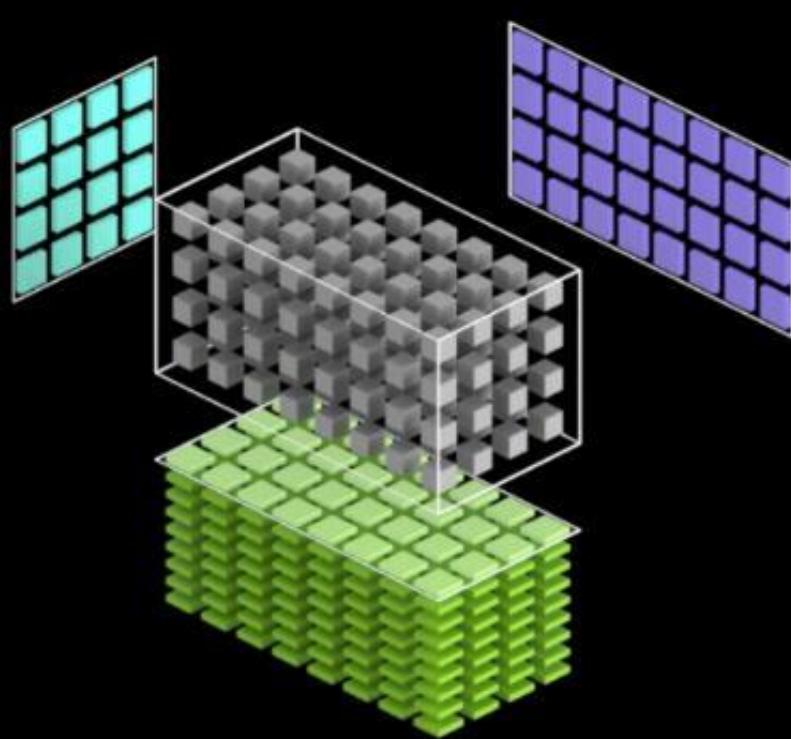
High Speed Connecting between
GPUs for Distributed Algorithms

CUDA-X AI



NVIDIA GPU Acceleration Libraries
for Data Science and AI

TENSOR CORES FOR DIFFERENT NEEDS



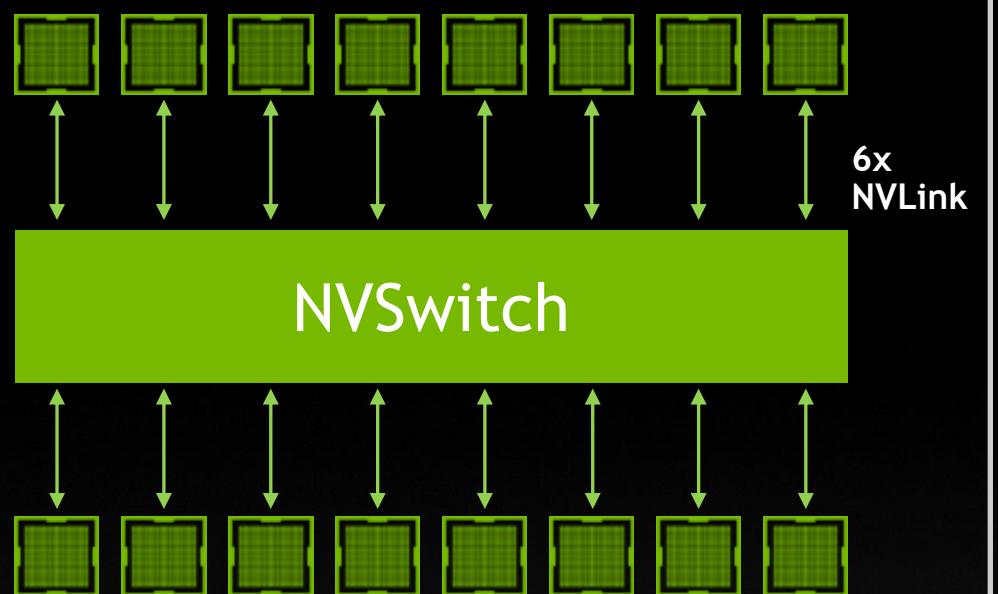
- No Code Change Speed-up for Training

PILLARS OF DATA SCIENCE PERFORMANCE

CUDA Architecture

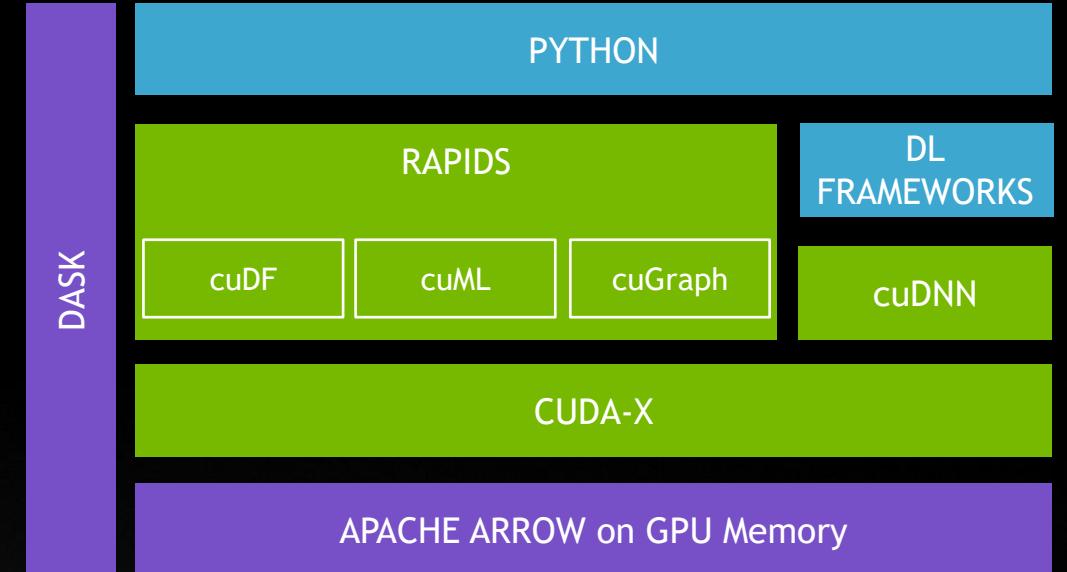


NVLink/NVSwitch



High Speed Connecting between
GPUs for Distributed Algorithms

CUDA-X AI

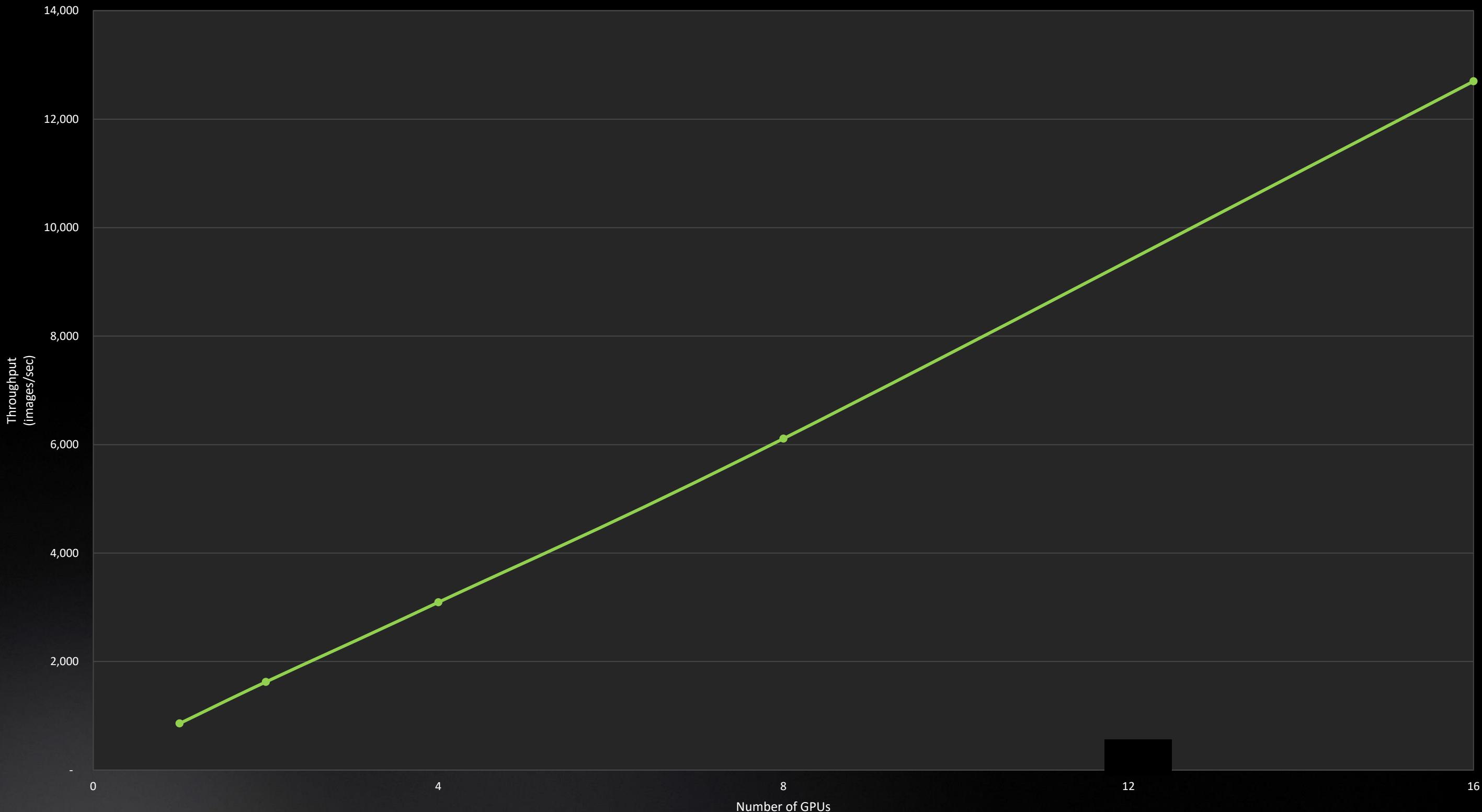


Massively Parallel Processing

NVIDIA GPU Acceleration Libraries
for Data Science and AI

Deep Learning GPU Acceleration on PyTorch

Training on ResNet-50 v1.5 (Image Classification)



Server: Dual Xeon Platinum 8168@3.6GHz with GPU configs as shown. Framework: PyTorch v1.5.0a0+8f84ded;
Mixed Precision; CUDA 10.2.89; NCCL 2.6.3; cuDNN 7.6.5.32; cuBLAS 10.2.2.89; DALI 0.19.0; NVIDIA Driver: 440.33.01; Batch size: 512, 256 for 16x V100



PyTorch
Deep Learning Training

PyTorch is a deep learning framework that puts Python first.

VERSION
1.5.0a0+8f84ded

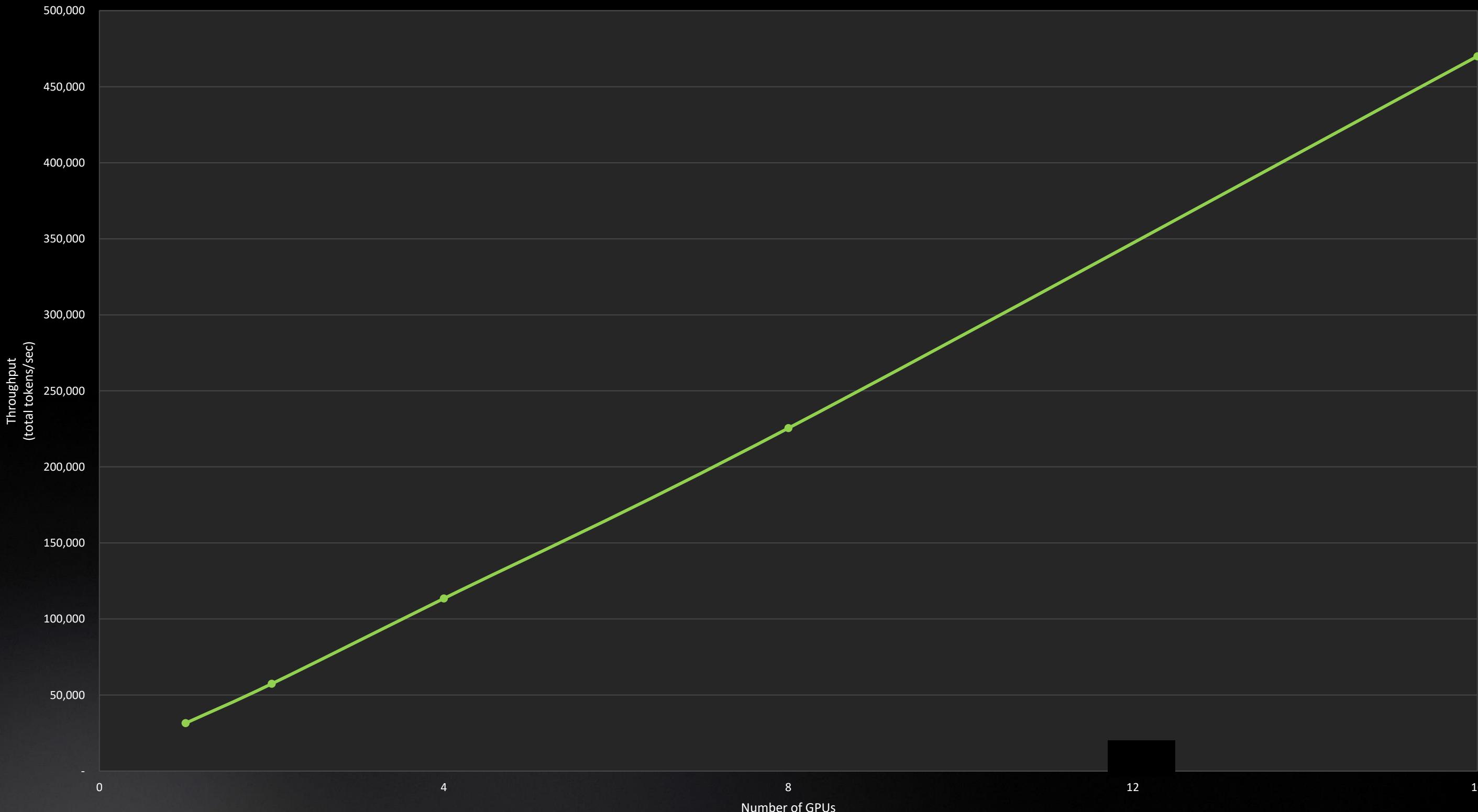
ACCELERATED FEATURES
Full framework accelerated

SCALABILITY
Multi-GPU, multi-node

More Information
www.pytorch.org
[PyTorch on NGC](#)

Deep Learning GPU Acceleration on PyTorch

Training on Transformer XL (Neural Machine Translation)



Server: Dual Xeon Platinum 8168@3.6GHz with GPU configs as shown. Framework: PyTorch v1.5.0a0+8f84ded;
Mixed Precision; CUDA 10.2.89; NCCL 2.6.3; cuDNN 7.6.5.32; cuBLAS 10.2.2.89; DALI 0.19.0; NVIDIA Driver: 440.33.01; Batch size: 32



PyTorch
Deep Learning Training

PyTorch is a deep learning framework that puts Python first.

VERSION
1.5.0a0+8f84ded

ACCELERATED FEATURES
Full framework accelerated

SCALABILITY
Multi-GPU, multi-node

More Information
www.pytorch.org
[PyTorch on NGC](#)

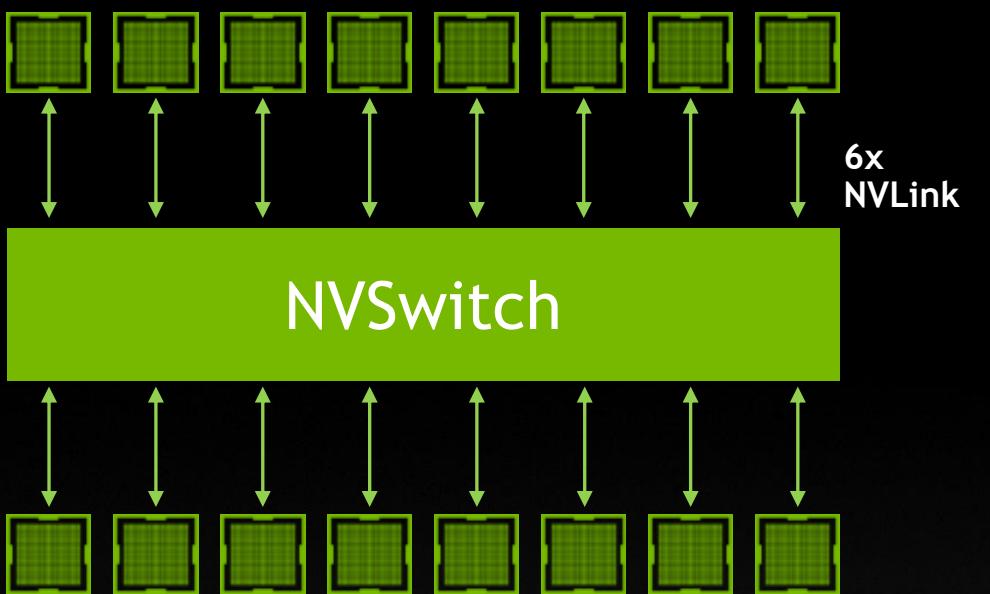
PILLARS OF DATA SCIENCE PERFORMANCE

CUDA Architecture



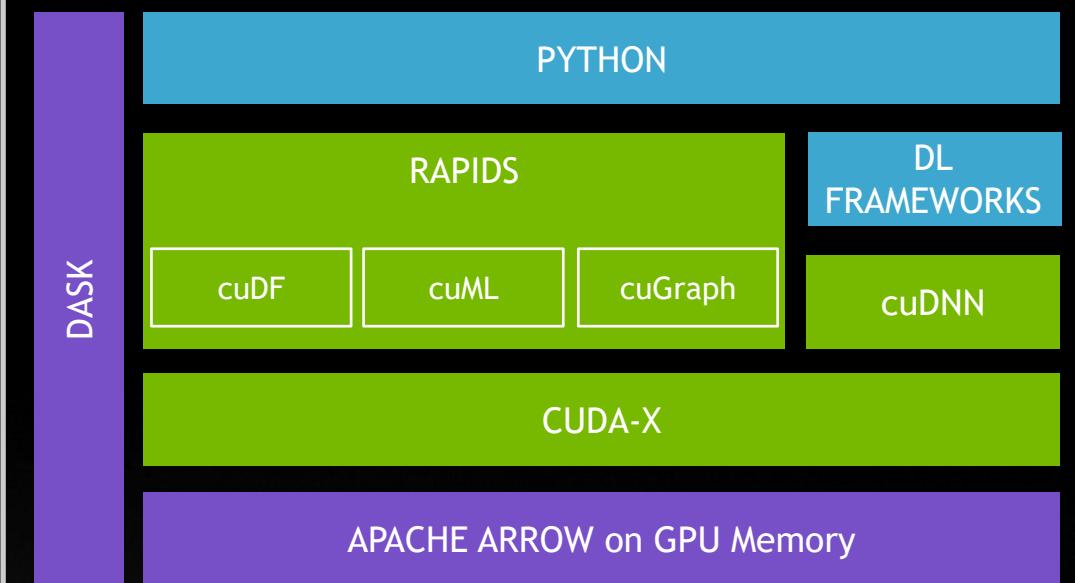
Massively Parallel Processing

NVLink/NVSwitch



High Speed Connecting between
GPUs for Distributed Algorithms

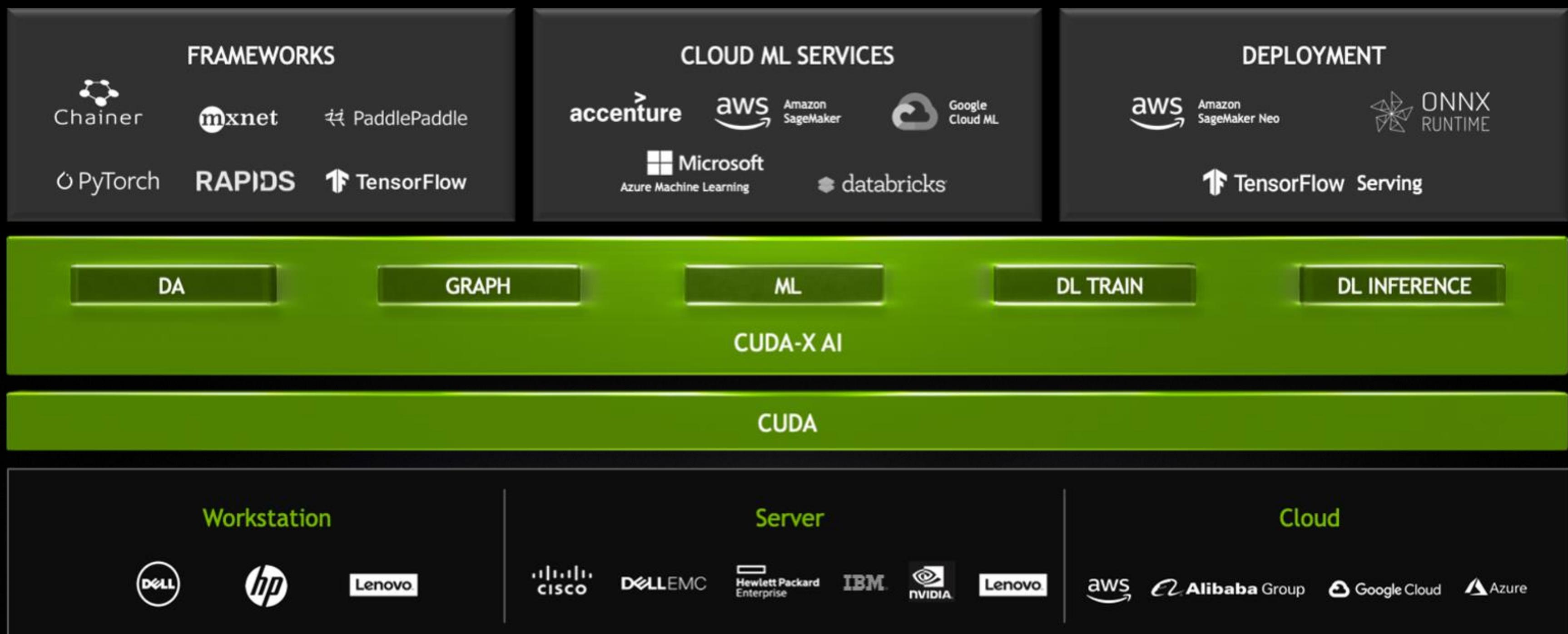
CUDA-X AI



NVIDIA GPU Acceleration Libraries
for Data Science and AI

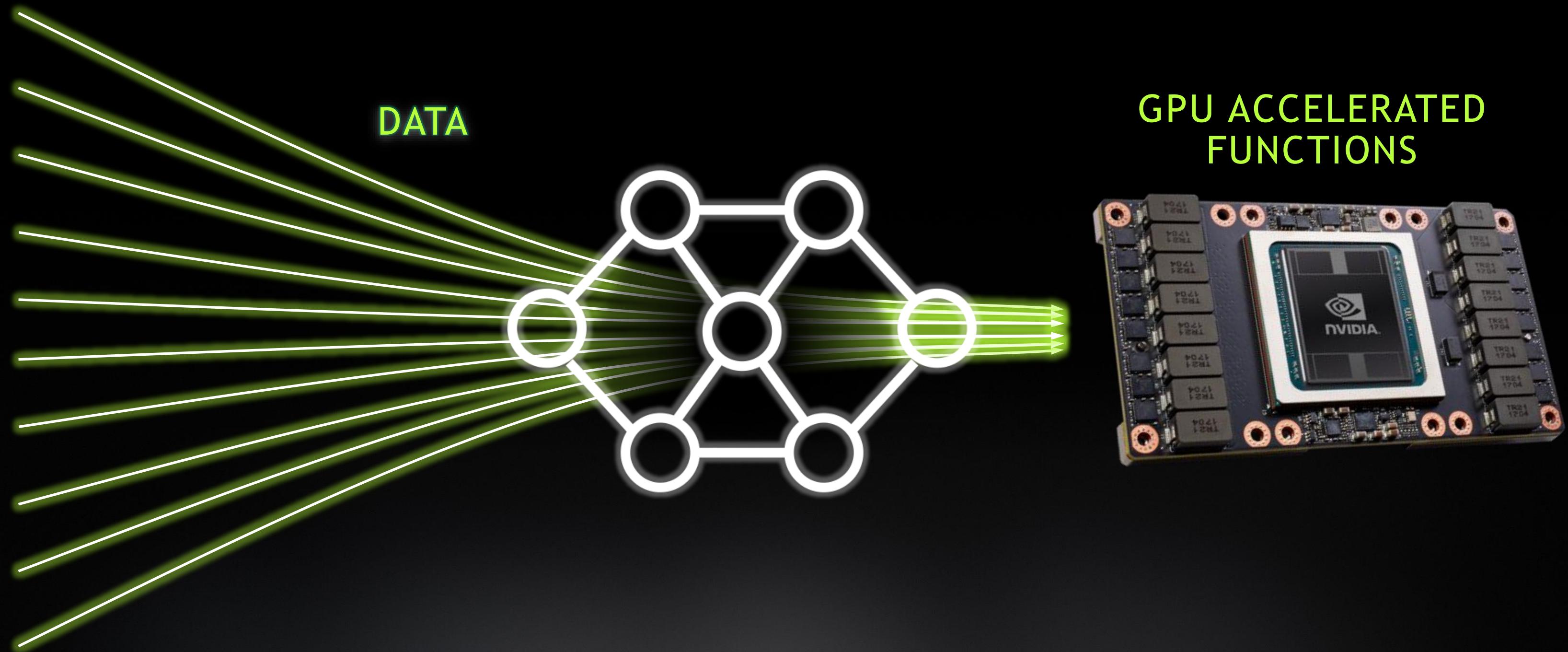
CUDA-X AI TRANSFORMS DATA SCIENCE

From Data Science to NVIDIA Accelerated Data Science with CUDA-X AI



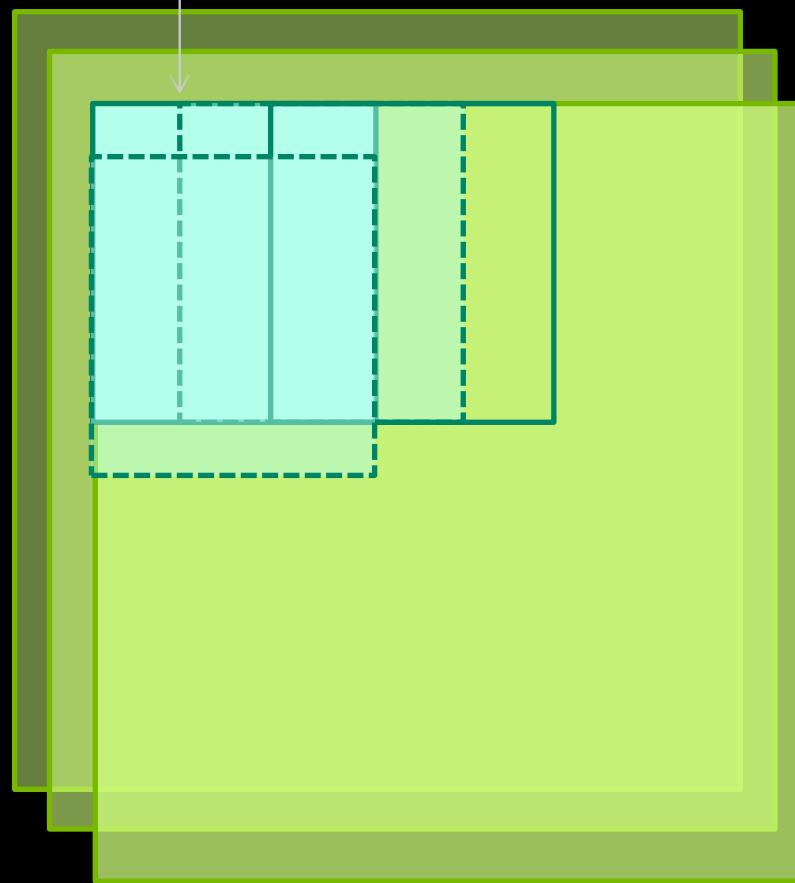
LEARNED FUNCTIONS ARE GPU ACCELERATED

Next level software. No porting required.

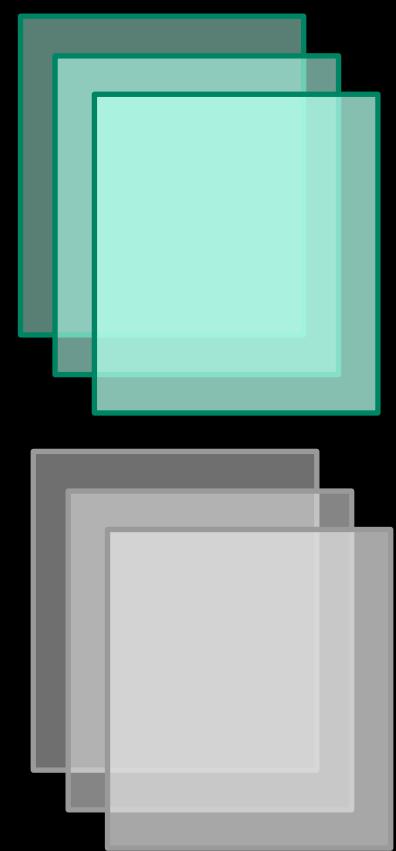


CONVOLUTION OPERATION

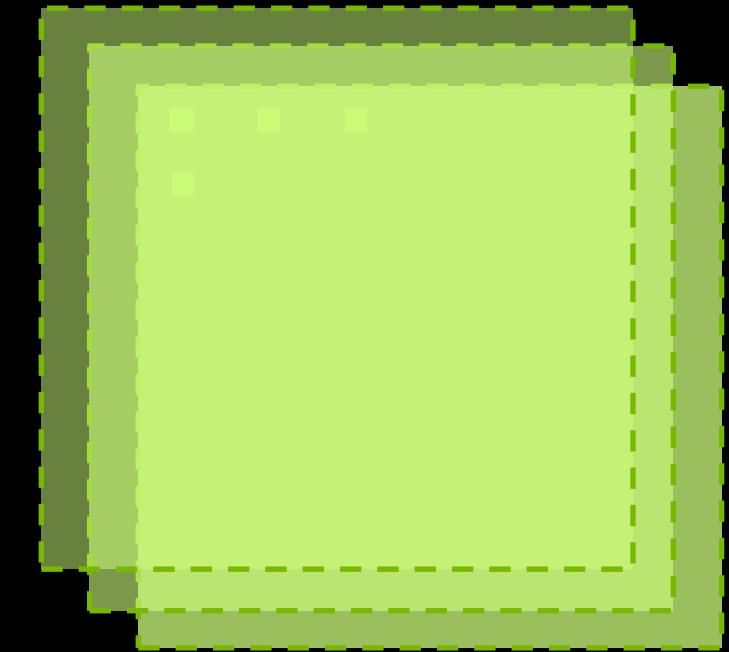
Pointwise multiply and sum, scalar output



Input Image

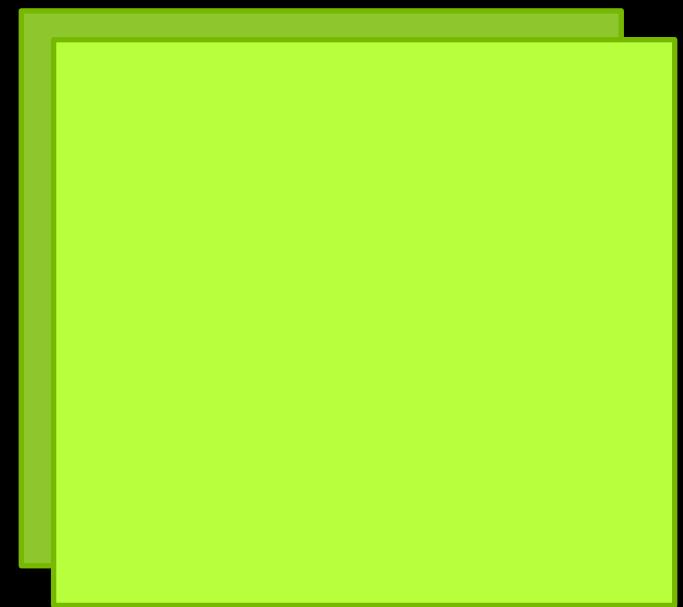


Input Filter



Intermediate output

$$int[c, p, q] = \sum_{i, j \in filter} Im[c][istart + i, jstart + j] . Filt[c][i, j]$$
$$output[p, q] = \sum_c int[c, p, q]$$



Final Output

Why do it once if you can do it n times ? Batch the whole thing.

VERIFYING YOUR GPU

TensorFlow

```
from tensorflow.python.client import device_lib  
print('GPU' in str(device_lib.list_local_devices()))
```

True

PyTorch

```
from torch import cuda  
print(f"cuda.is_available: {cuda.is_available()}")  
print(f"cuda.device_count(): {cuda.device_count()}")  
print(cuda.get_device_name(cuda.current_device()))
```

```
cuda.is_available: True  
cuda.device_count(): 2  
Quadro GV100
```

Keras

```
from keras import backend as K  
K.tensorflow_backend._get_available_gpus()
```

Using TensorFlow backend.

```
['/job:localhost/replica:0/task:0/device:GPU:0',  
 '/job:localhost/replica:0/task:0/device:GPU:1']
```

Julia

```
using CUDADrv, Printf  
@printf("device 0 = %s \n" , CUDADrv.name(CuDevice(0)))  
@printf("device 1 = %s \n" , CUDADrv.name(CuDevice(1)))
```

```
device 0 = Quadro GV100  
device 1 = Quadro GV100
```

TRAINING ON A SINGLE GPU

Keras

Automatically uses GPU if available

Julia

```
use_gpu = true
to_device(x) = use_gpu ? gpu(x) : x
x      = to_device.(x)
y      = to_device.(y)
model = to_device(model)
```

PyTorch

```
device = torch.device("cuda:1" if torch.cuda.is_available() else "cpu")

# move data onto the GPU
model = model.to(device)
X,Y   = X.to(device), Y.to(device)

# Alocate on the GPU directly
dtype = torch.cuda.FloatTensor
X = torch.zeros(100).type(dtype)

# set default location for tensors
torch.set_default_tensor_type('torch.cuda.FloatTensor')
```

NVIDIA-SMI

System Management Interface

```
x - Terminal File Edit View Search Terminal Help
Every 0.5s: nvidia-smi                                         Tue Sep 17 10:02:34 2019

Tue Sep 17 10:02:34 2019
+-----+
| NVIDIA-SMI 396.82          Driver Version: 396.82 |
|-----+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|-----+
| 0  Quadro GV100        Off  | 00000000:04:00.0 On   |                   Off |
| 49%   61C     P2    51W / 250W | 1451MiB / 32500MiB | 3%       Default |
+-----+
| 1  Quadro GV100        Off  | 00000000:84:00.0 Off  |                   Off |
| 41%   58C     P2   116W / 250W | 1701MiB / 32508MiB | 41%       Default |
+-----+
| Processes:                               GPU Memory |
| GPU  PID  Type  Process name           Usage  |
|-----+
| 0    1661  G   /usr/lib/xorg/Xorg            328MiB |
| 0    2381  G   compiz                      70MiB  |
| 0    8598  G   ...quest-channel-token=8542253777236559196 102MiB |
| 0    10245 C   /home/dhall/anaconda3/envs/tf/bin/python  946MiB |
| 1    10245 C   /home/dhall/anaconda3/envs/tf/bin/python 1690MiB |
+-----+
```

Memory Utilizaiton

Processor Utilization

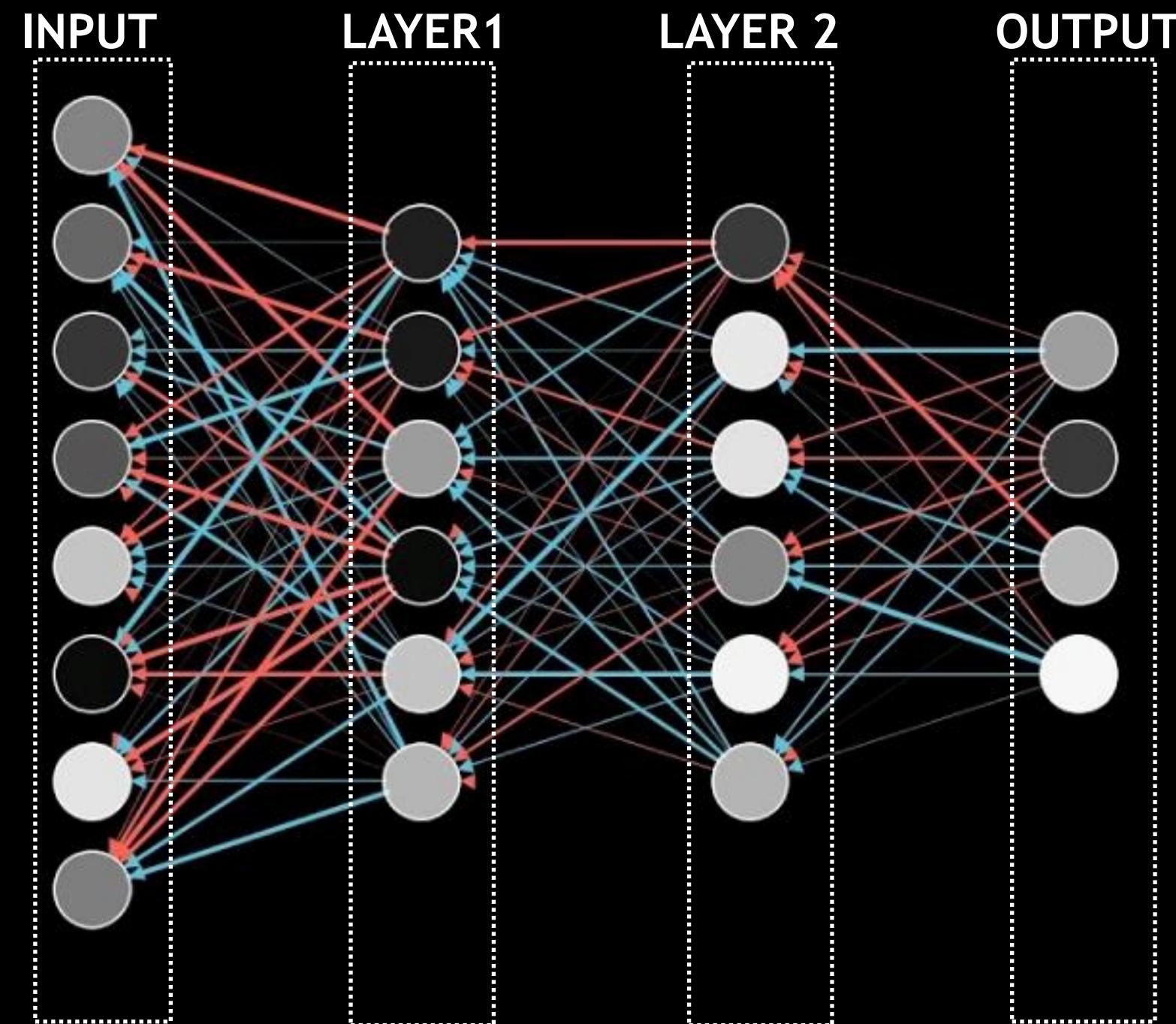
Process Info



FULLY CONNECTED NETWORKS (MULTI-LAYER PERCEPTRONS)

FULLY CONNECTED NETWORKS

A given neuron is connected to every neuron in the previous layer

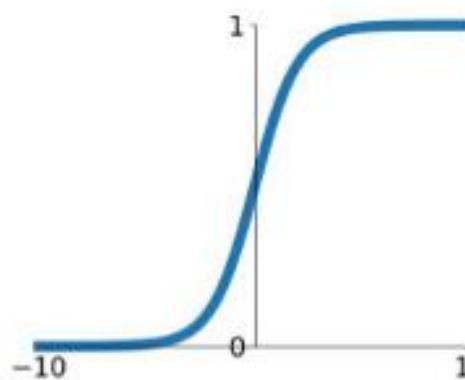


ACTIVATION FUNCTIONS

Many to choose from. But most use ReLU or LeakyReLU

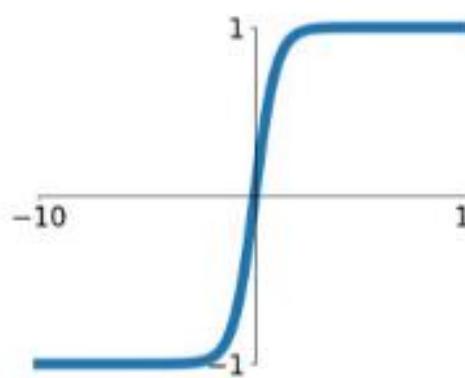
Sigmoid

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



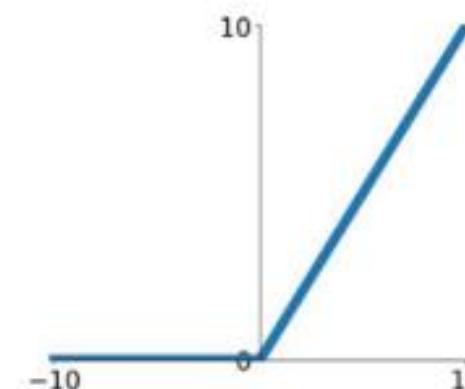
tanh

$$\tanh(x)$$



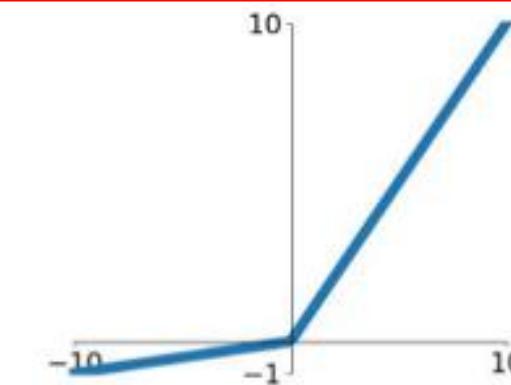
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

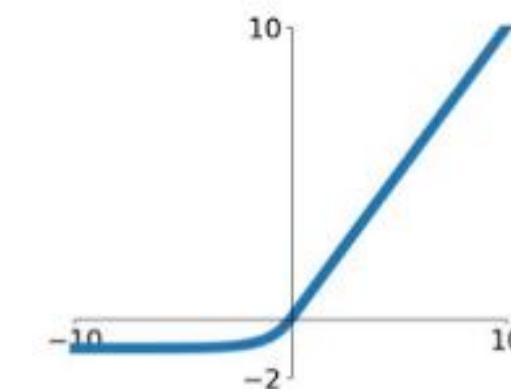


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



RELU BASIS FUNCTIONS

Piecewise continuous basis functions

Training Loop

```
import torch.nn as nn
xtrain, ytrain, xval, yval = load_data(npts=500, train_fraction = 0.30)

# MODEL
n_bases = 100
model  = nn.Sequential(
    nn.Linear(1, n_bases),
    nn.ReLU(),
    nn.Linear(n_bases, 1))

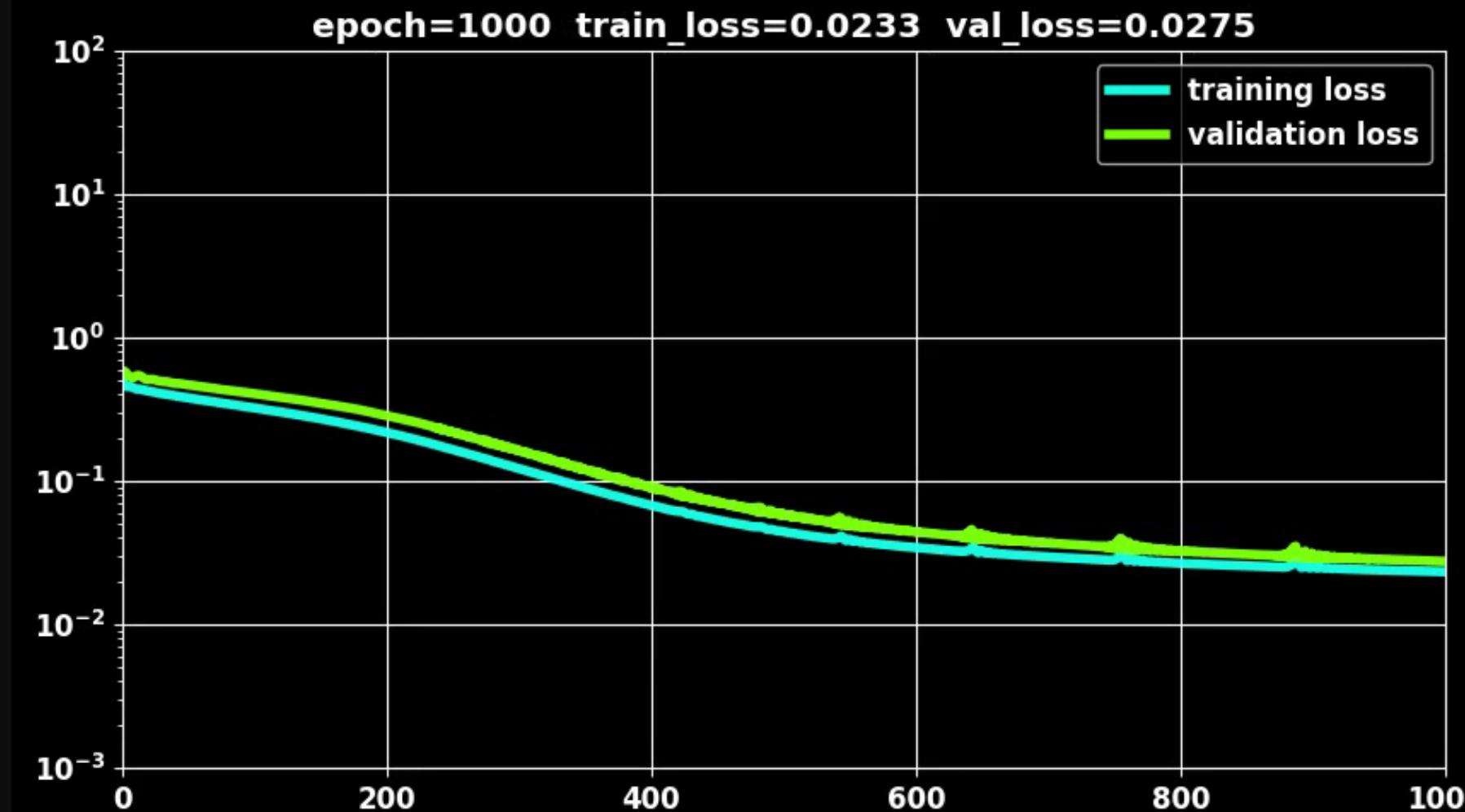
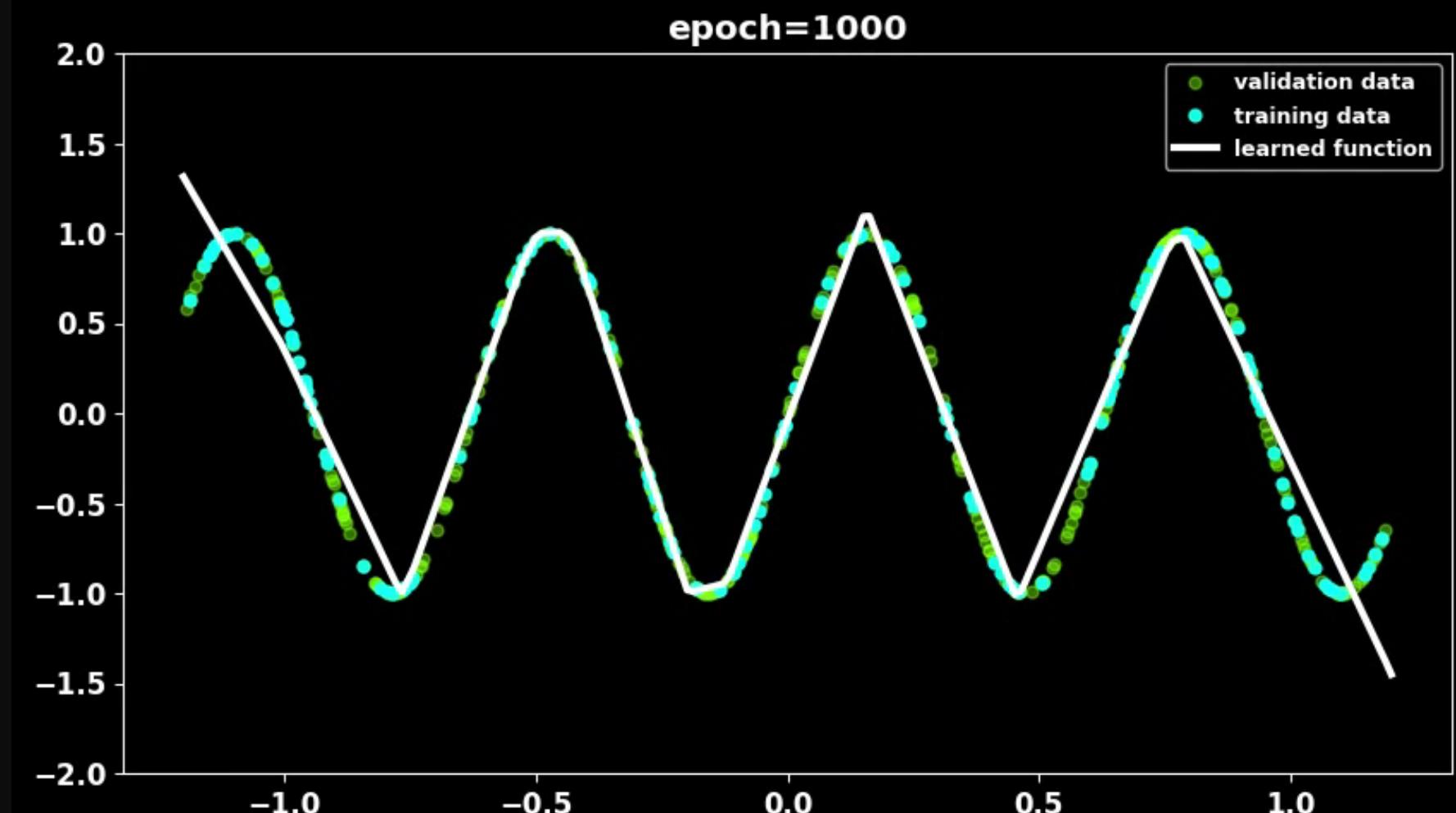
optimizer = torch.optim.Adam(model.parameters(), lr=5.0e-3)
epochs   = 6000

for i in range(epochs+1):

    # training
    optimizer.zero_grad()
    yhat = model(xtrain)
    loss = (yhat - ytrain).pow(2).mean()
    loss.backward()
    optimizer.step()

    # validation
    yval_hat = model(xval)
    loss_val = (yval_hat - yval).pow(2).mean()
```

(Pytorch Code)



MULTI-LAYER NETWORKS

Piecewise continuous basis functions

Training Loop

```
import torch.nn as nn

xtrain,ytrain,xval,yval=load_data(npts=500, train_fraction = 0.30)

# MODEL
n1,n2,n3 = 100,100,10

model = nn.Sequential(
    nn.Linear(1, n1), nn.ReLU(),
    nn.Linear(n1, n2), nn.ReLU(),
    nn.Linear(n2, n3), nn.ReLU(),
    nn.Linear(n3, 1))

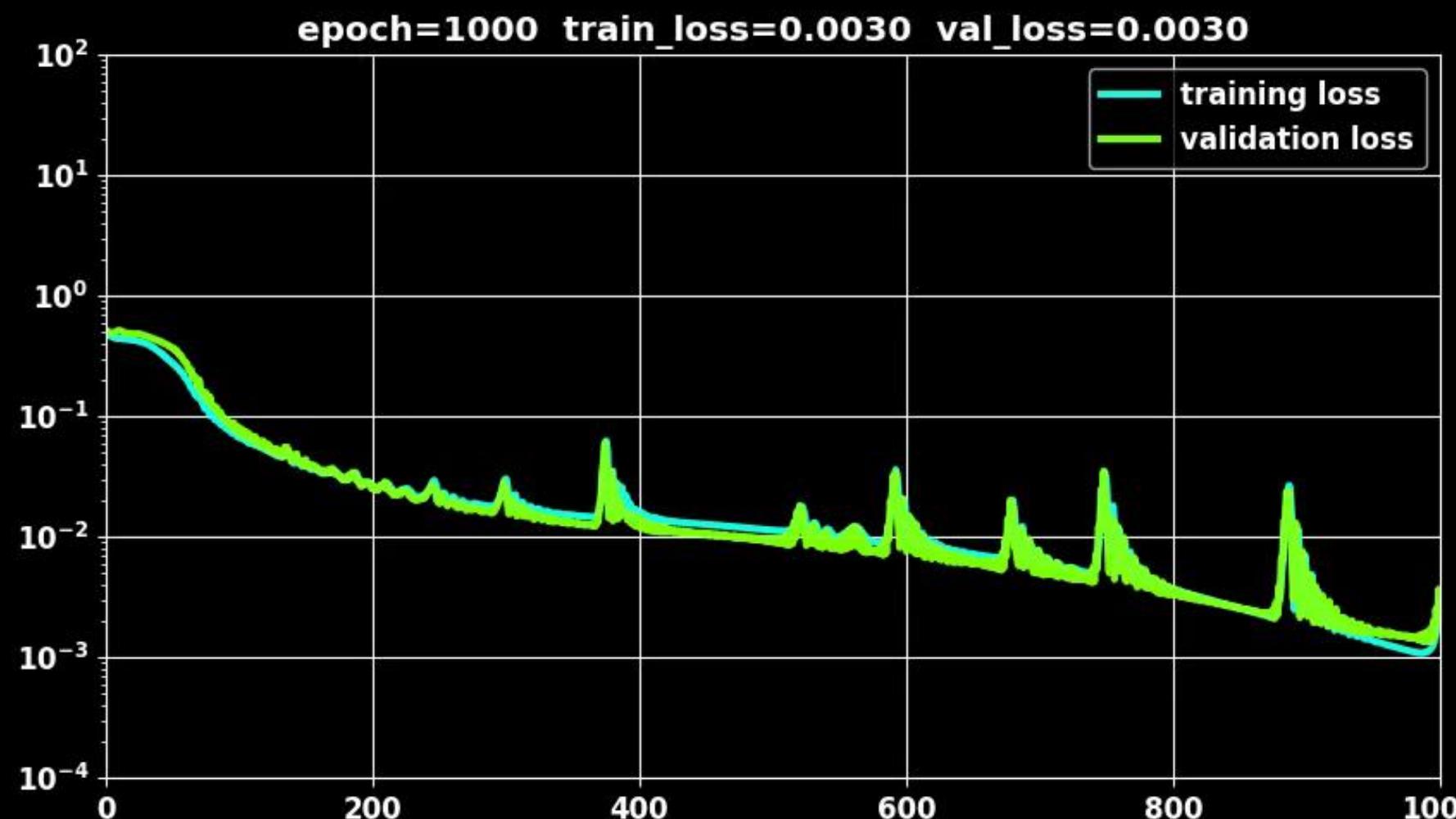
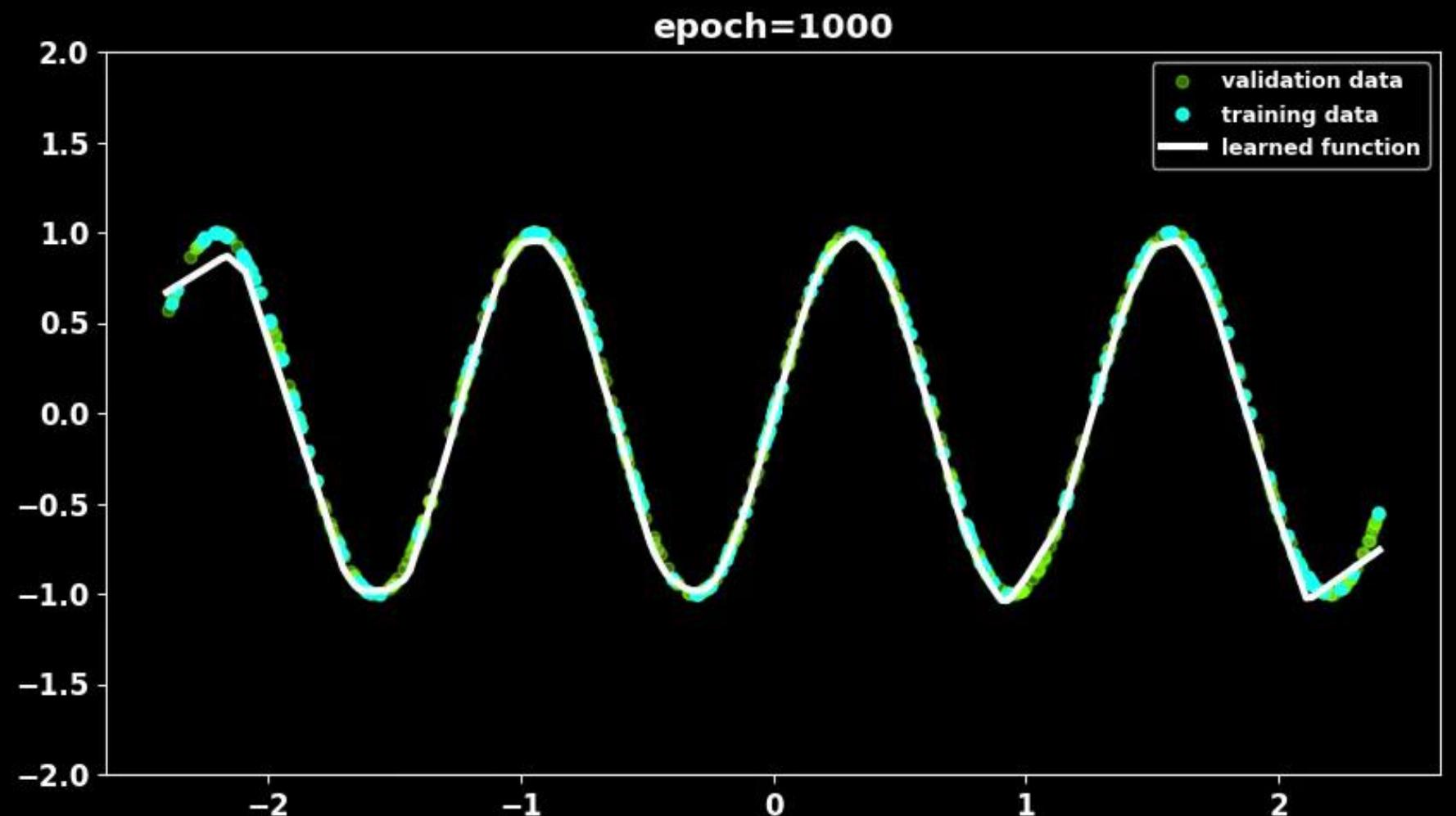
optimizer = torch.optim.Adam(model.parameters(), lr=5.0e-3)

epochs      = 1000

for i in range(epochs+1):

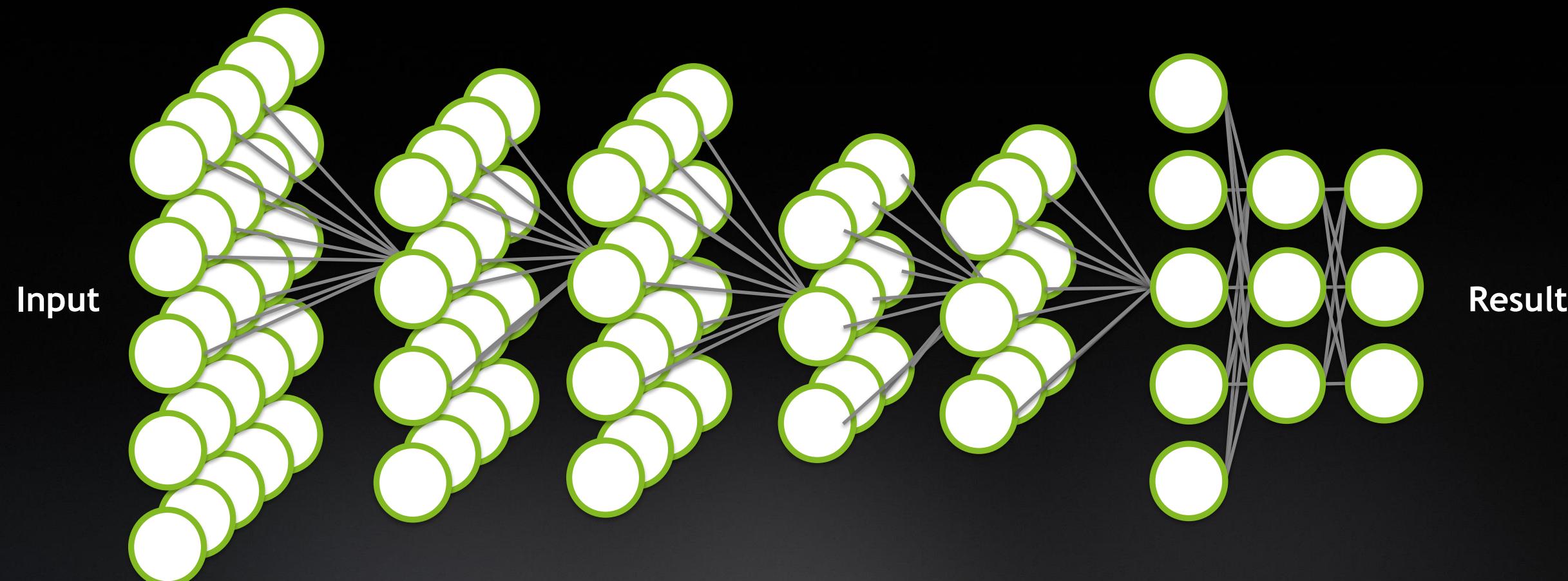
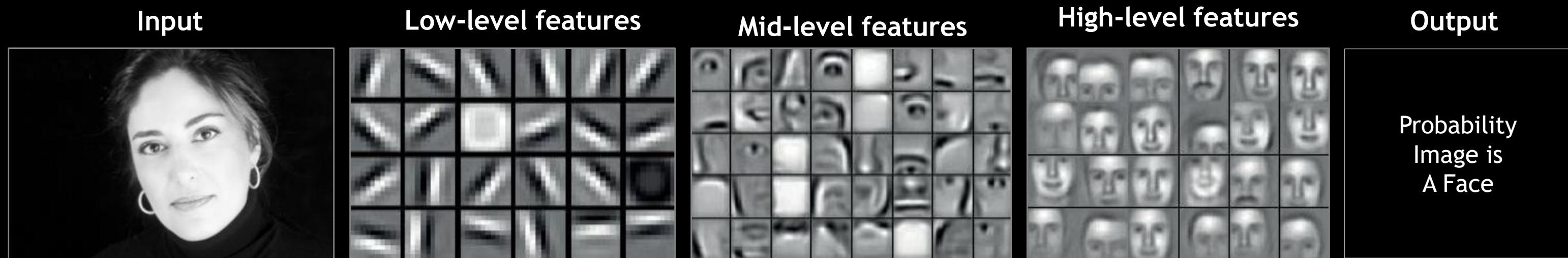
    # training
    optimizer.zero_grad()
    yhat = model(xtrain)
    loss = (yhat - ytrain).pow(2).mean()
    loss.backward()
    optimizer.step()

    # validation
    yval_hat = model(xval)
    loss_val = (yval_hat - yval).pow(2).mean()
```



DEEPER NEURAL NETWORKS

More layers allows for more levels of abstraction



Large Scale Visual Recognition Challenge 2012



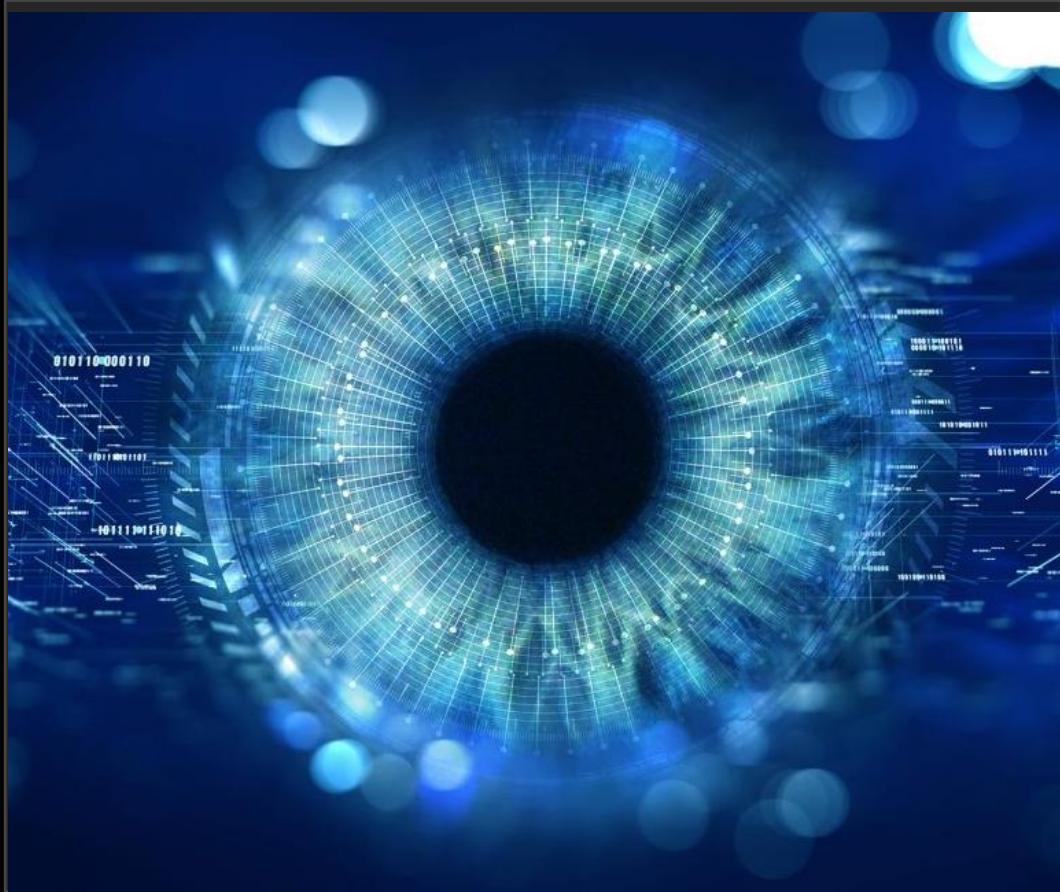
The Imagenet competition: Automatically classify images from 1000 different categories



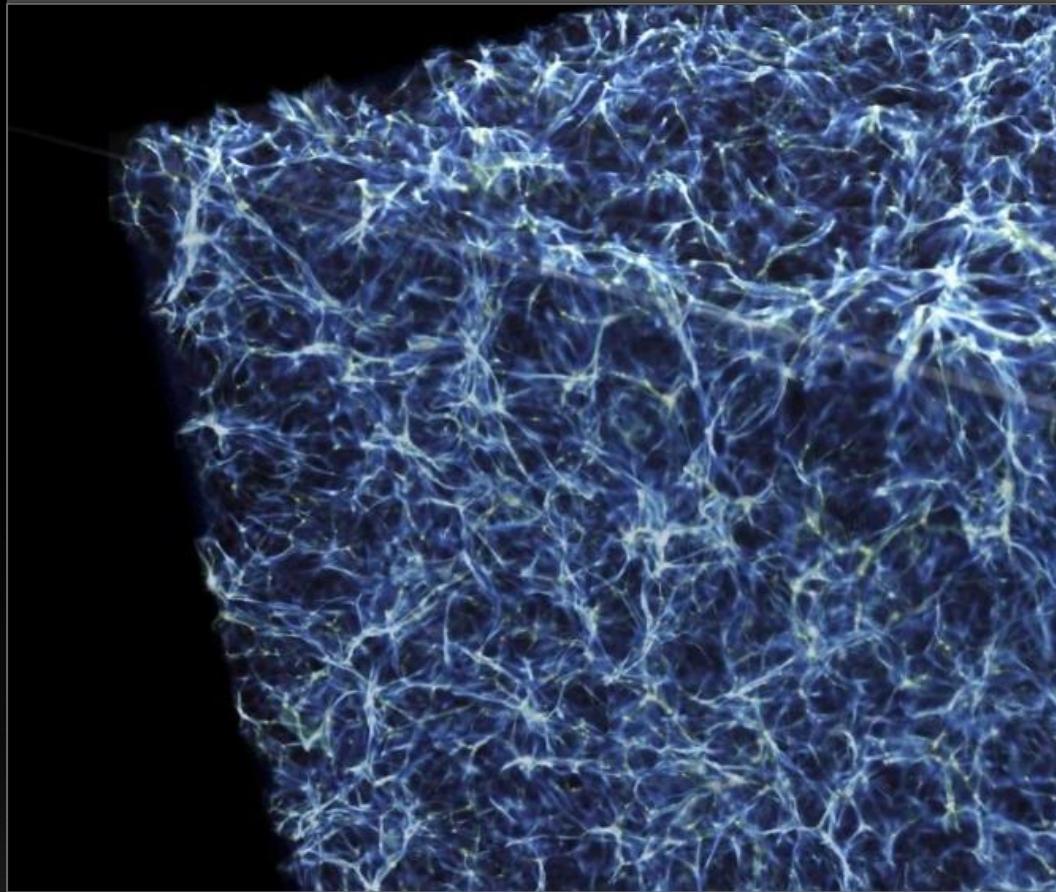
CONVOLUTIONAL NEURAL NETWORKS

WHAT ARE CNNS USED FOR?

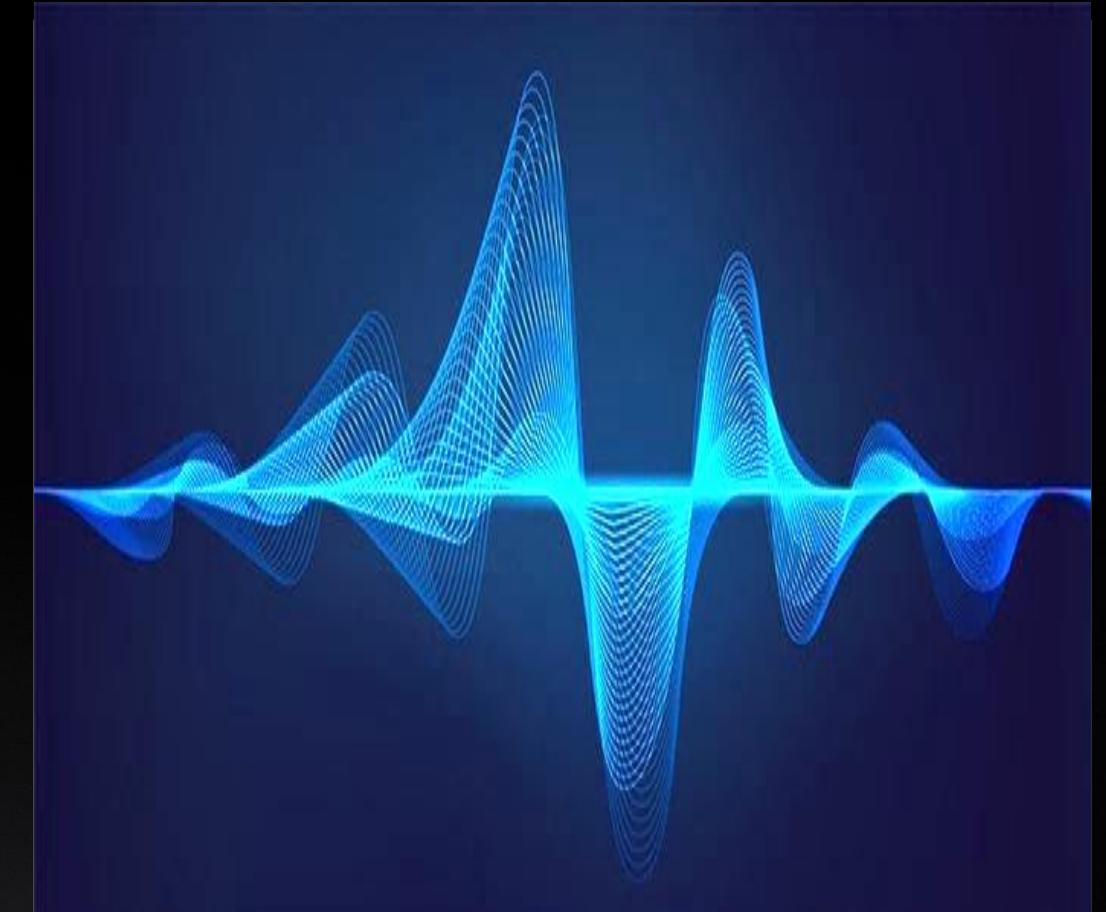
Problems with translational invariance



Computer Vision
Invariance in 2d space



Computational Physics
Invariance in 3d space



Audio and Time Series
Invariance in time

COMPUTER VISION TASKS

Each task requires a different model and data setup

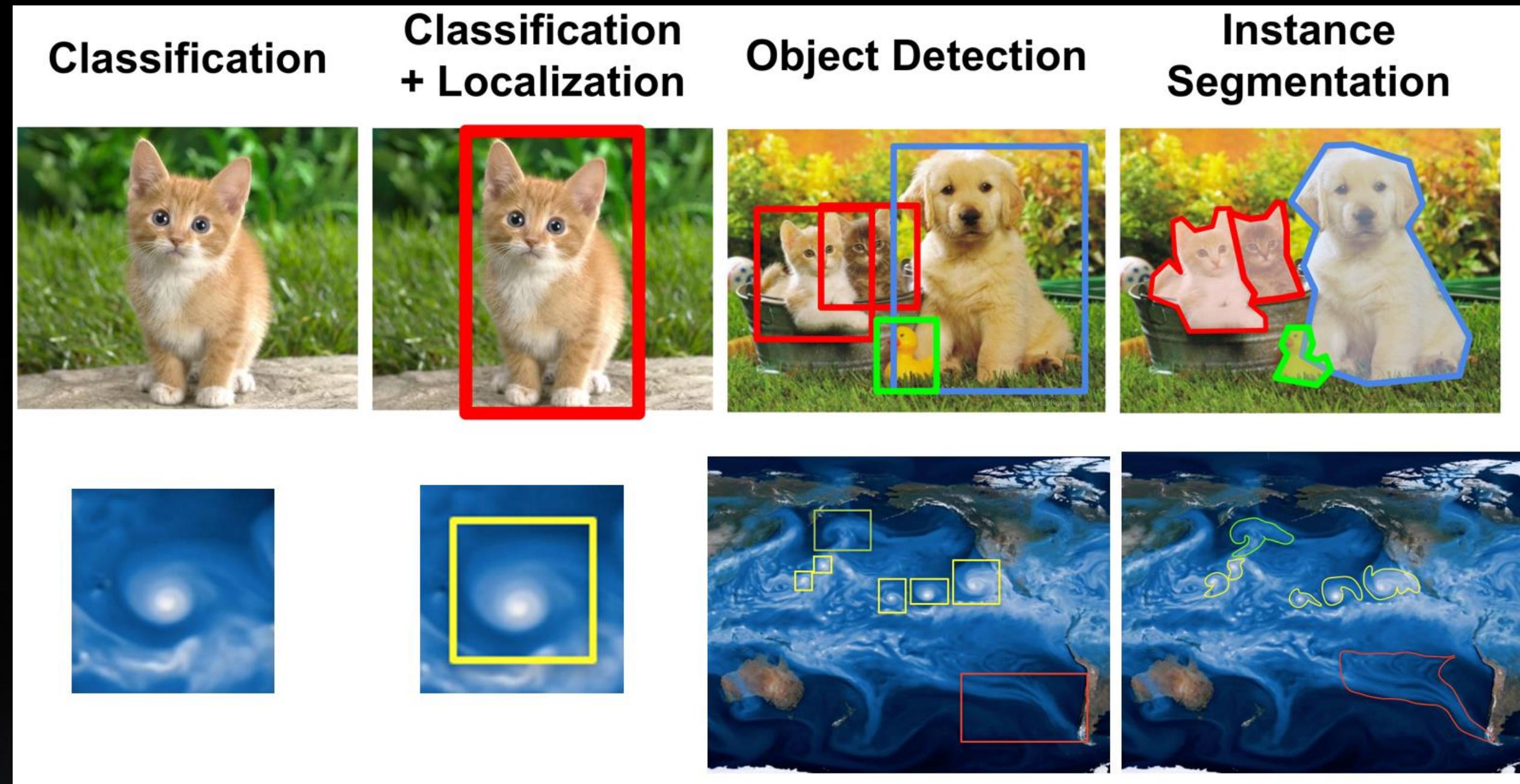


Image Credit: NERSC

CLASSIFICATION

Example: Classifying Land Use



UC Merced Land Use Database

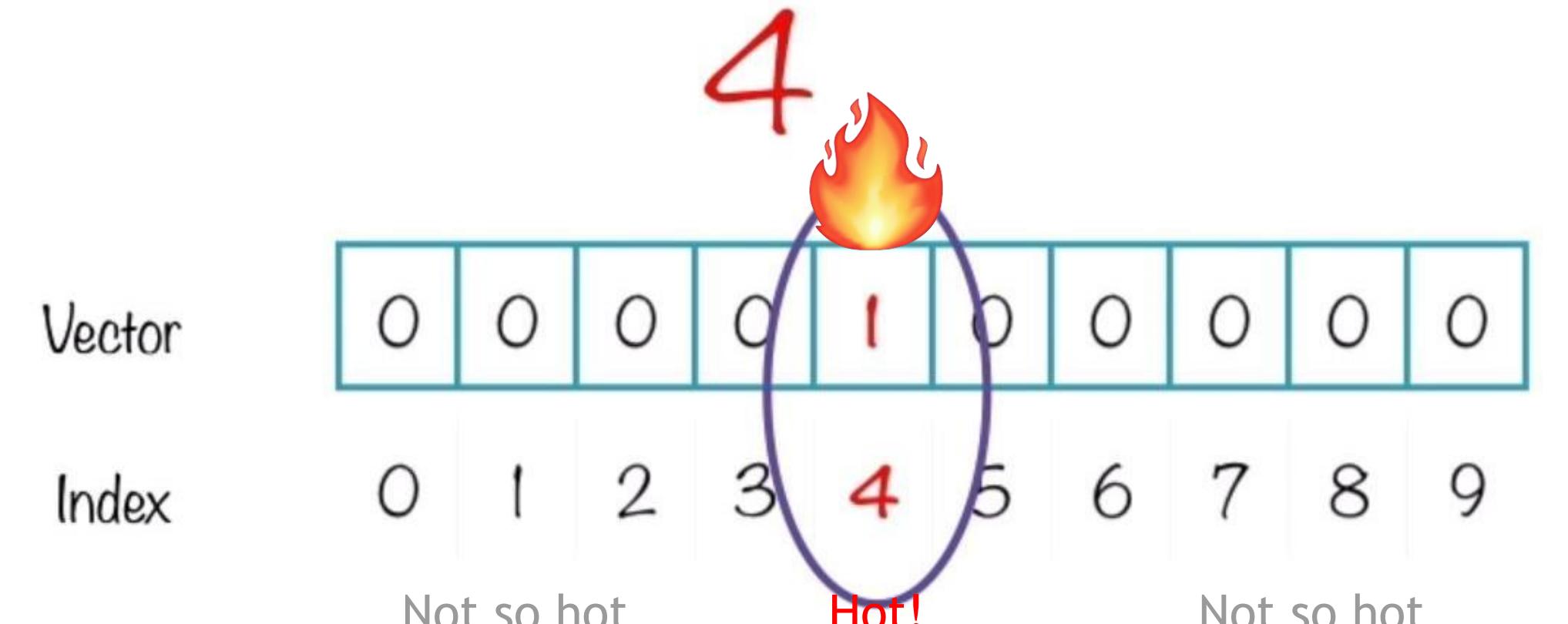
ONE-HOT ENCODING

Input: Pixels, Output: One-hot encoding

INPUT:PIXEL VALUES



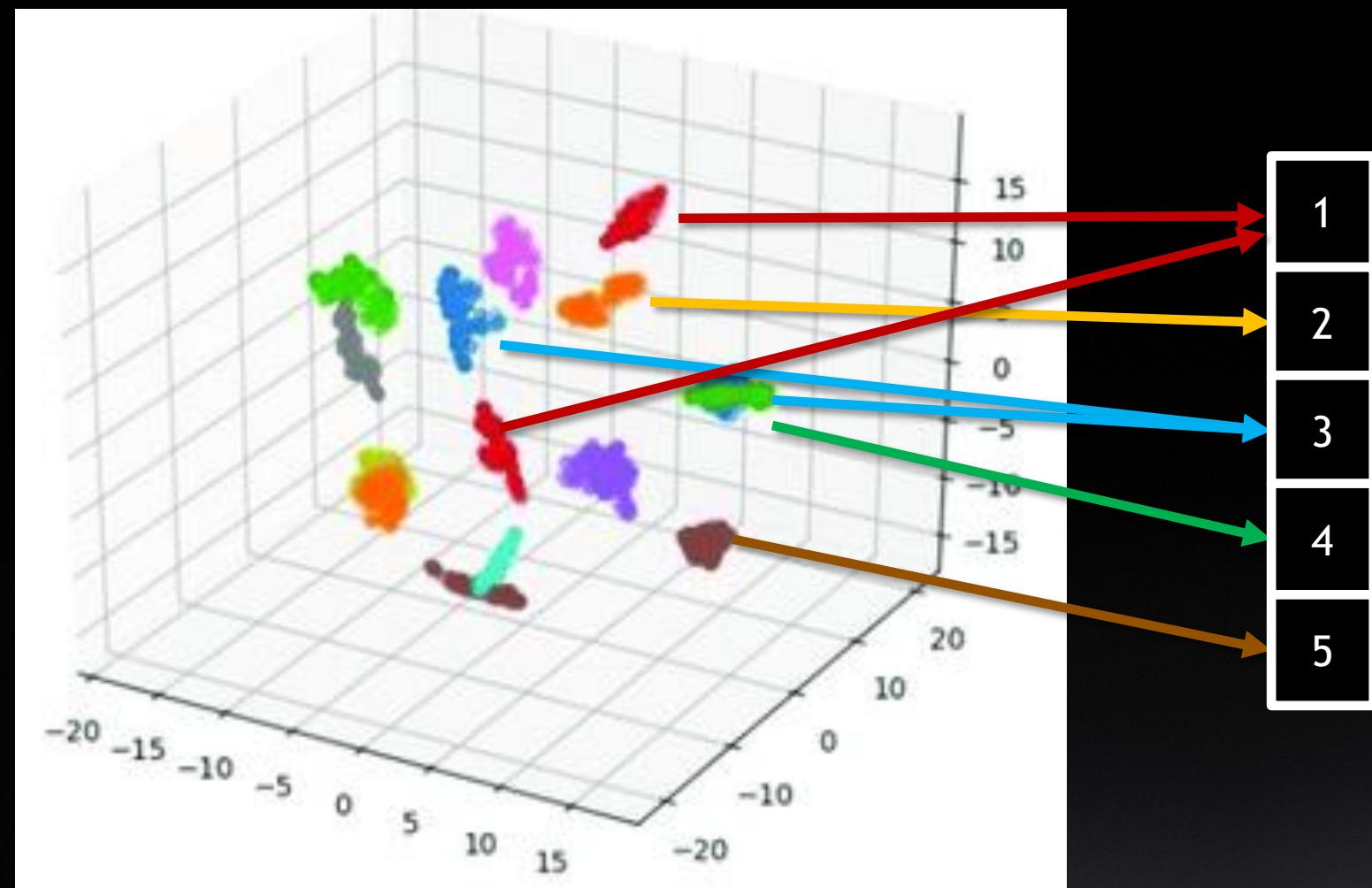
OUTPUT: ONE-HOT VECTOR



<https://blog.carboniteq.com/practical-image-recognition-with-tensorflow/>

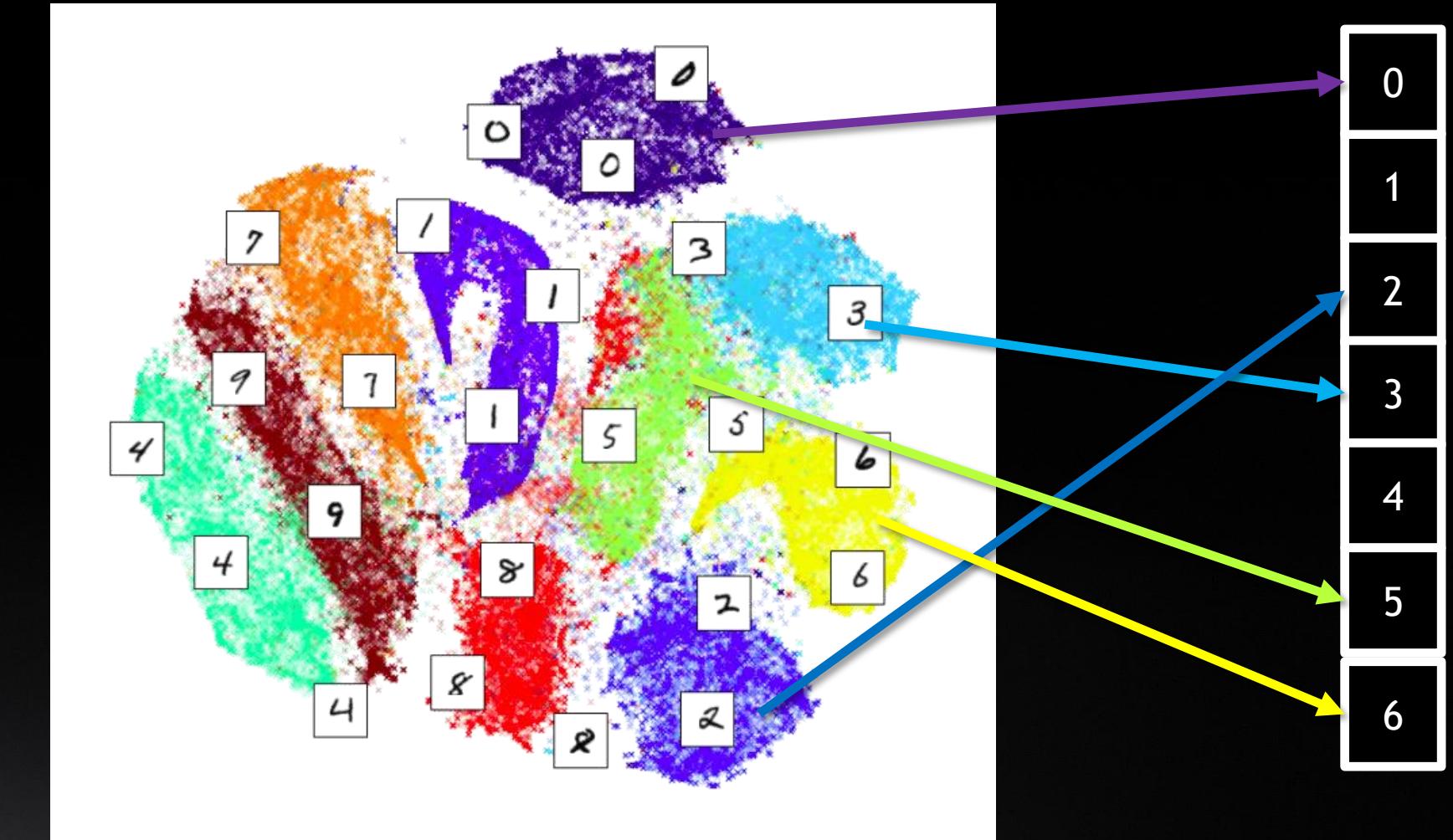
IMAGES ARE POINTS, WITH MANY DIMENSIONS

IN: 3-D Vector



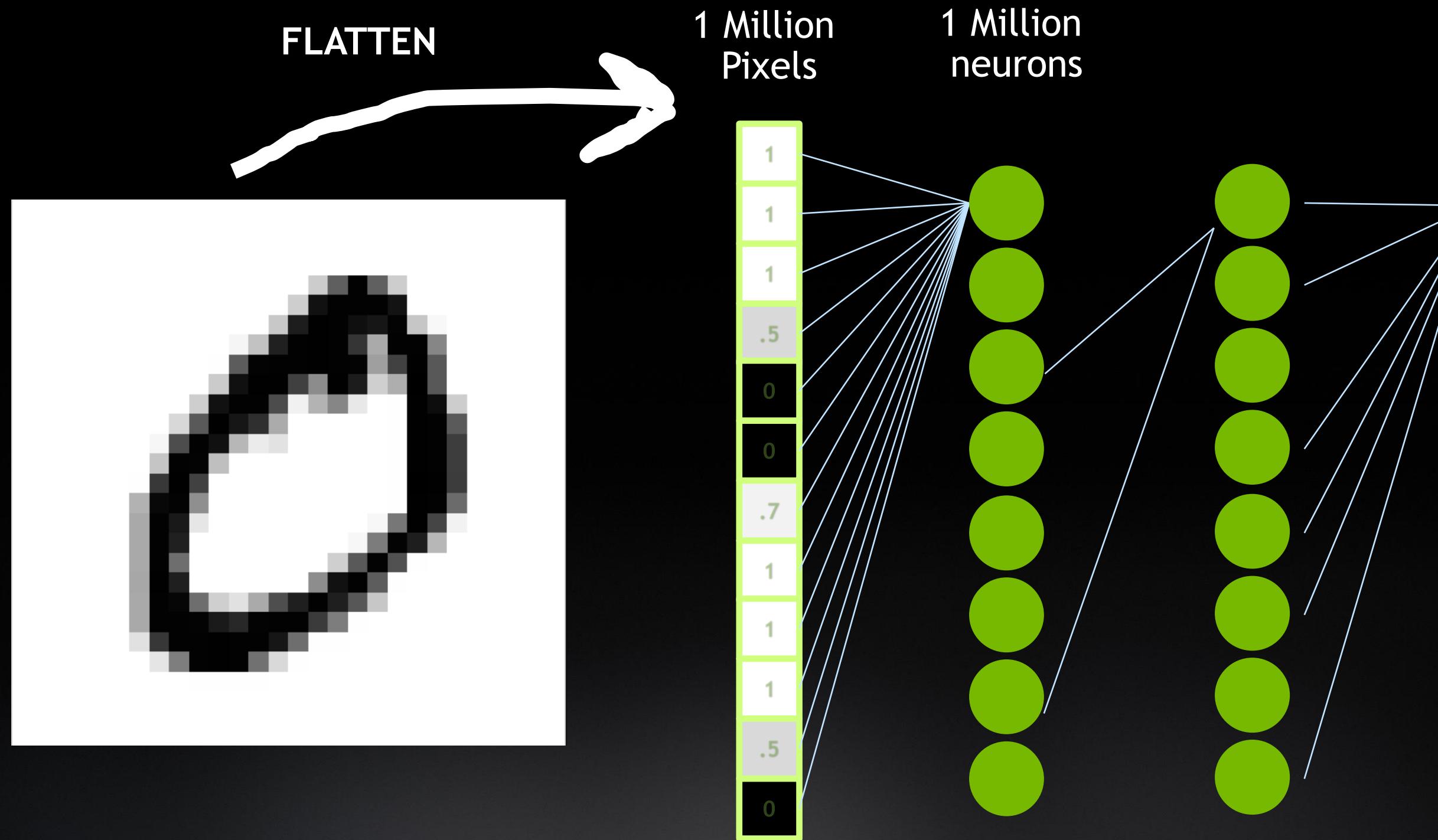
OUT: 1 hot vector

IN: 784-D Vector



OUT: 1-hot vector

FULLY CONNECTED NETWORKS AND IMAGES DON'T MIX



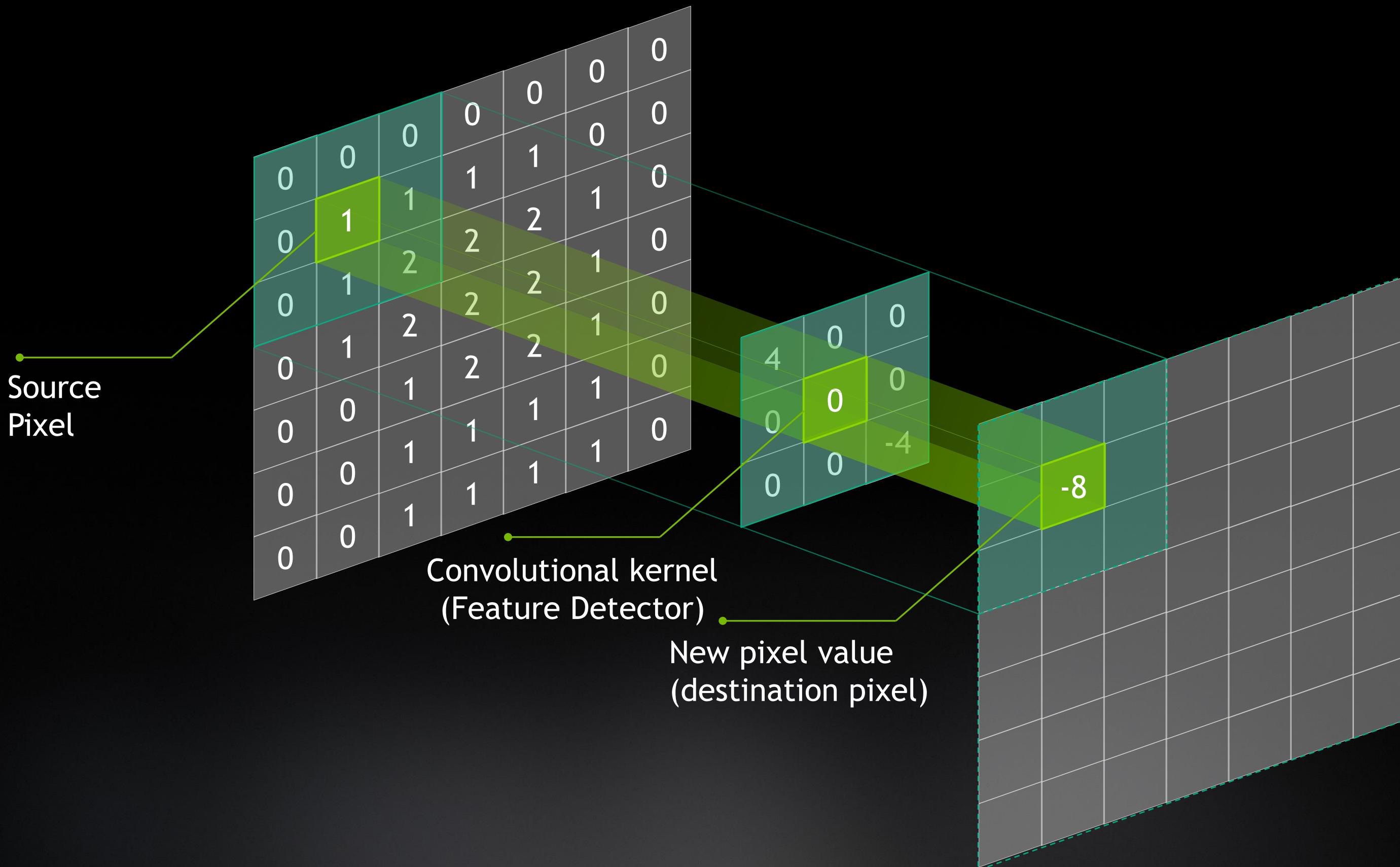
TRANSLATIONAL EQUIVARIANCE

Objects in nature look the same from place to place

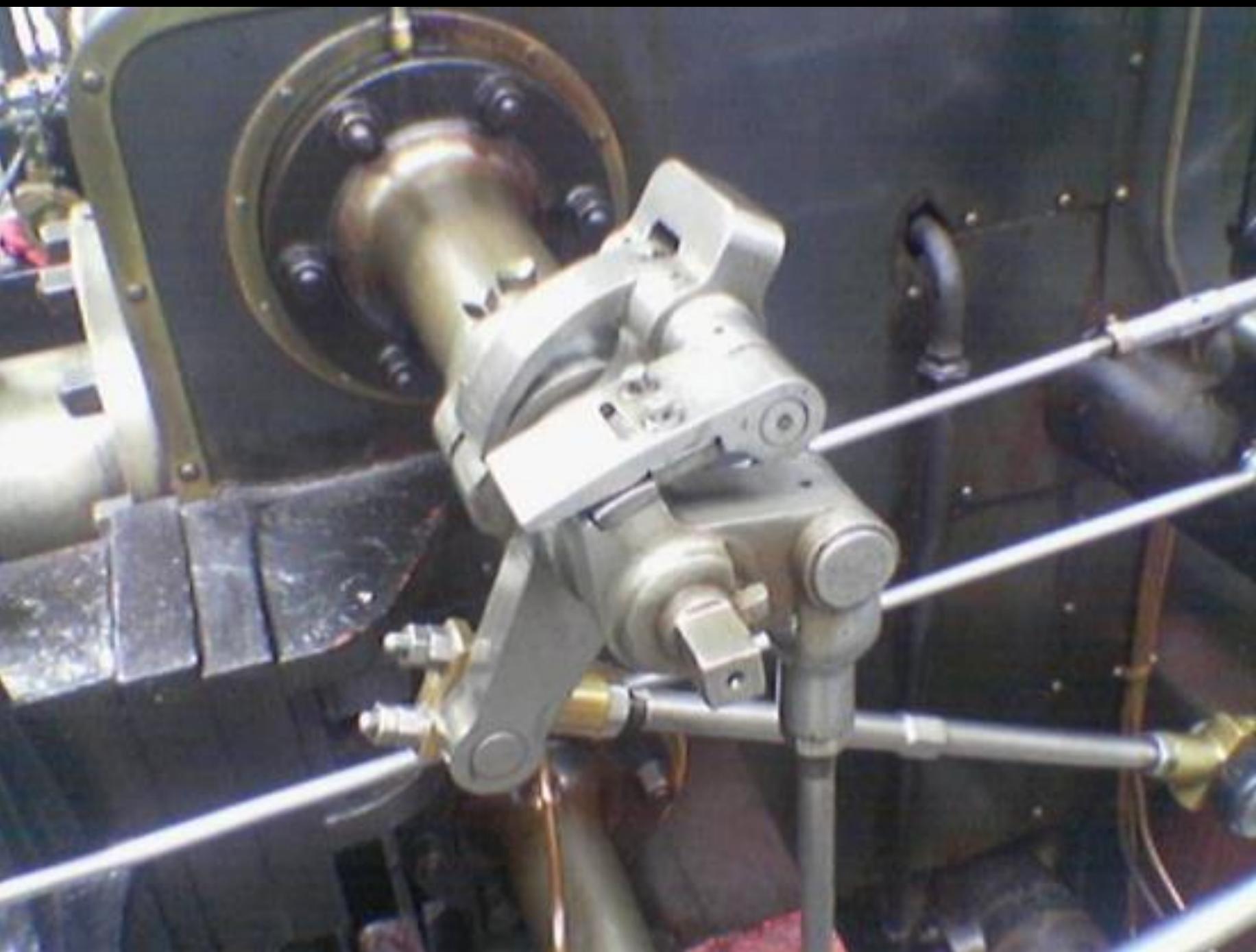


WHAT IS A CONVOLUTION?

A small matrix transformation, applied at each point of the image



CONVOLUTION EXAMPLE: SOBEL FILTER



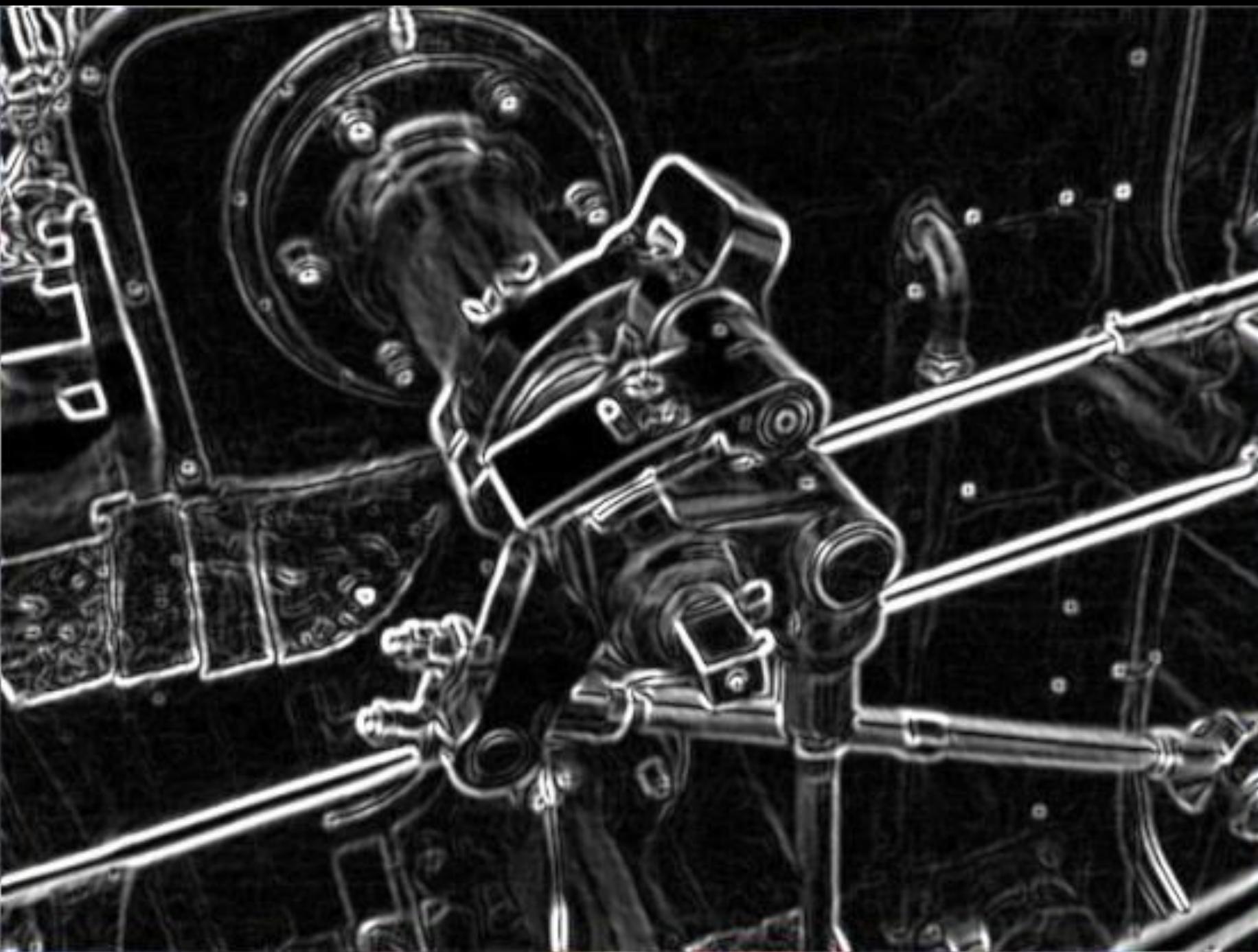
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Image source: https://en.wikipedia.org/wiki/Sobel_operator

CONVOLUTION EXAMPLE: SOBEL FILTER



$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Image source: https://en.wikipedia.org/wiki/Sobel_operator



CLASSIFICATION

CLASSIFIER EVOLUTION OVER TIME



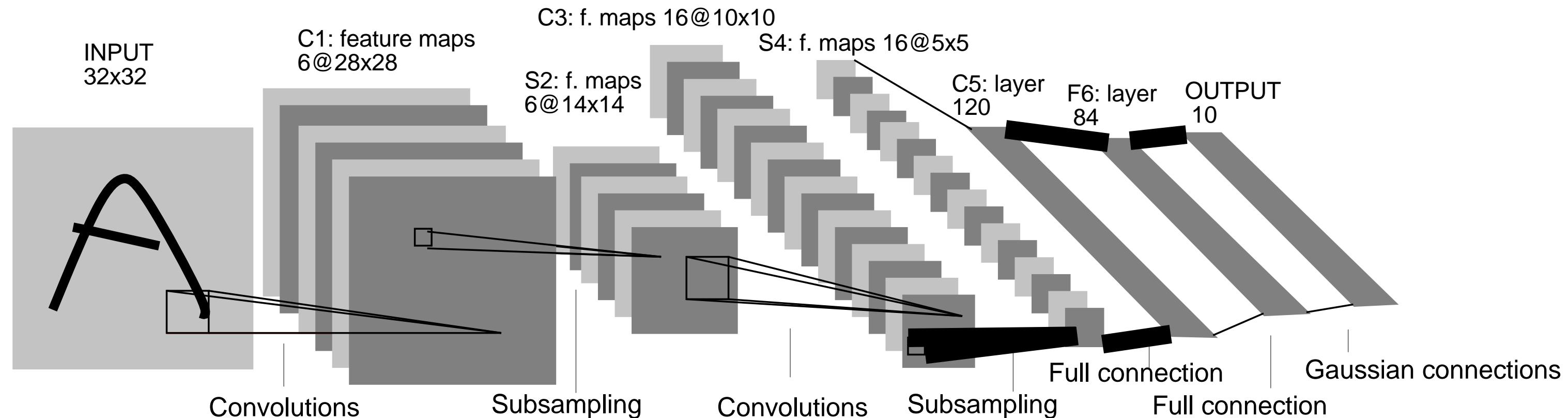
LENET-5

(1988) Yann LeCun. Hand written recognition. 60k parameters.

PROC. OF THE IEEE, NOVEMBER 1998

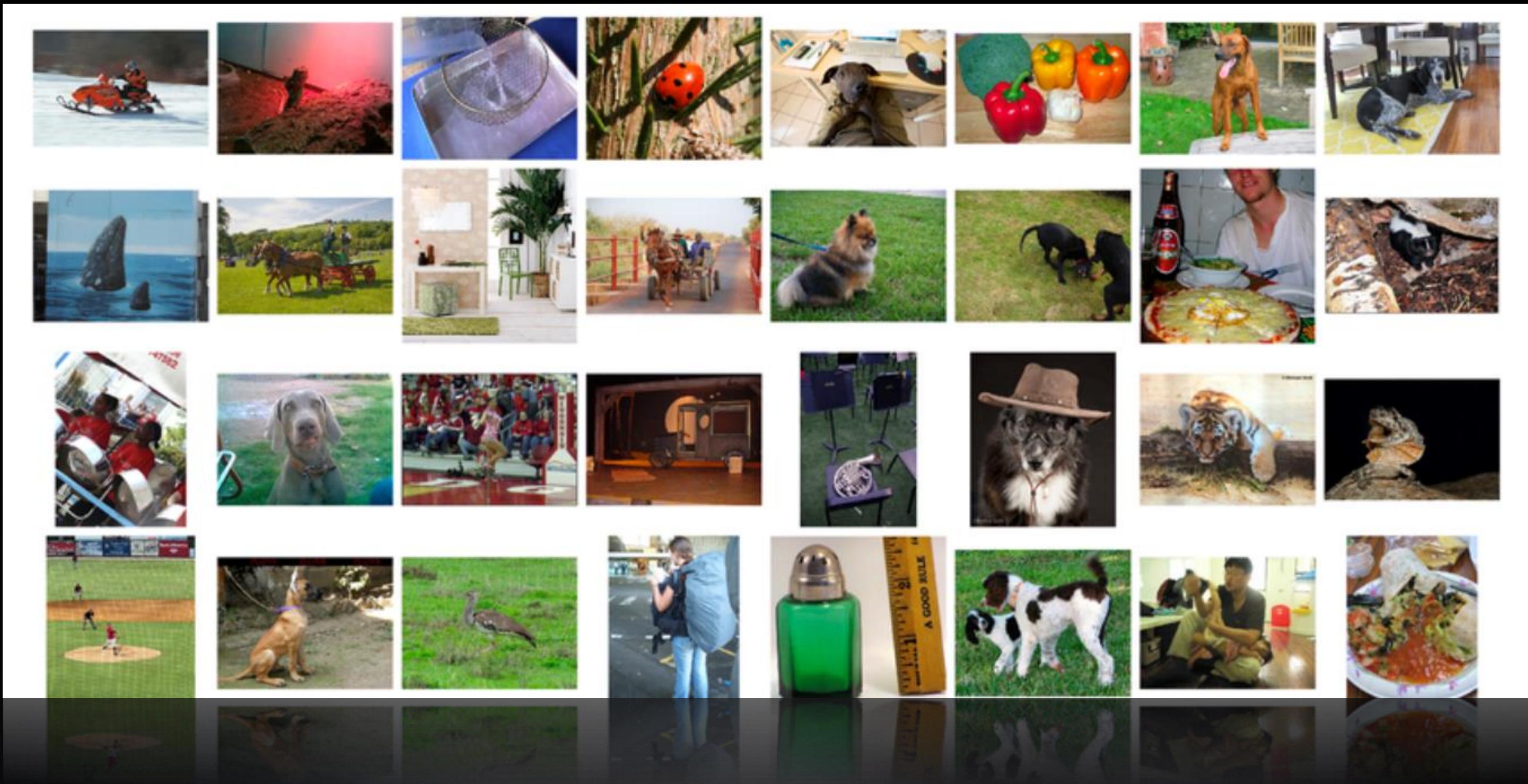
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner



IMAGENET ILSVR COMPETITION

Large Scale Visual Recognition Competition (2010-2017)



<https://en.wikipedia.org/wiki/ImageNet>

ALEXNET

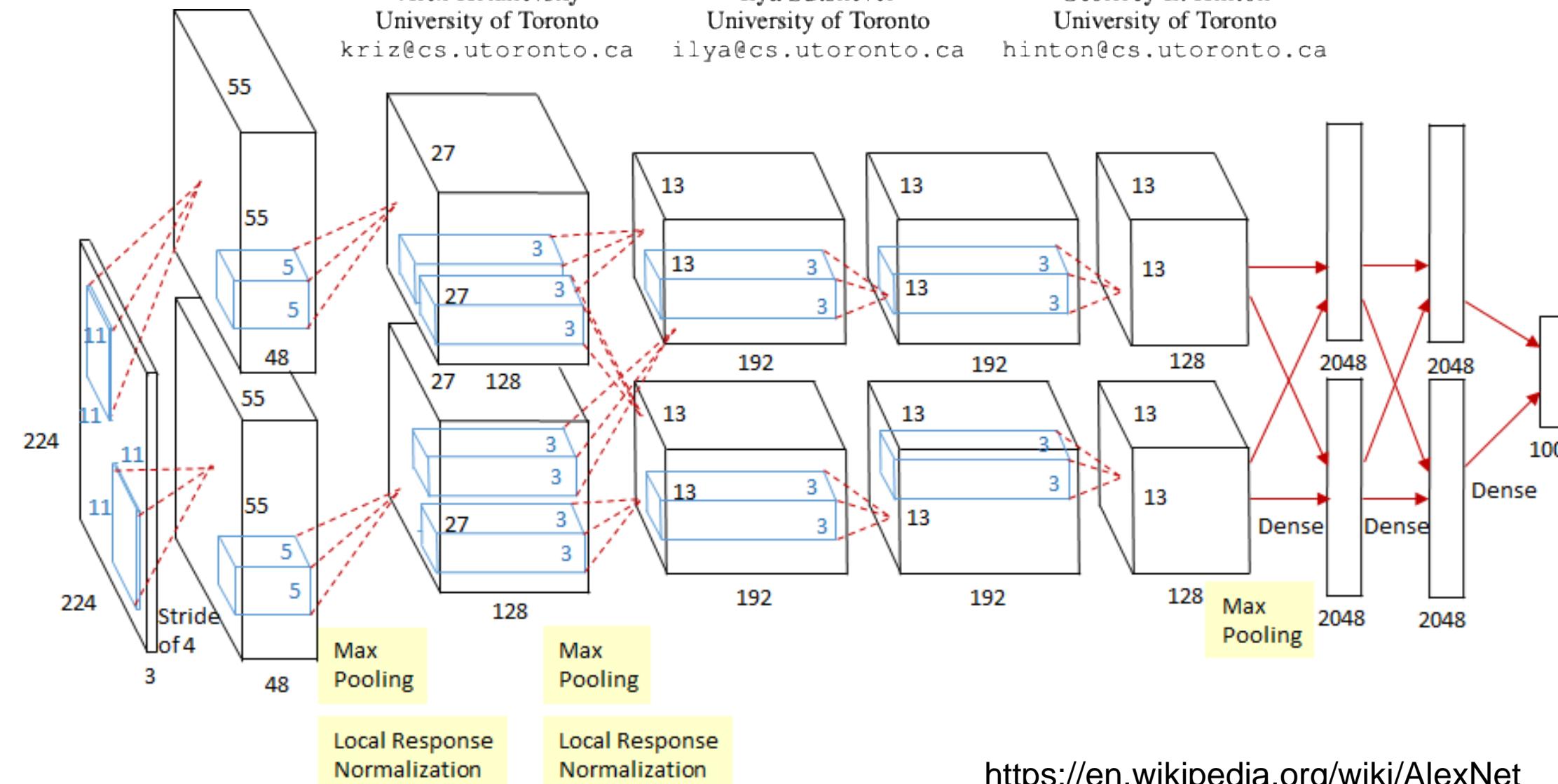
(2012): Krizhevsky, Sutskever, Hinton. ImageNet winner.

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



<https://en.wikipedia.org/wiki/AlexNet>

VGG-16

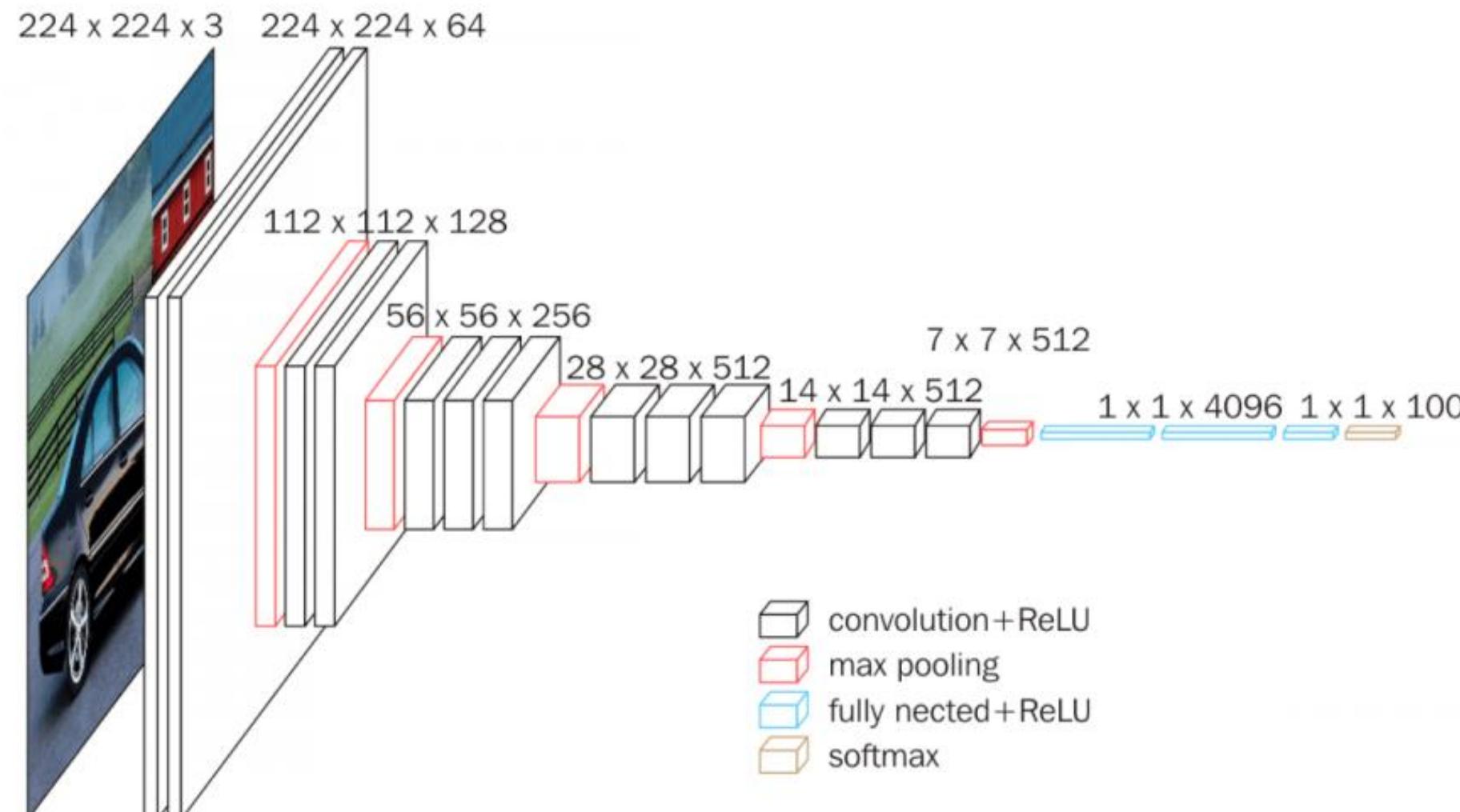
2014. ImageNet runner up. Simple, clean architecture.



VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

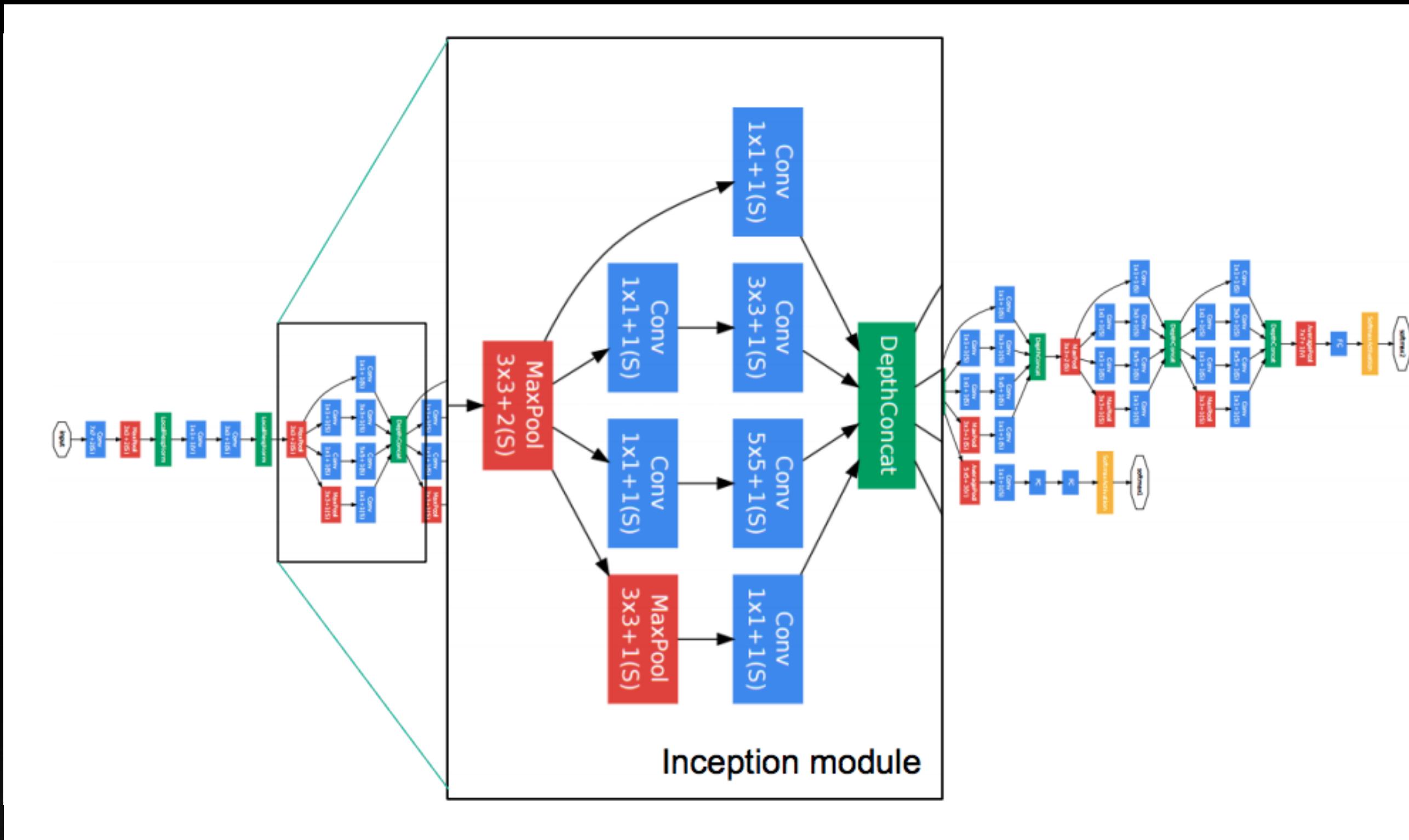
Karen Simonyan* & Andrew Zisserman⁺

Visual Geometry Group, Department of Engineering Science, University of Oxford



INCEPTION-V1 (GOOGLENET)

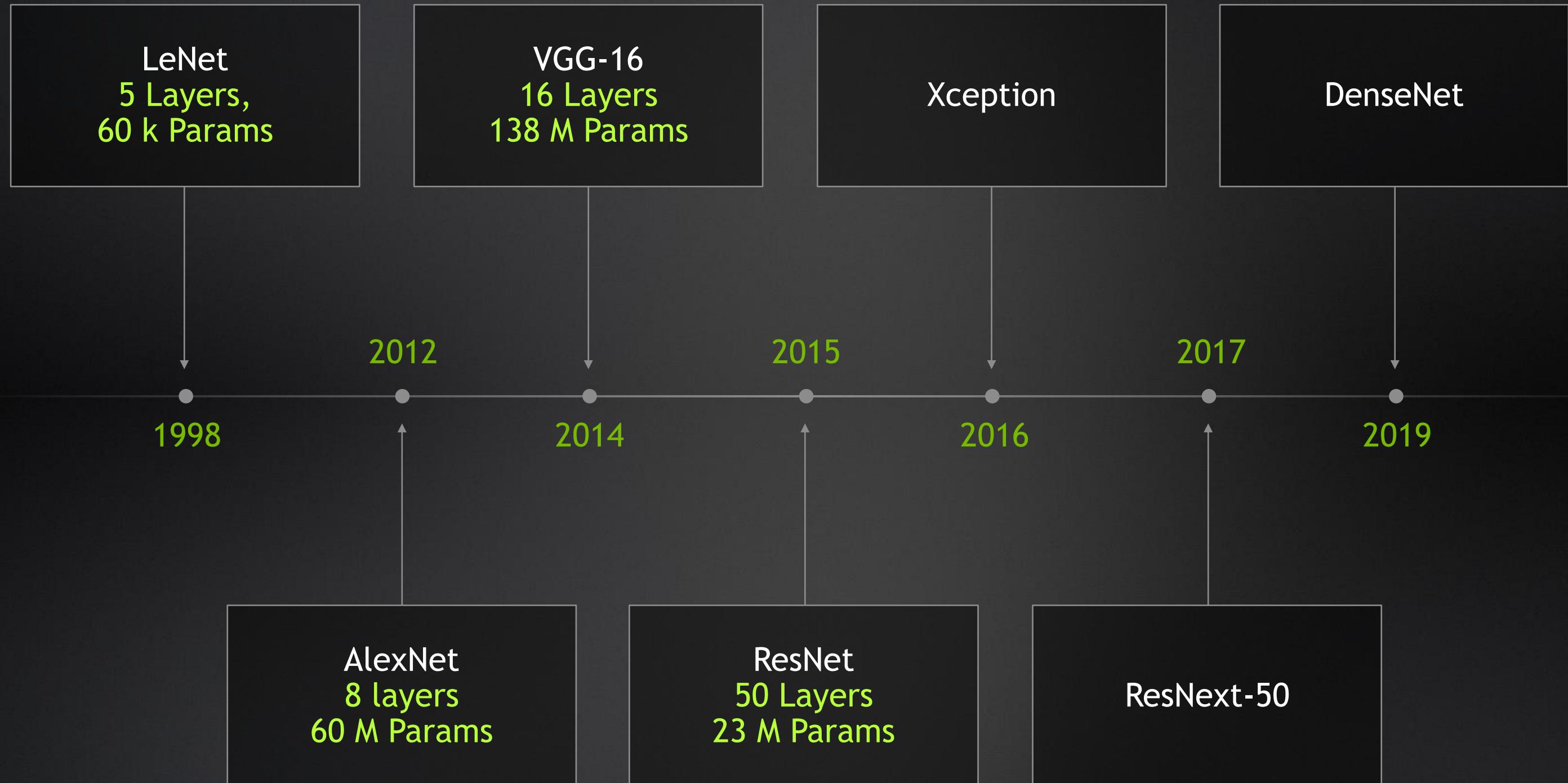
2014. Train different size convolutions in parallel



The background of the slide features a complex network graph. It consists of numerous small, semi-transparent white dots representing nodes, connected by thin grey lines representing edges. Interspersed among these are several larger, solid green dots, which also have grey connecting lines. The overall effect is one of a dense, interconnected system.

RESNETS

MODELING TRENDS: DEEPER AND LARGER

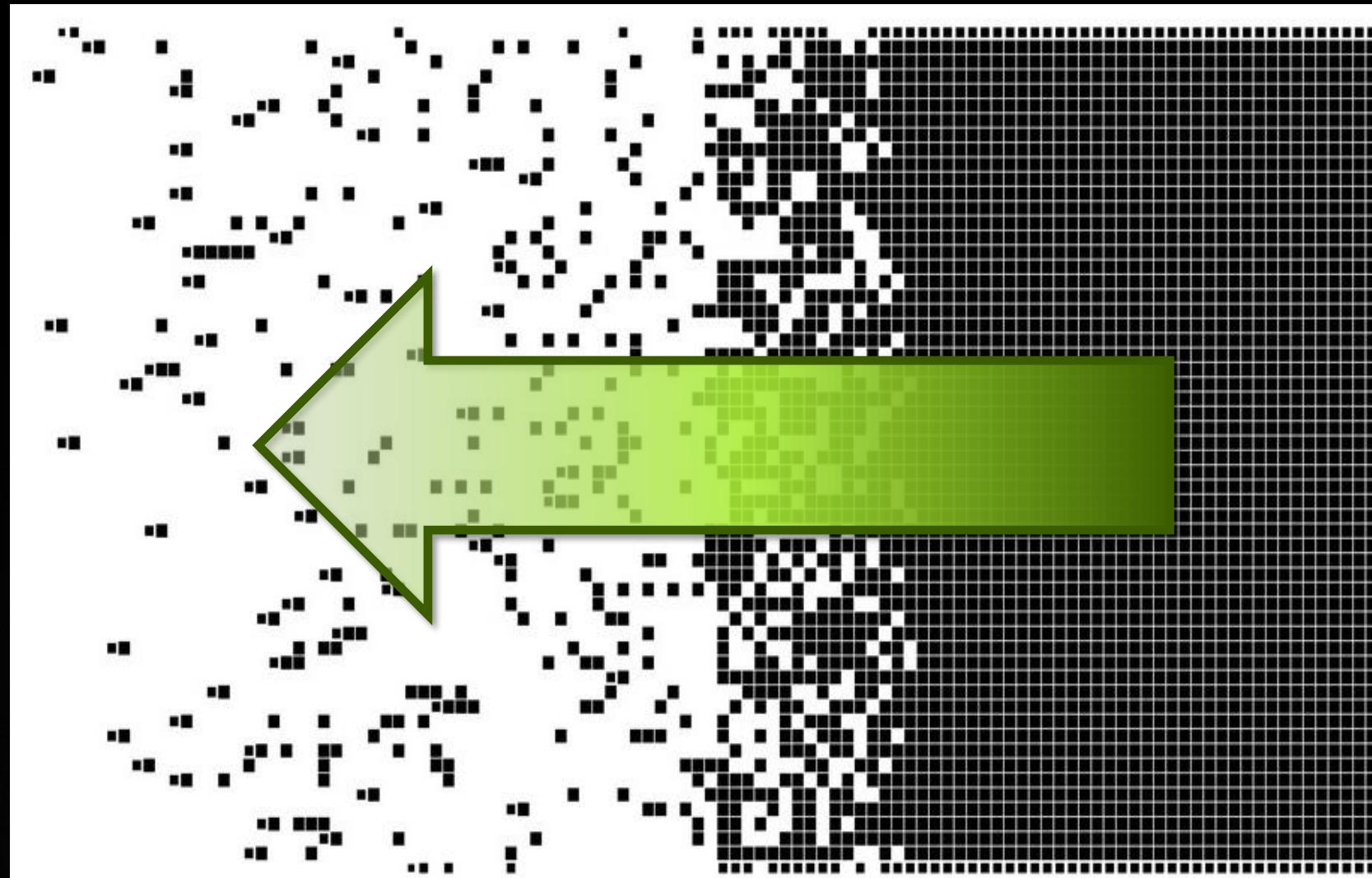


Source: Source information is 14 pt, italic

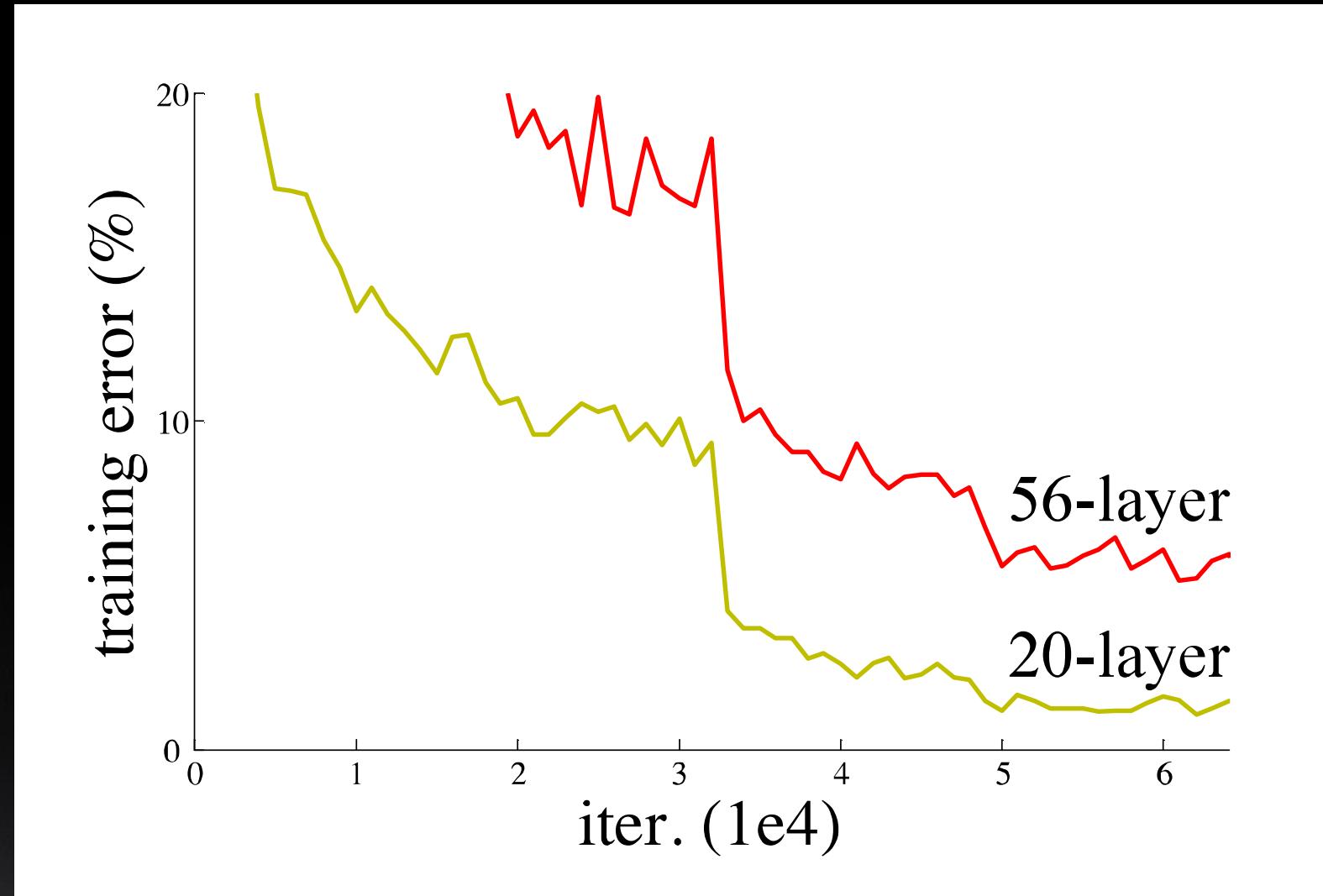
PROBLEM: VANISHING GRADIENTS

Error signal decays exponentially as it propagates backward through the network

ERROR SIGNAL VANISHES DURING BACKPROP



DEEPER NETWORKS WERE HARDER TO TRAIN

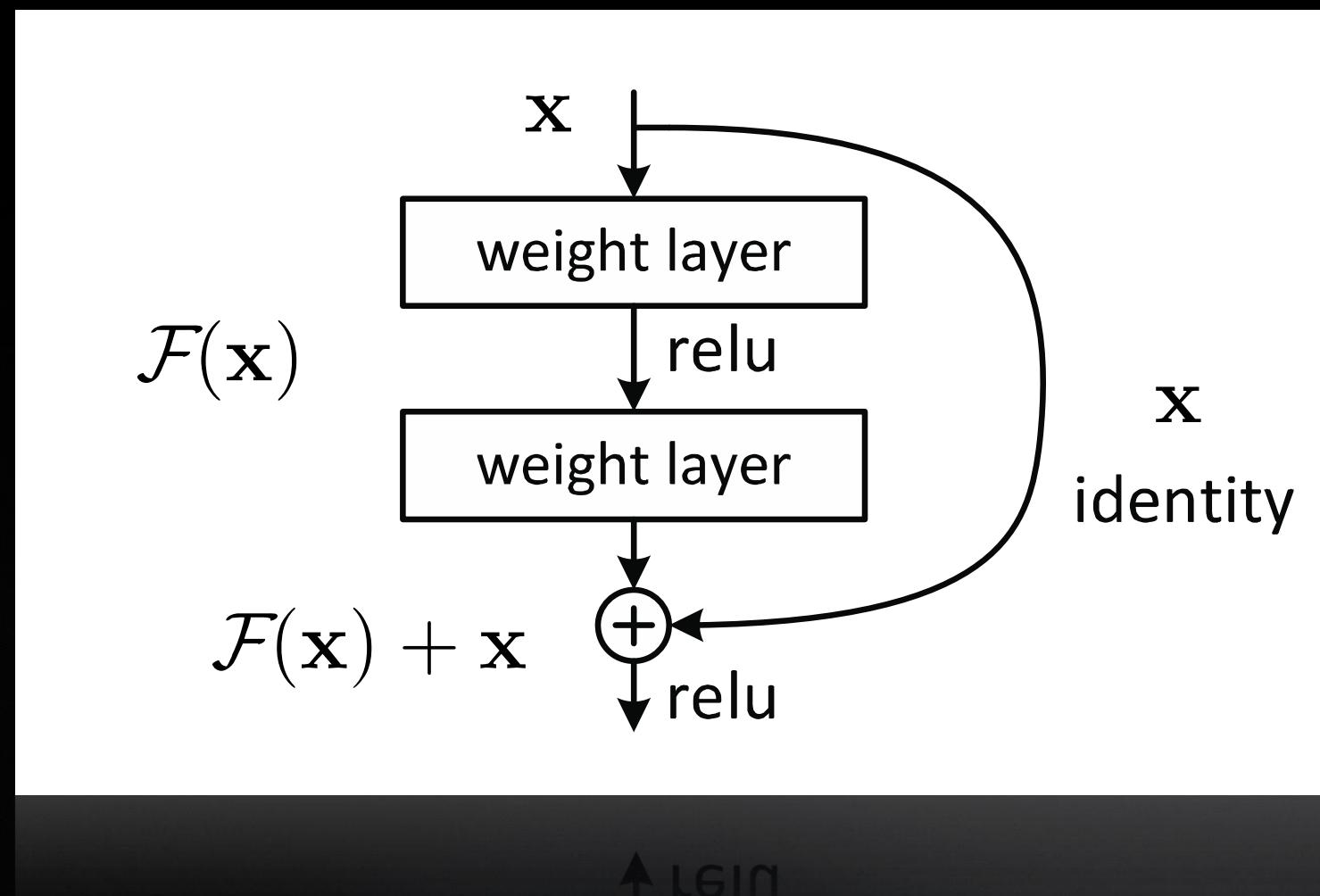


<https://www.arxiv-vanity.com/papers/1512.03385/>

RESNETS AND SKIP CONNECTIONS

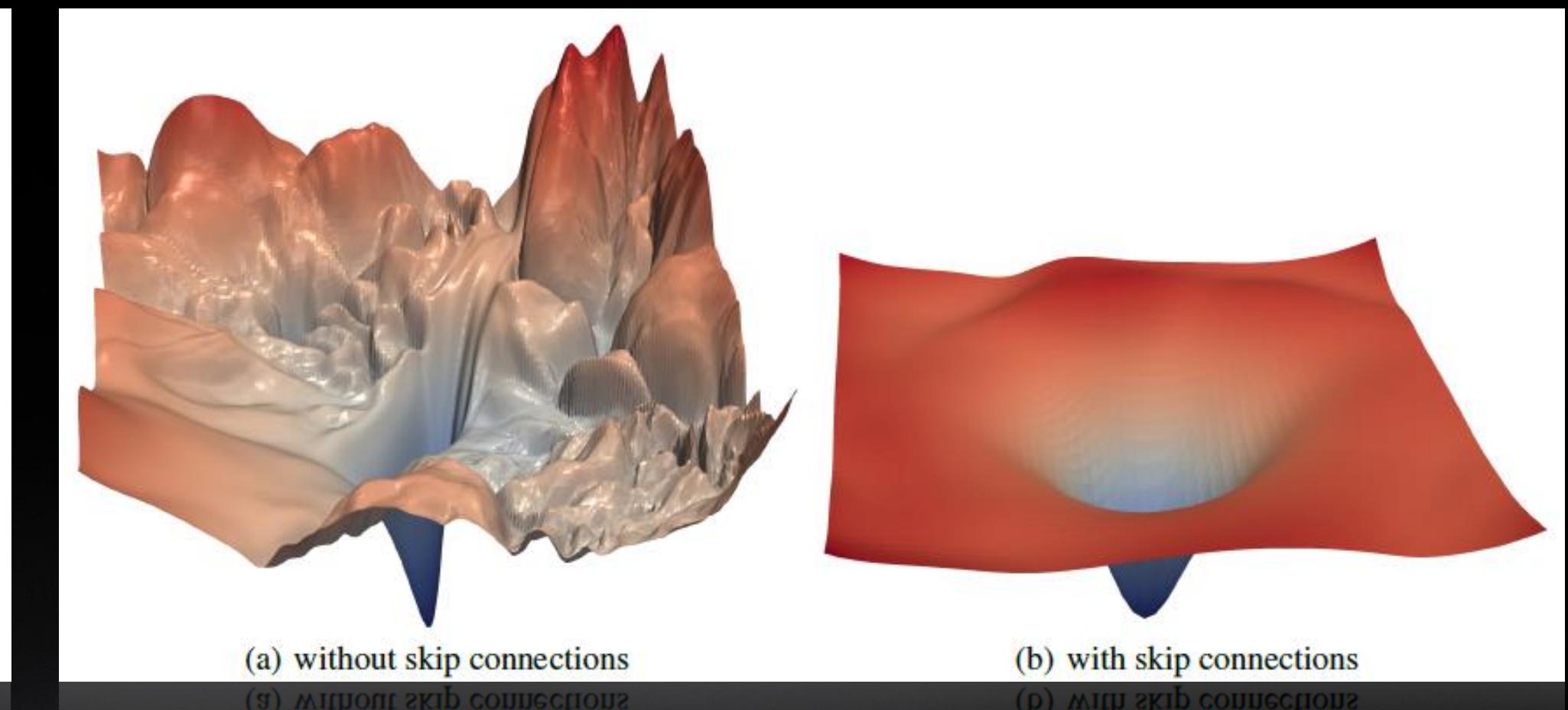
(aka Highway Networks)

ADD THE INPUT TO OUTPUT



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

DRAMATICALLY SIMPLIFIES THE LOSS LANDSCAPE



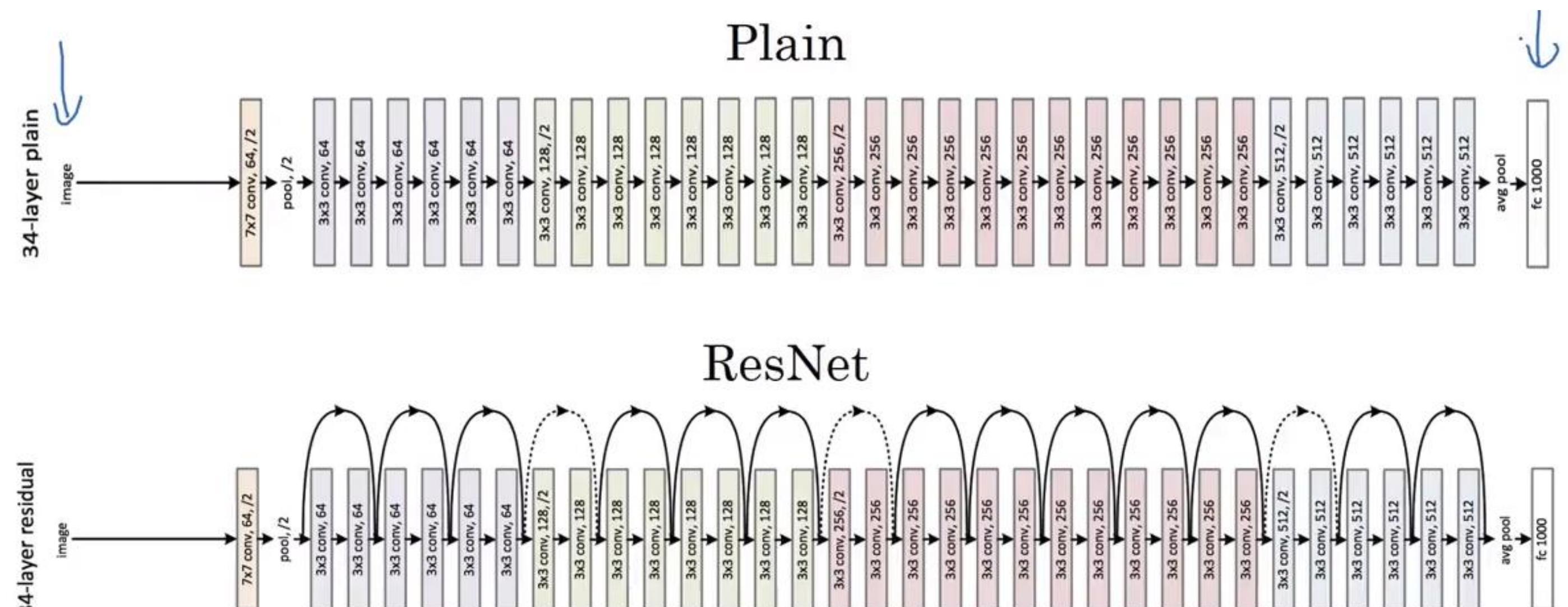
<https://arxiv.org/abs/1712.09913>
https://jithinjk.github.io/blog/nn_loss_visualized.md.html

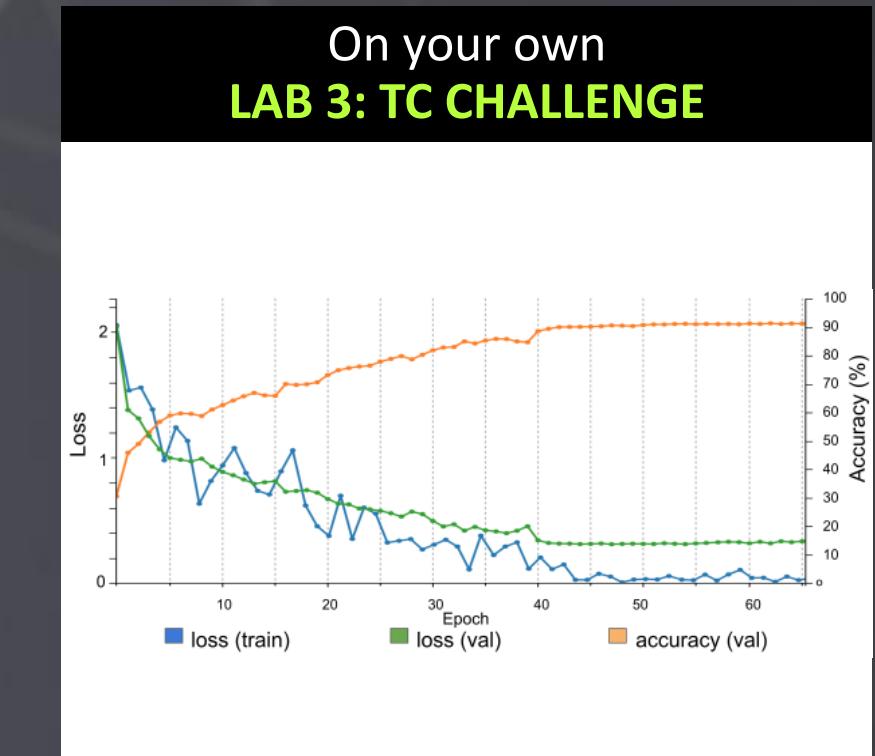
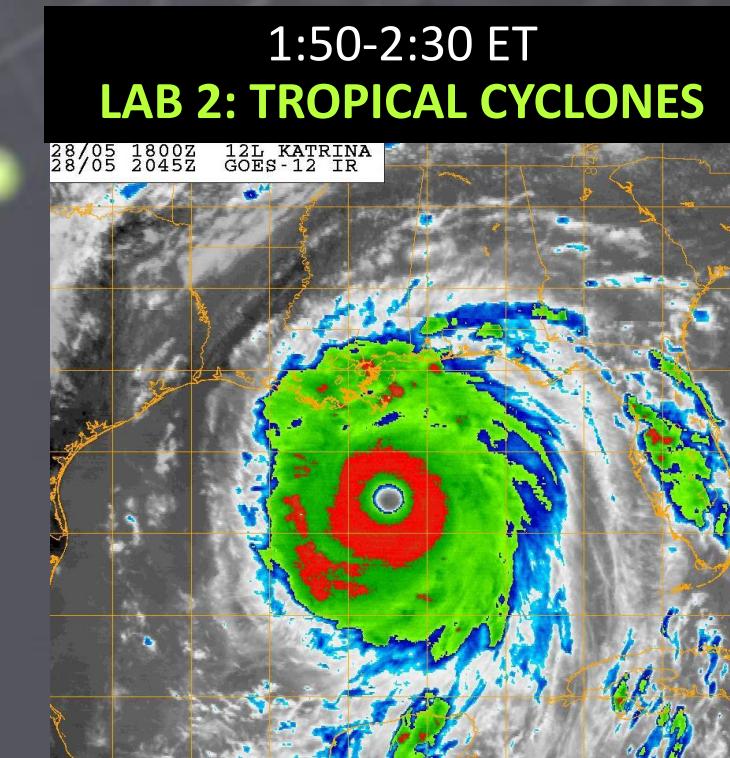
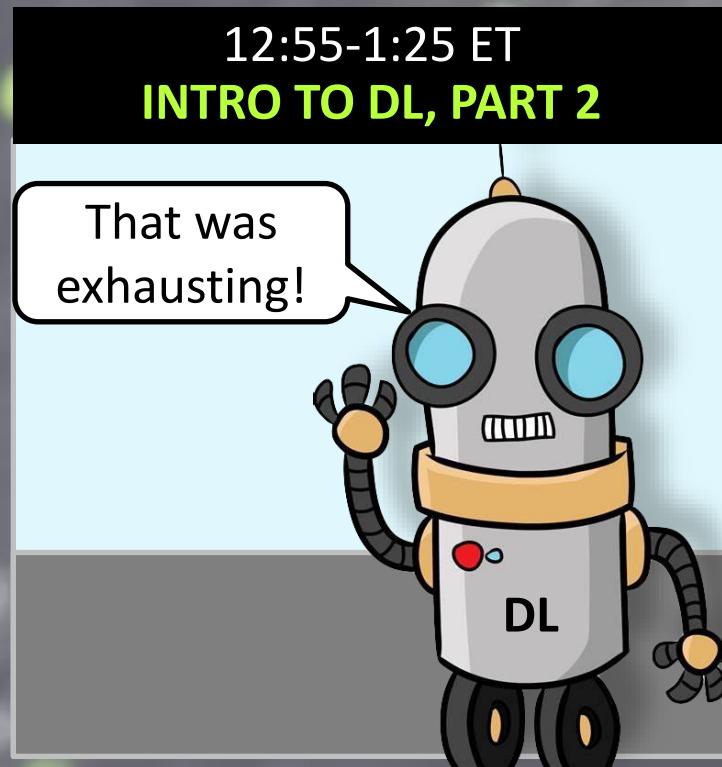
RESNET-50

2015 Microsoft Research. 50 Layers, 23M params.

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research





LAB PART 1

CNN AND KERAS 101

MNIST

The standard ‘hello world’ problem for deep learning



MNIST

Keras implementation



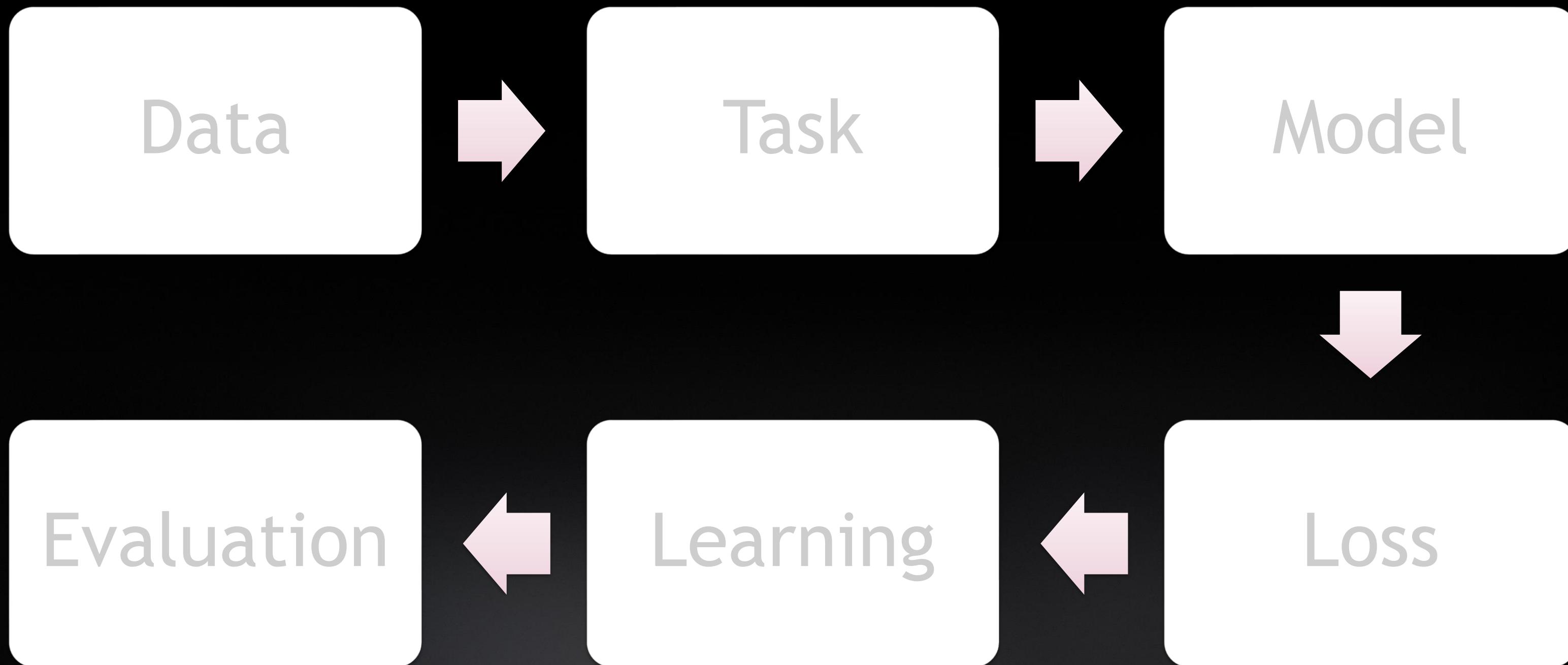
```
1 from tensorflow import keras
2 from tensorflow.keras.datasets import mnist
3 from tensorflow.keras.models import Sequential
4 from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
5 from tensorflow.keras import backend as K
6
7 num_classes = 10
8 img_rows, img_cols = 28, 28
9
10 # DATA
11 (x_train, y_train), (x_test, y_test) = mnist.load_data()
12
13 x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
14 x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
15 y_train = keras.utils.to_categorical(y_train, num_classes)
16 y_test = keras.utils.to_categorical(y_test, num_classes)
17
18 # MODEL
19 input_shape = (img_rows, img_cols, 1)
20 model = Sequential()
21 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
22 model.add(Conv2D(64, (3, 3), activation='relu'))
23 model.add(MaxPooling2D(pool_size=(2, 2)))
24 model.add(Dropout(0.25))
25 model.add(Flatten())
26 model.add(Dense(128, activation='relu'))
27 model.add(Dropout(0.5))
28 model.add(Dense(num_classes, activation='softmax'))
29 model.compile(loss=keras.losses.categorical_crossentropy,
30                 optimizer=keras.optimizers.Adadelta(),
31                 metrics=['accuracy'])
32
33 # TRAIN
34 model.fit(x_train, y_train, batch_size=128, epochs=12,
35             verbose=1, validation_data=(x_test, y_test))
36
37 # TEST
38 score = model.evaluate(x_test, y_test, verbose=0)
39 print('Test loss:', score[0])
40 print('Test accuracy:', score[1])
```

FASHION MNIST

A slightly more interesting version of MNIST

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

6 STEPS APPROACH



LAUNCH CNN PRIMER AND KERAS 101

12:30-1:00 ET

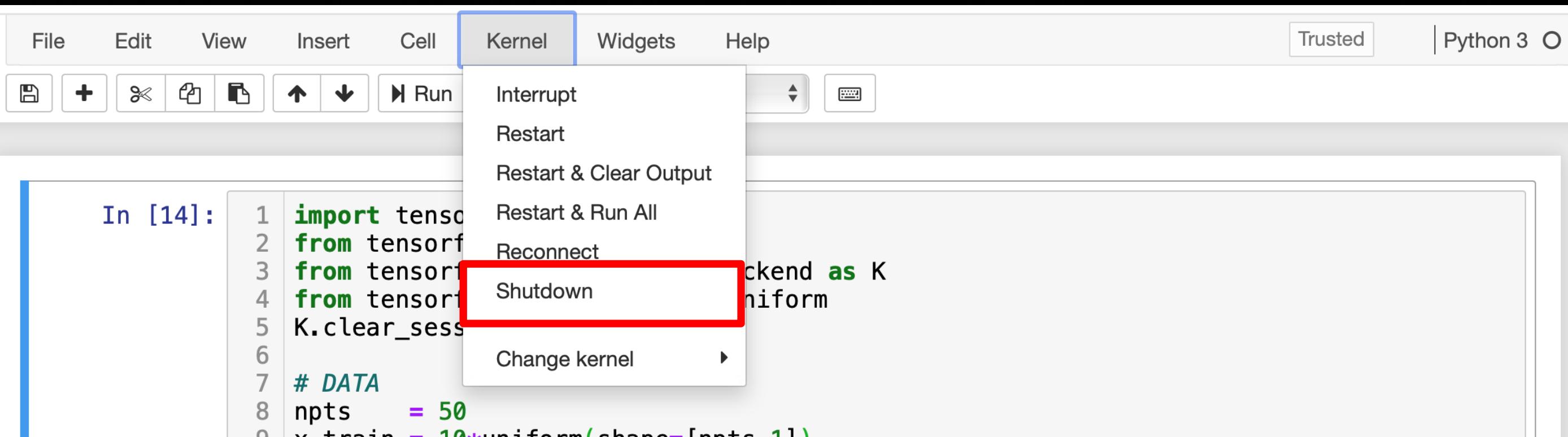
Step through the primer on your own (shift + enter on each cell)

The following contents will be covered during the Bootcamp :

- [CNN Primer and Keras 101 \(Intro to DL/Part 2.ipynb\)](#)
- [Tropical Cyclone Intensity Estimation using Deep Convolution Neural Networks.](#)

CLICK HERE

Shutdown the kernel before clicking on “Next Notebook” to free up the GPU memory



FORGOT TO SHUTDOWN YOUR KERNELS?

Don't worry, you can fix it.

`NotFoundError: Failed to get convolution algorithm. This is probably because cuDNN failed to initialize, so try looking to see if a warning log message was printed above.`

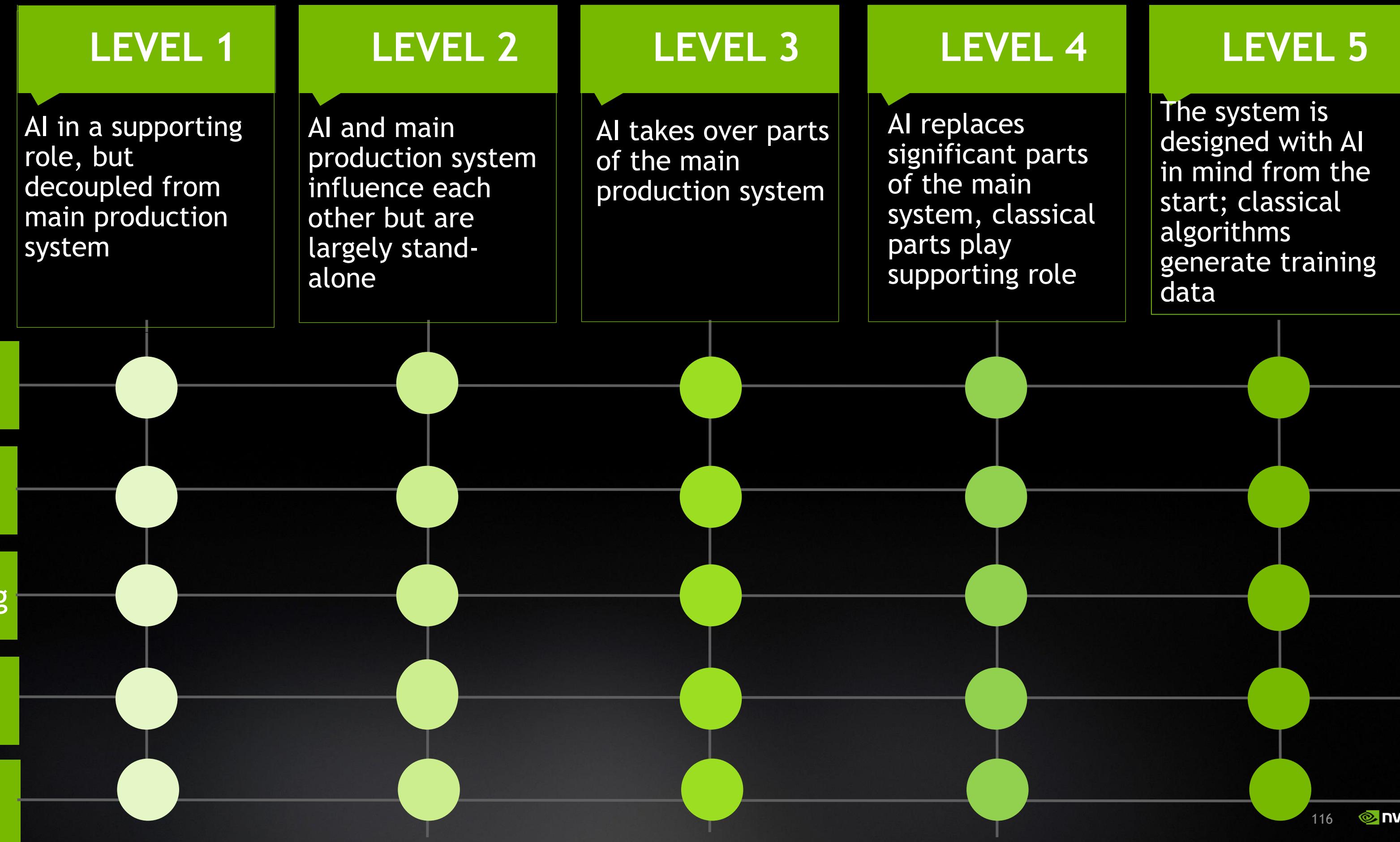
Go to Home Tab, Click Running Tab, Kill notebooks you aren't using

The screenshot shows the Jupyter Home interface. At the top, there are tabs for 'My Int' (closed), 'Home' (highlighted with a red box), 'Start...', 'Part_2...', and 'CNN's...'. Below the tabs is a toolbar with icons for back, forward, search, and a dropdown menu. The main area is titled 'jupyter' and contains three tabs: 'Files', 'Running' (highlighted with a red box), and 'Clusters'. Under 'Running', it says 'Currently running Jupyter processes' and 'There are no terminals running.' In the 'Notebooks' section, three entries are listed: '/Start_Here.ipynb' (Python 3), '/Intro_to_DL/Part_2.ipynb' (Python 3), and '/Intro_to_DL/CNN's.ipynb' (Python 3). Each entry has a 'Shutdown' button highlighted with a red box.

Restart & Clear Output on the Kernel you are using

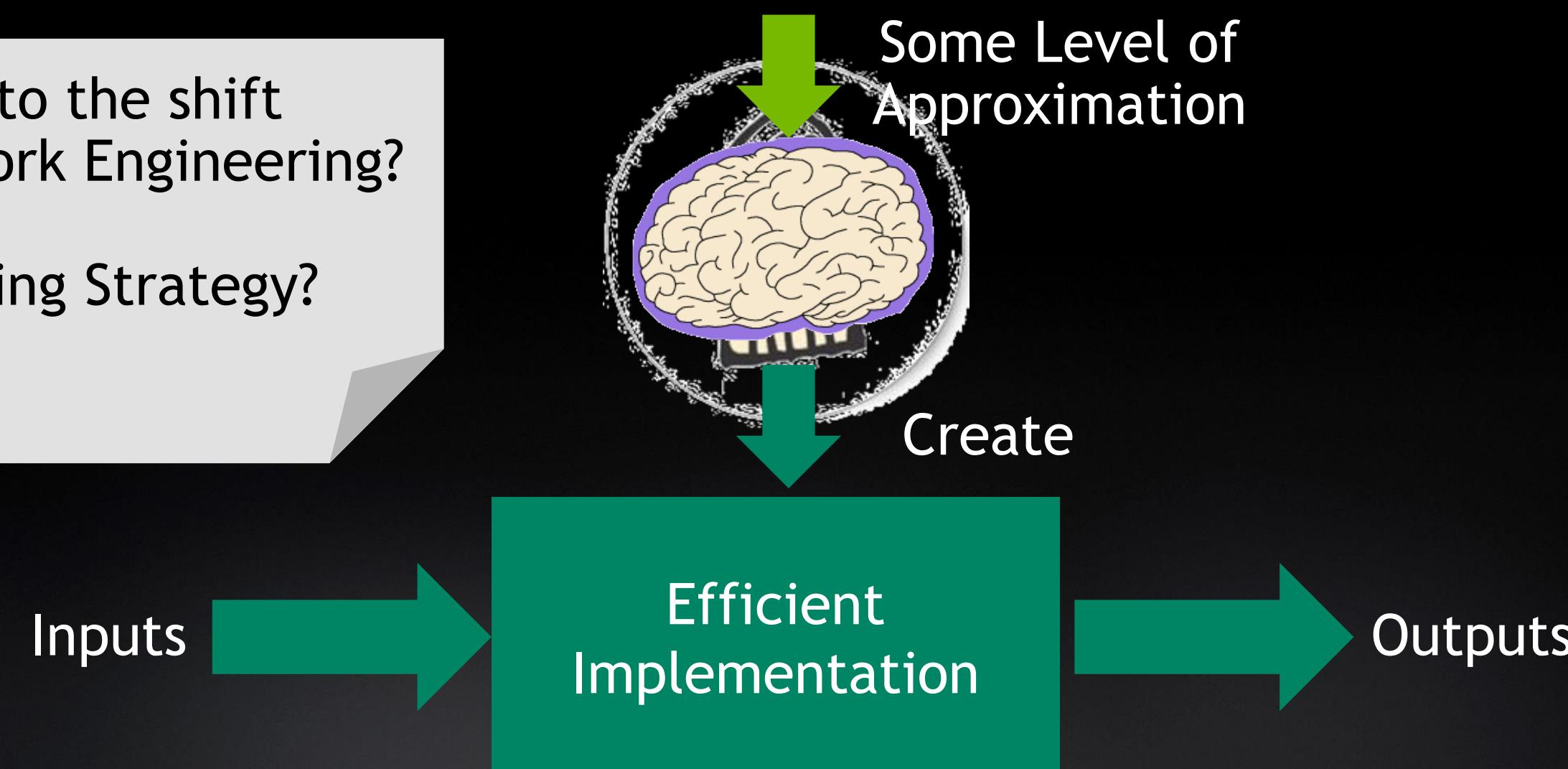
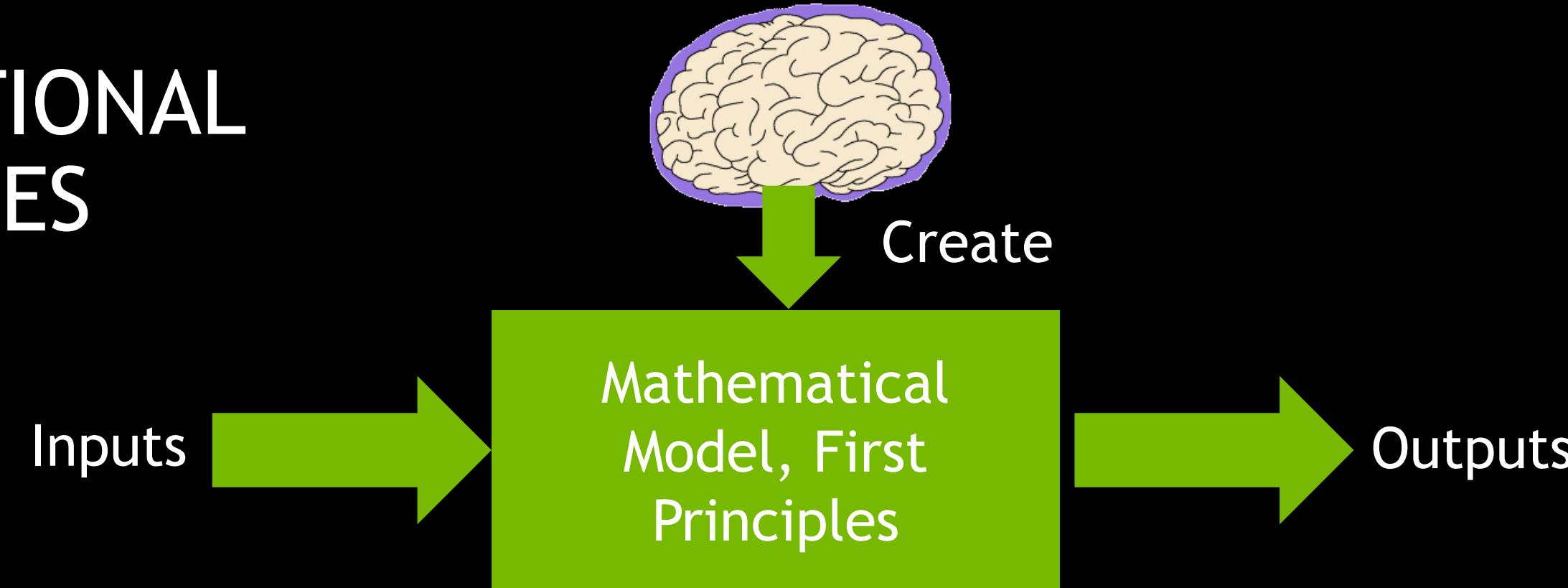
The screenshot shows a Jupyter notebook titled 'CNN Primer'. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel' (which is currently selected and highlighted with a red box), and 'Help'. A dropdown menu from the 'Kernel' tab lists several options: 'Interrupt', 'Restart', 'Restart & Clear Output' (highlighted with a red box), 'Restart & Run All', 'Reconnect', 'Shutdown', and 'Change kernel'. The notebook content area displays the text: 'This notebook covers introduction to Convolutional Ne terminologies.'

LEVELS OF AI ENGAGEMENT



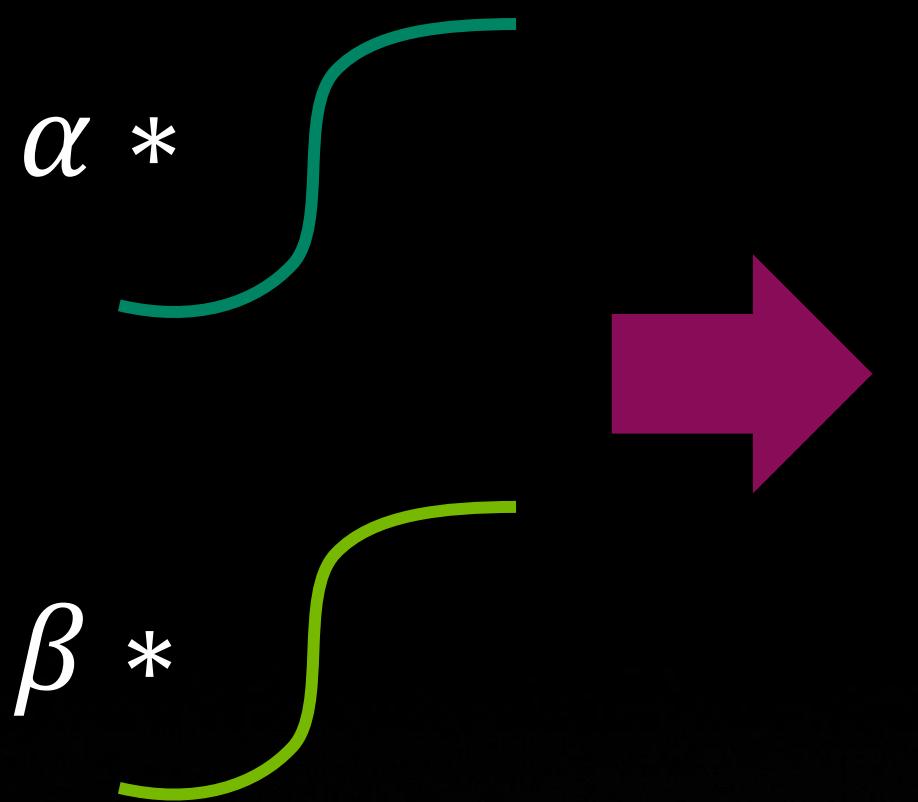
COMPUTATIONAL SCIENCES

Similarities to the shift
Feature → Network Engineering?
NNs as a Porting Strategy?



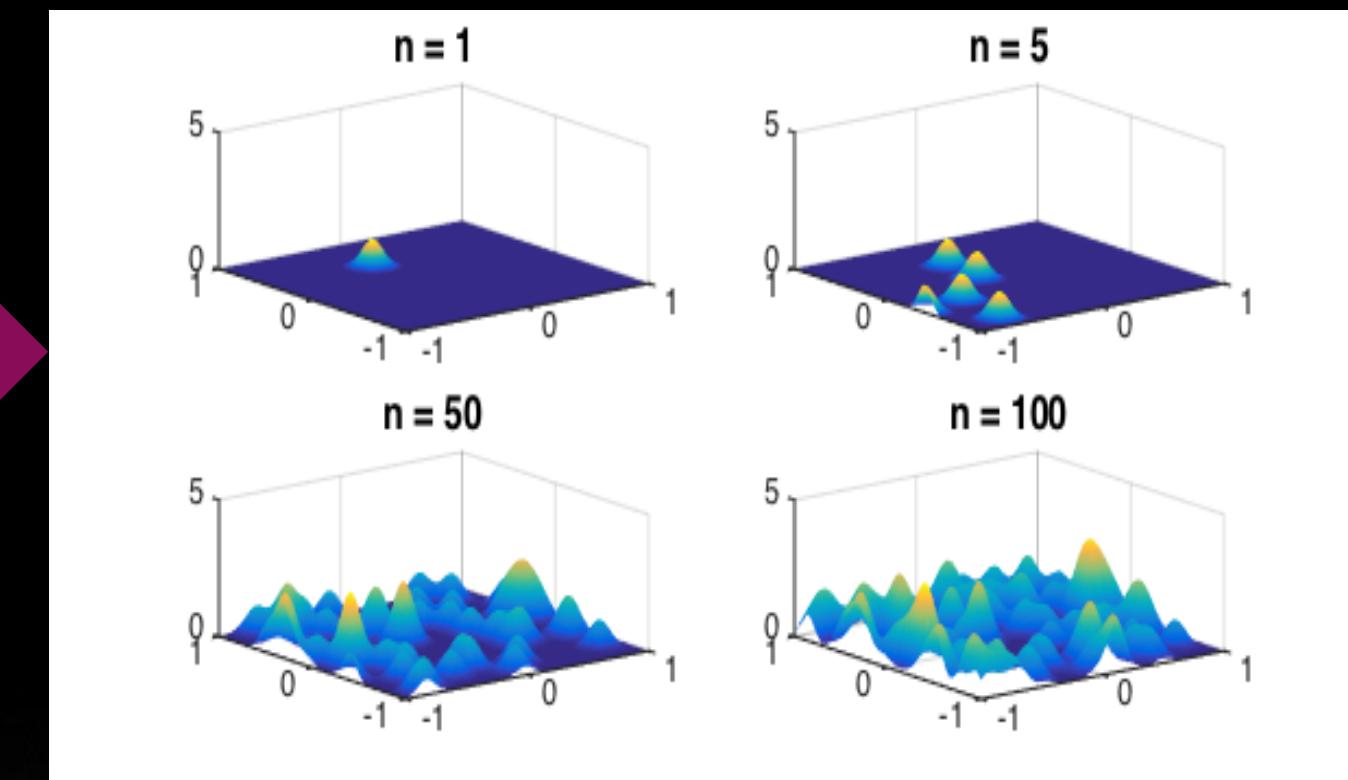
CAN THIS WORK ∀? ABSOLUTELY, YES!

Proof: Universal Approximation Theorem



Take many non-linearities

Combine to form peaks
(one hidden layer is enough!)



And assemble your arbitrary function with arbitrary ε

Problem: this is an essentially useless theorem for practical purposes

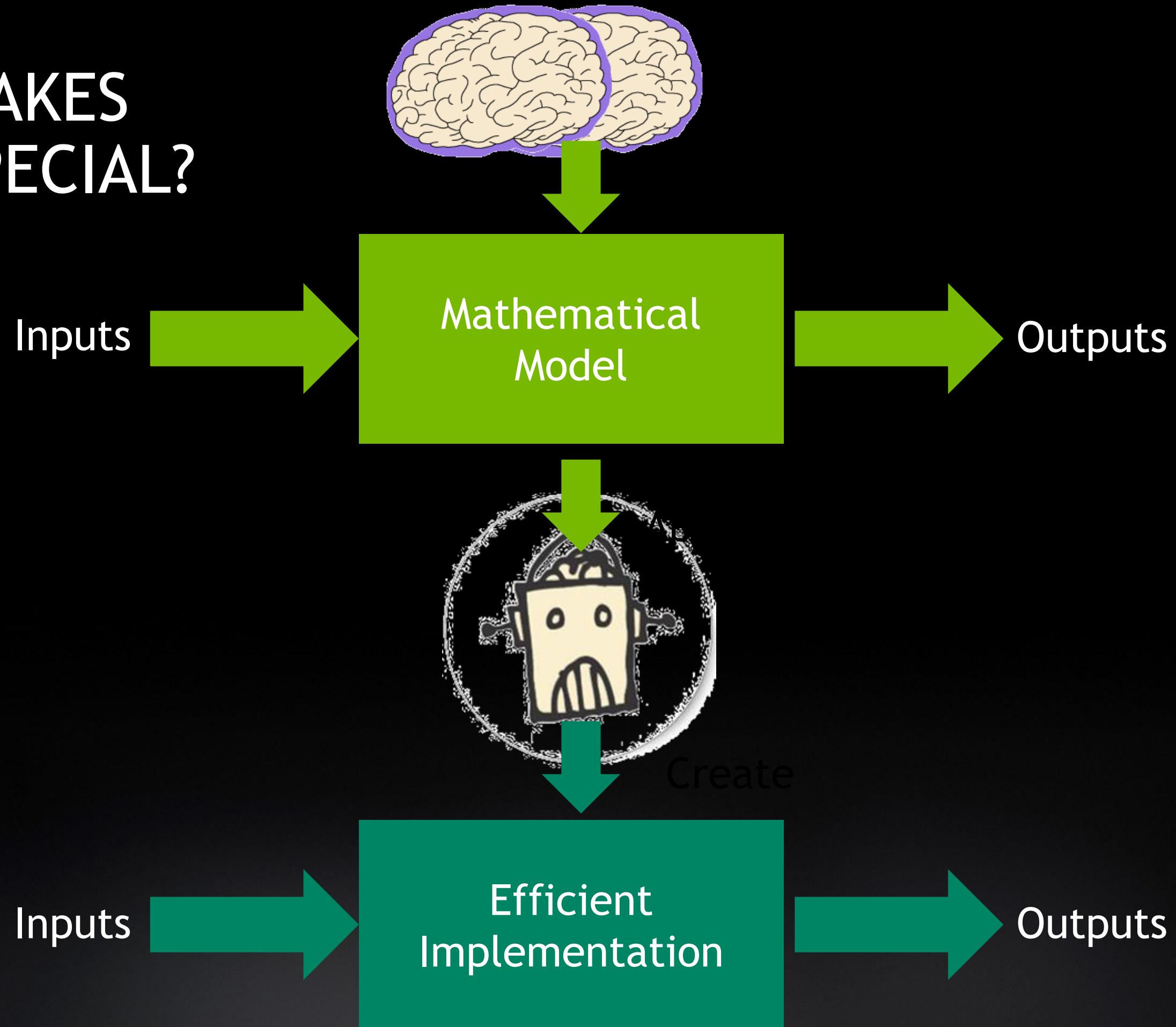
WILL THIS WORK \forall ?

Considering pesky practical constraints, like memory and performance

- Anecdotal Evidence: \exists scientific cases where NNs seem to do work extremely well
- Safe bet: it will not work for \forall
- Therefore, by induction (sort of):
 - There exists \exists a subspace in \forall HPC applications, for which AI works well
 - Need to explore the **size** and **shape** of this subspace
 - **Currently I think it is fair to say we don't understand this domain very well**
 - **But:** Each individual case promising 10x, 100x, 1000x performance improvement is probably worth exploring; those can be groundbreaking!

But Intuition is Misleading

WHAT MAKES AI * HPC SPECIAL?



HOW TO FILL IN THE



Experience,
Intuition, and Art



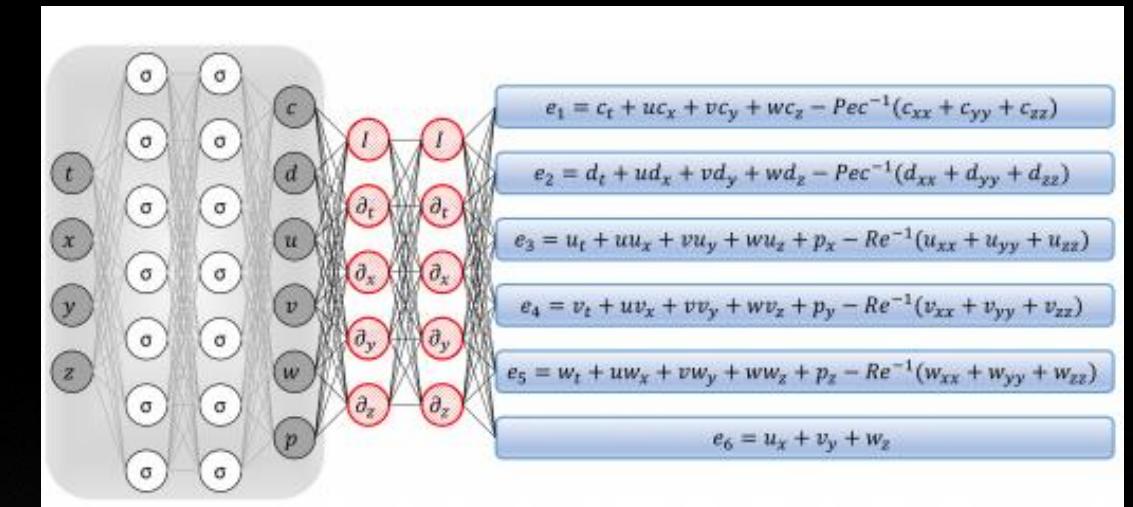
+ Tools Support



E.g. Adversarial Fuzzing



E.g. Declarative Building
Blocks to NN Translation



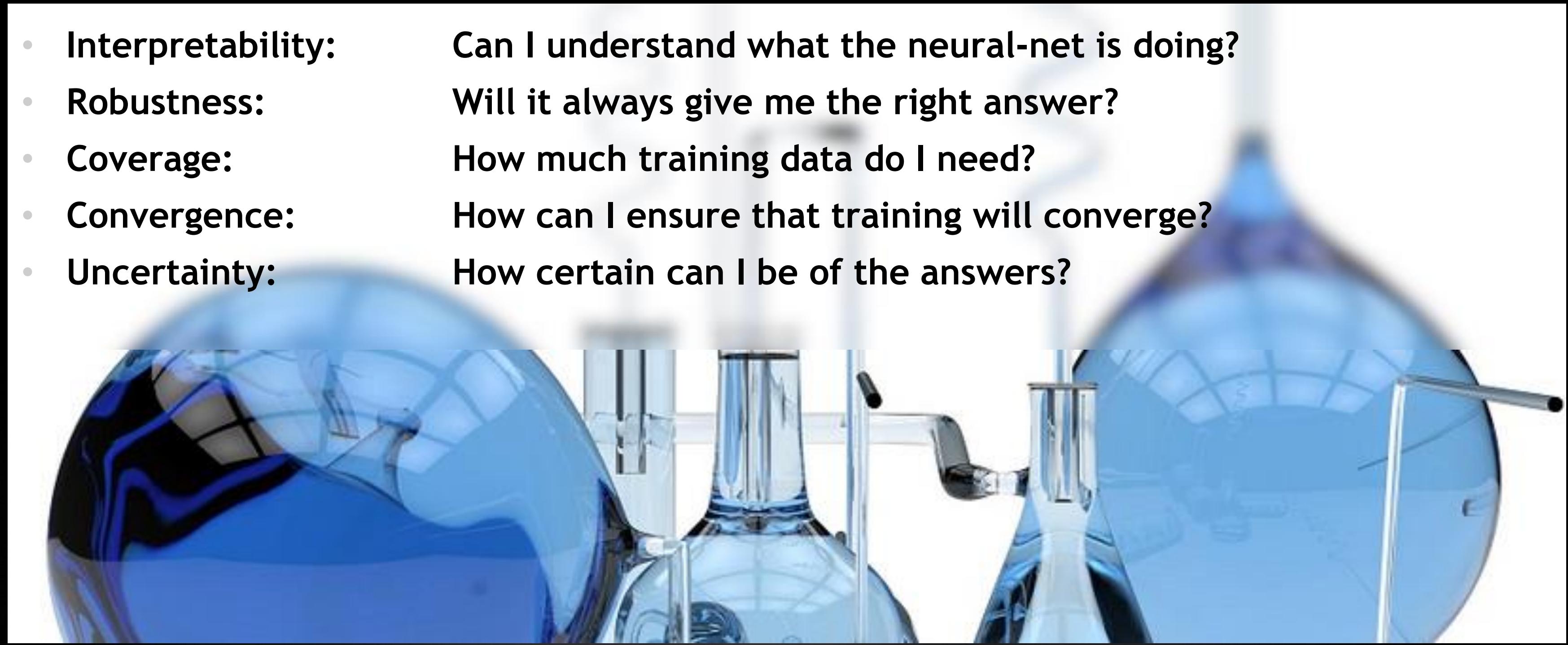
E.g. Physics Informed
Networks¹⁾,
ODE Networks²⁾

- 1) Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework, M. Raissi et al.
- 2) Neural Ordinary Differential Equations, R.T.Q. Chen et al.

SCIENTIFIC CHALLENGES

Barriers to acceptance of deep learning as a tool for science

- **Interpretability:** Can I understand what the neural-net is doing?
- **Robustness:** Will it always give me the right answer?
- **Coverage:** How much training data do I need?
- **Convergence:** How can I ensure that training will converge?
- **Uncertainty:** How certain can I be of the answers?

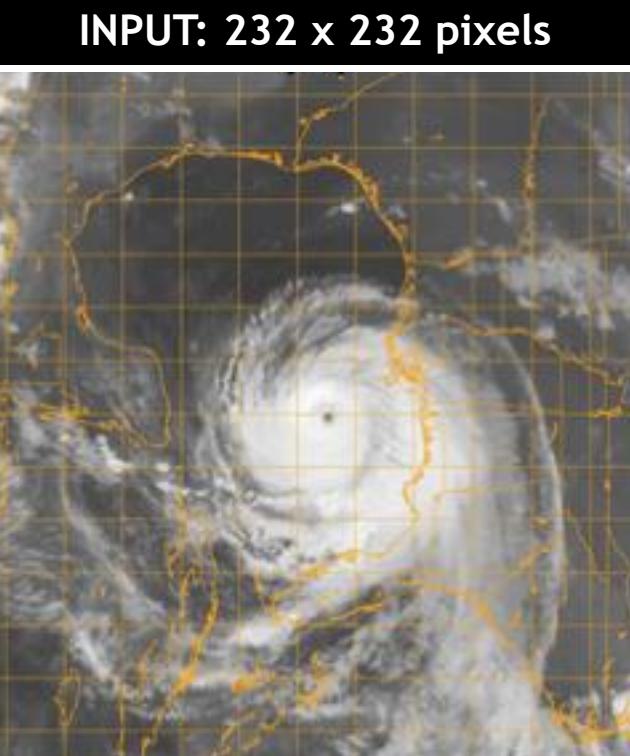


ESTIMATING TROPICAL CYCLONE INTENSITY

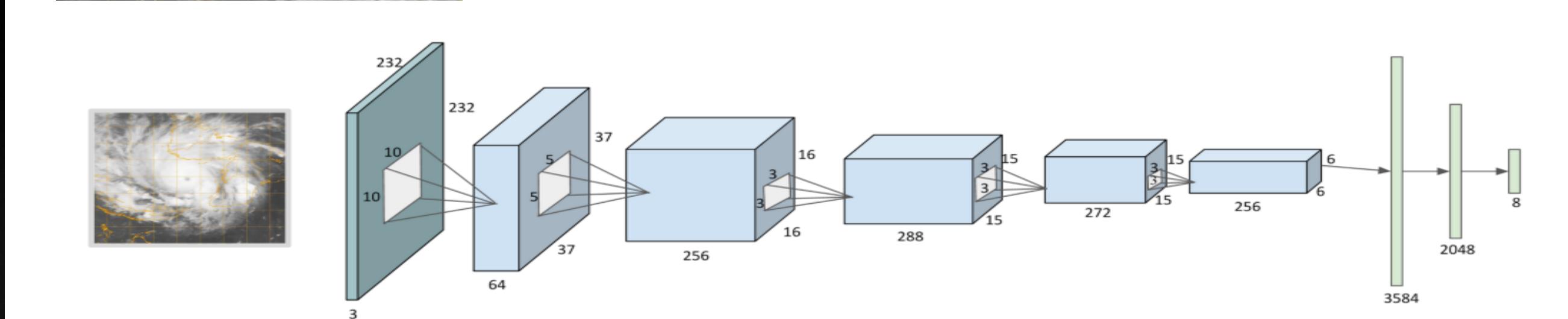
Paper Overview

Tropical Cyclone Intensity Estimation Using a Deep Convolutional Neural Network

Ritesh Pradhan, Ramazan Aygun, Senior Member, IEEE, Manil Maskey, Member, IEEE, Rahul Ramachandran, Senior Member, IEEE, and Daniel Cecil



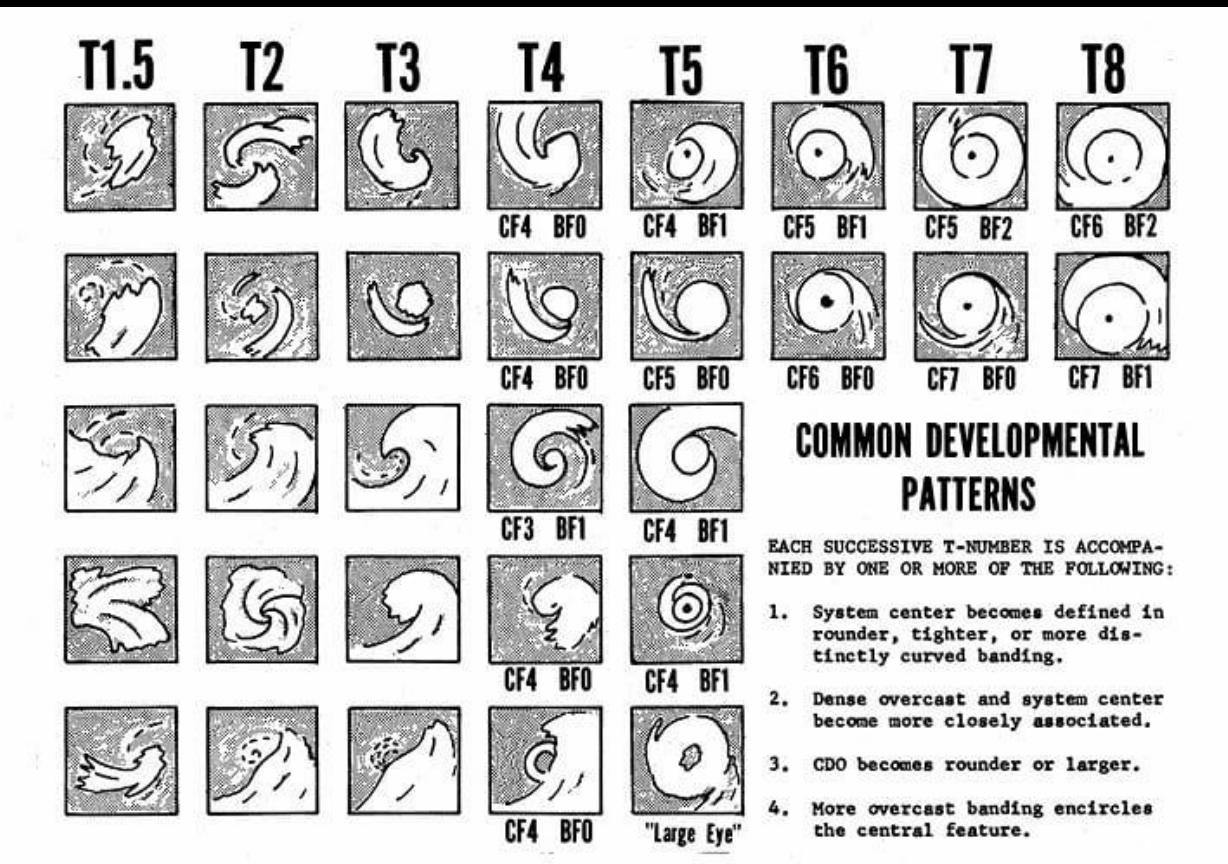
OUTPUT: 8 CLASSES			
Category	Symbol	Wind speeds	Damage
Five	H5	≥ 137 knots	Catastrophic
Four	H4	113- 136 knots	Catastrophic
Three	H3	96- 112 knots	Devastating
Two	H2	83- 95 knots	Extensive
One	H1	64- 82 knots	Significant
Tropical storm	TS	34- 63 knots	Significant
Tropical depression	TD	20- 33 knots	Small
No Category	NC	≤ 20 knots	-



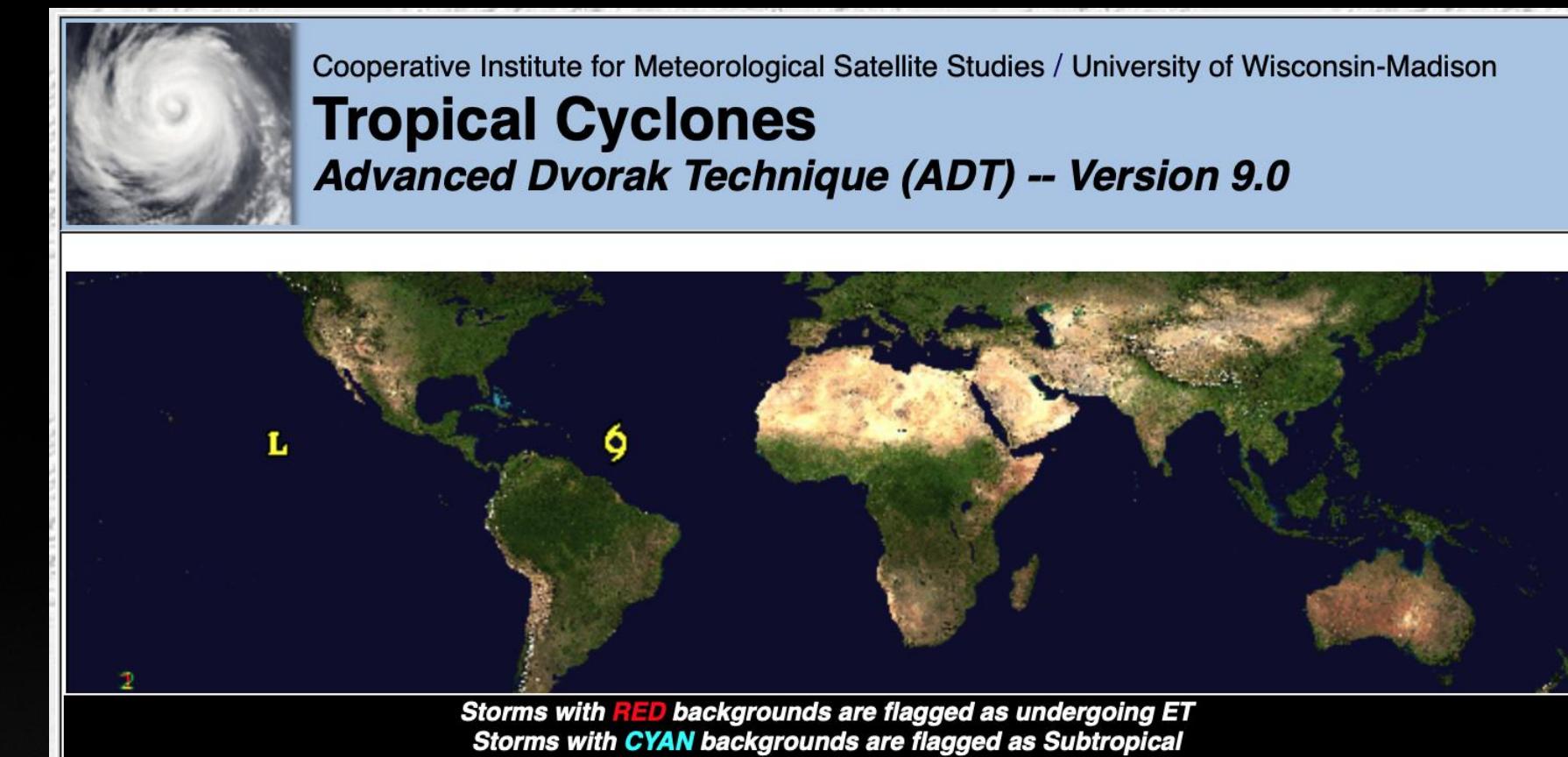
ESTIMATING TROPICAL CYCLONE INTENSITY

Background: Dvorak technique

Dvorak Technique (1974)



Advanced Dvorak Technique- version 9 (2019)

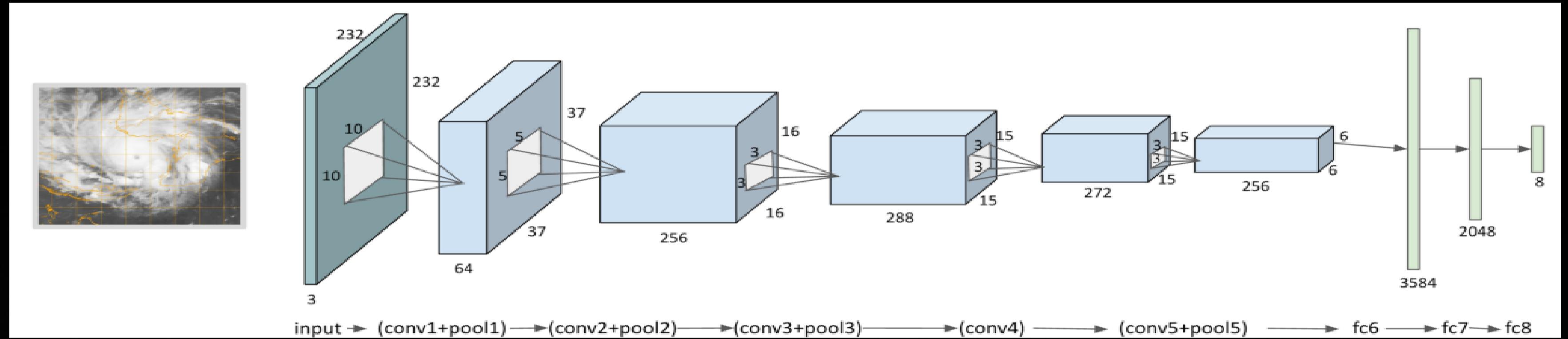


[https://doi.org/10.1175/1520-0493\(1975\)103%3C0420:TCIAAF%3E2.0.CO;2](https://doi.org/10.1175/1520-0493(1975)103%3C0420:TCIAAF%3E2.0.CO;2)

<https://doi.org/10.1175/WAF-D-19-0007.1>

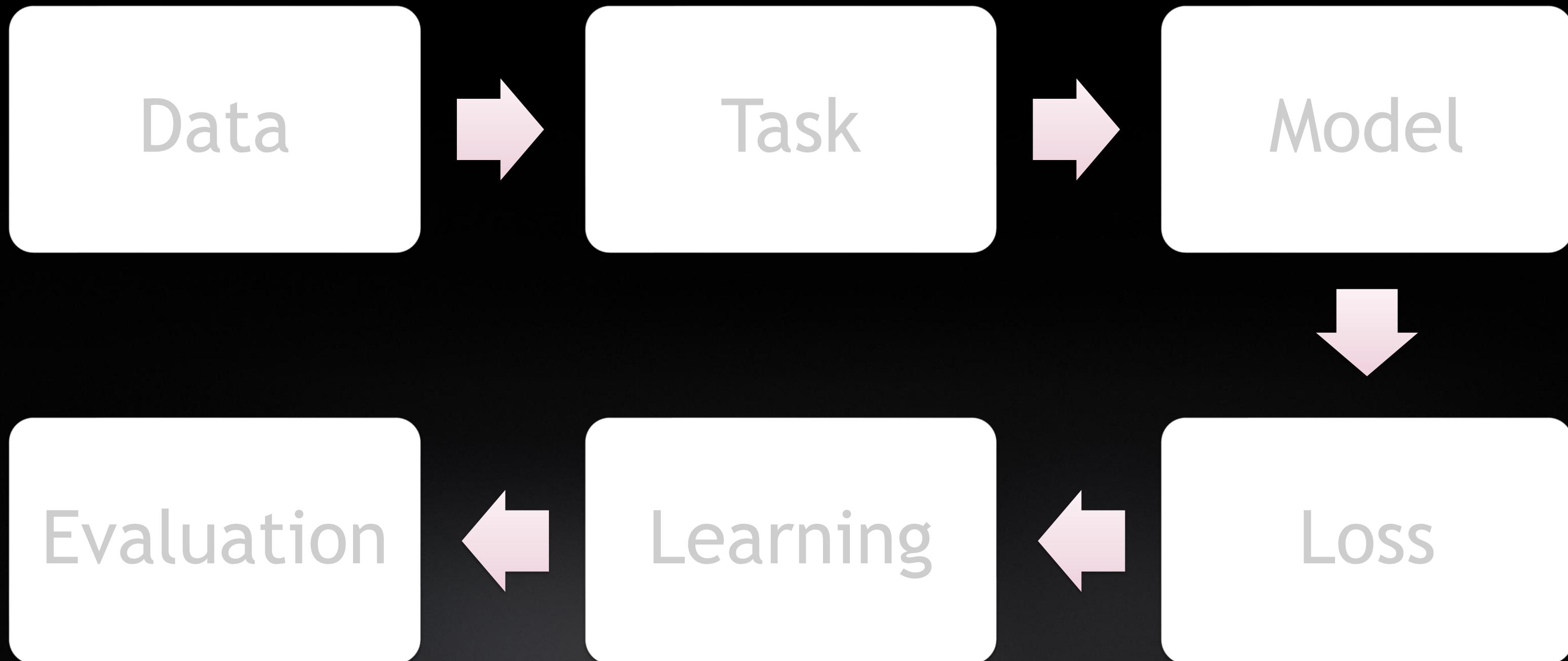
ESTIMATING TROPICAL CYCLONE INTENSITY

CNN Model

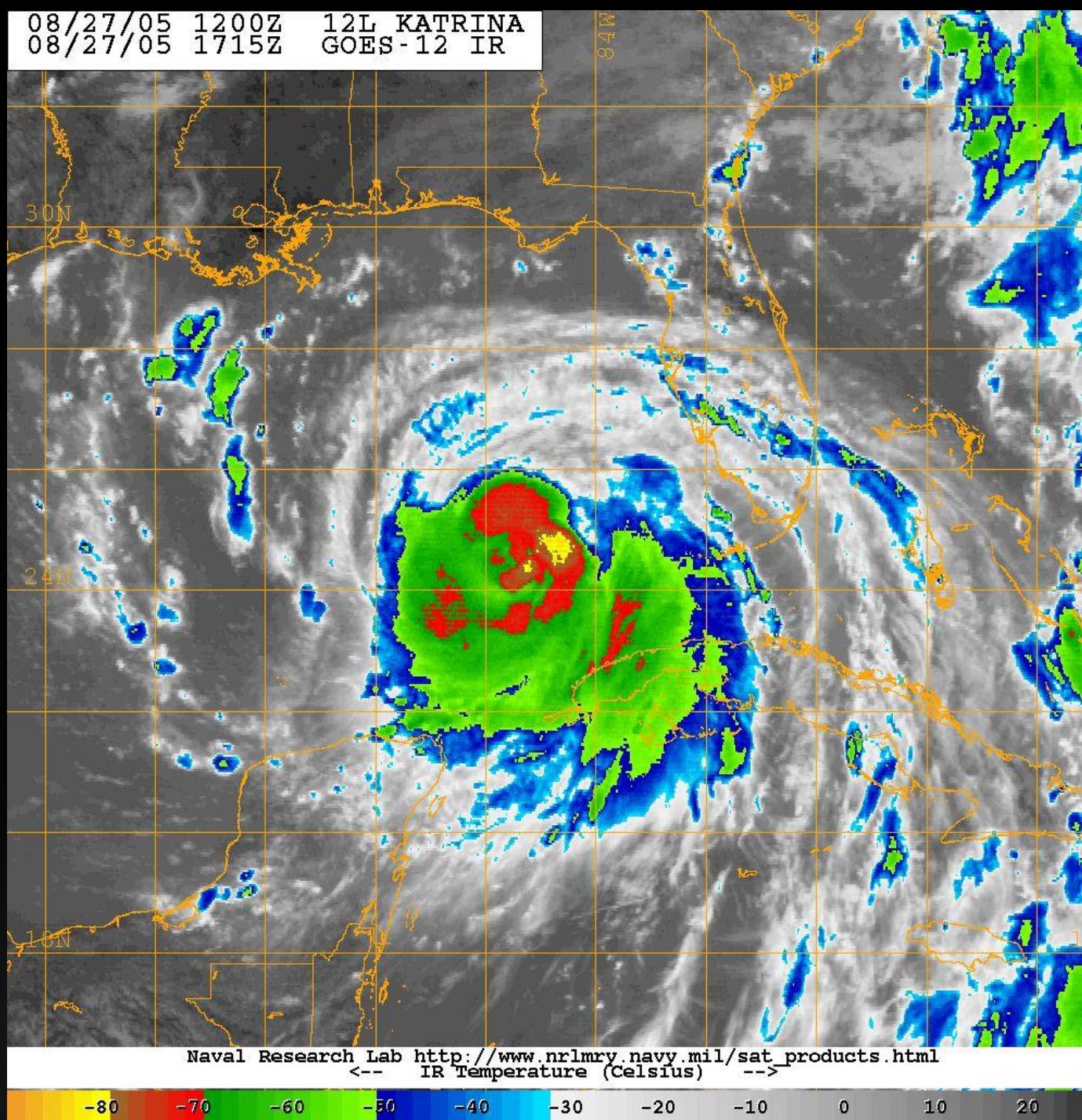


6 STEPS APPROACH

Steps to follow while solving a Machine Learning problem



DATA



SAFFIR-SIMPSON HURRICANE WIND SCALE
AND RELATED CLASSIFICATIONS

Category	Symbol	Wind speeds	Damage
Five	H5	≥ 137 knots	Catastrophic
Four	H4	113–136 knots	Catastrophic
Three	H3	96–112 knots	Devastating
Two	H2	83–95 knots	Extensive
One	H1	64–82 knots	Significant
Tropical storm	TS	34–63 knots	Significant
Tropical depression	TD	20–33 knots	Small
No Category	NC	≤ 20 knots	-

TASK

Multi-class Classification.

NC (No Category , ≤ 20 knots)

TD (Tropical Depression , 20-33 knots)

TS (Topical Storm , 34-63 knots)

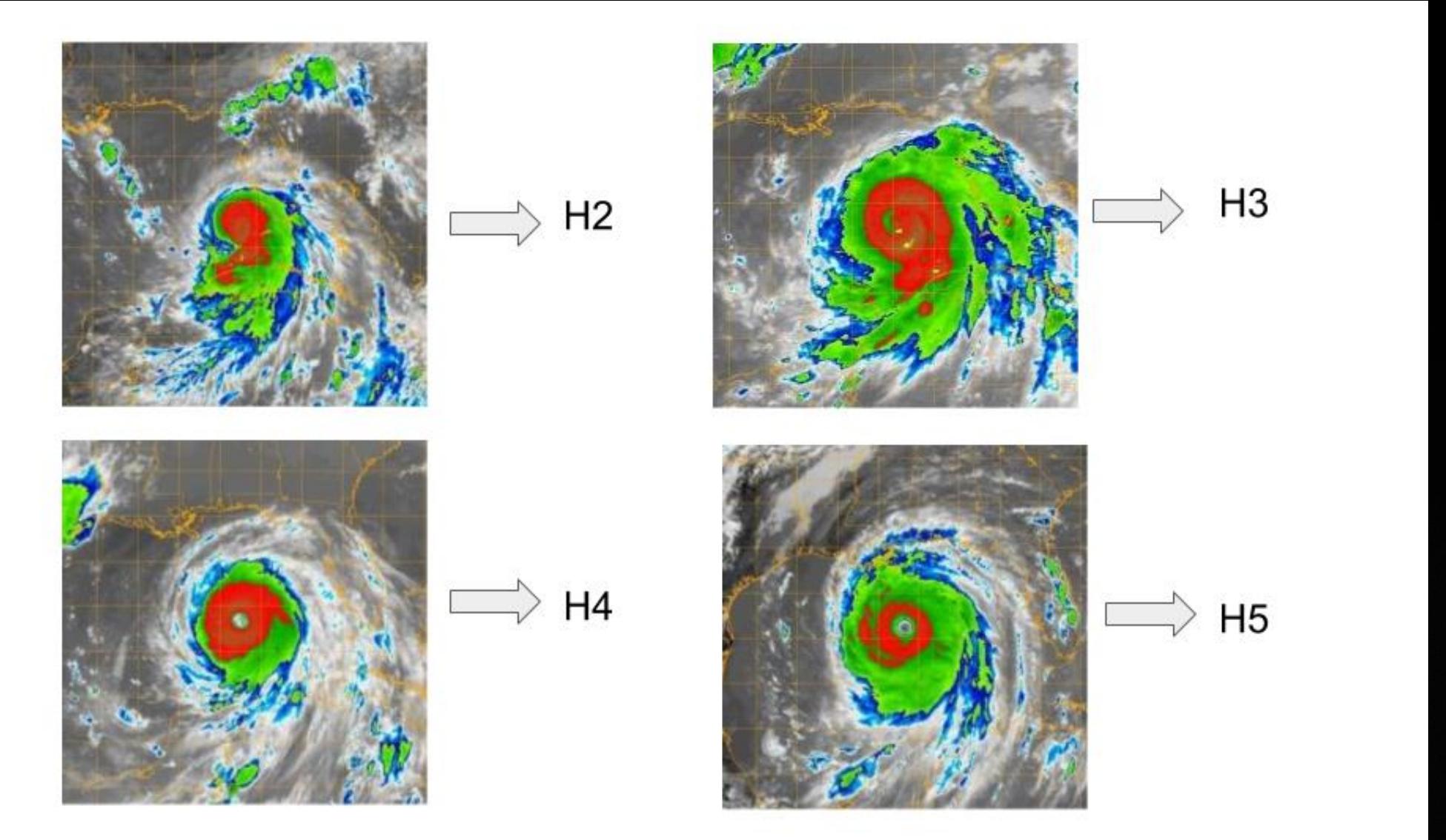
H1 (Category One , 64-82 knots)

H2 (Category Two , 83-95 knots)

H3 (Category Three , 96-112 knots)

H4 (Category Four , 113-136 knots)

H5 (Category Five , ≥ 137 knots)



MODEL 1

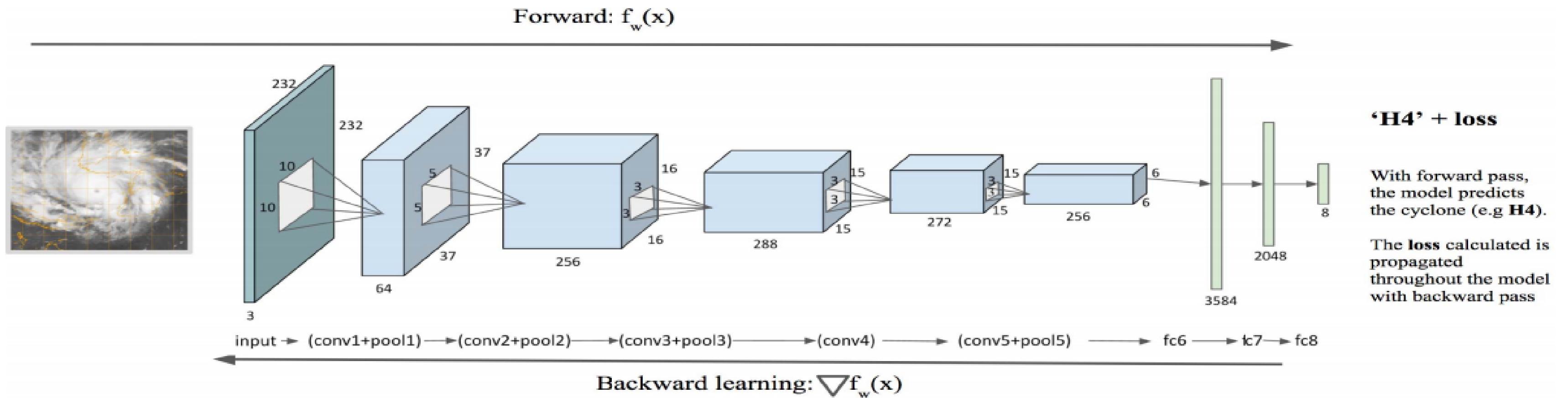


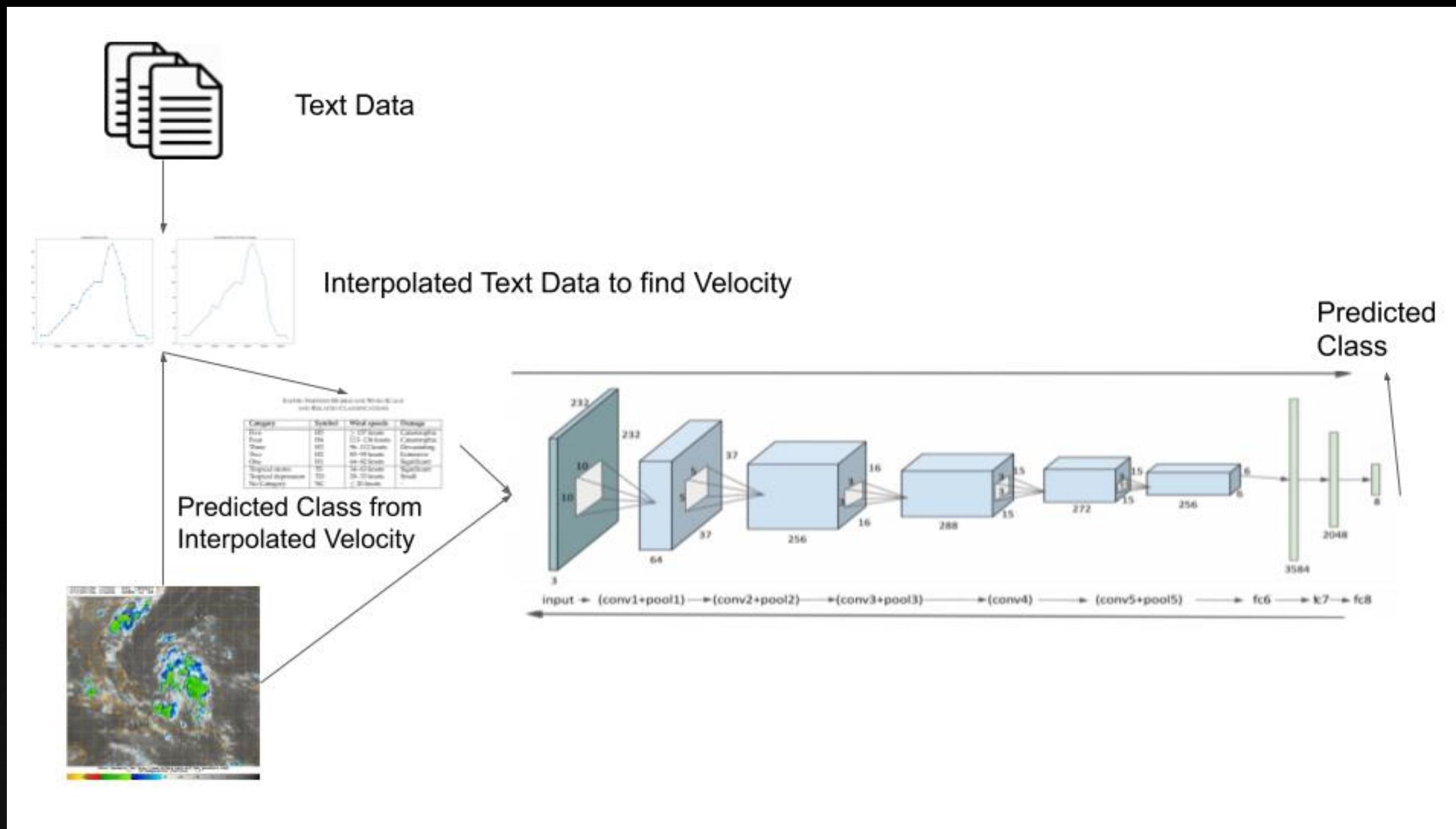
Fig. 2. Network architecture for hurricane intensity estimation showing different steps of convolution and pooling.

Loss Function: Multi-class Cross-Entropy loss functions

Optimizer SGD (Stochastic Gradient Descent)

Training and Evaluation: Training Set 72 % , Test Set, 8 %, Validation Set 10%

SUMMARY OF APPROACH



PREPROCESSING DATA

Pre-Processing Data:

Step 1 : Resize Image from (1024, 1024 ,3) to (256 , 256 ,3)

Step 2 : Choose a random (232 , 232 , 3) patch from the (256 , 256 , 3) and feed into our model.

There are different types of Resizing:

- cv2.INTER_AREA (Preferable for Shrinking)
- cv2.INTER_CUBIC (Preferable for Zooming but slow)
- cv2.INTER_LINEAR (Preferable for Zooming and the default option)

LAUNCH TROPICAL CYCLONE NOTEBOOK

1:00-2:00 ET

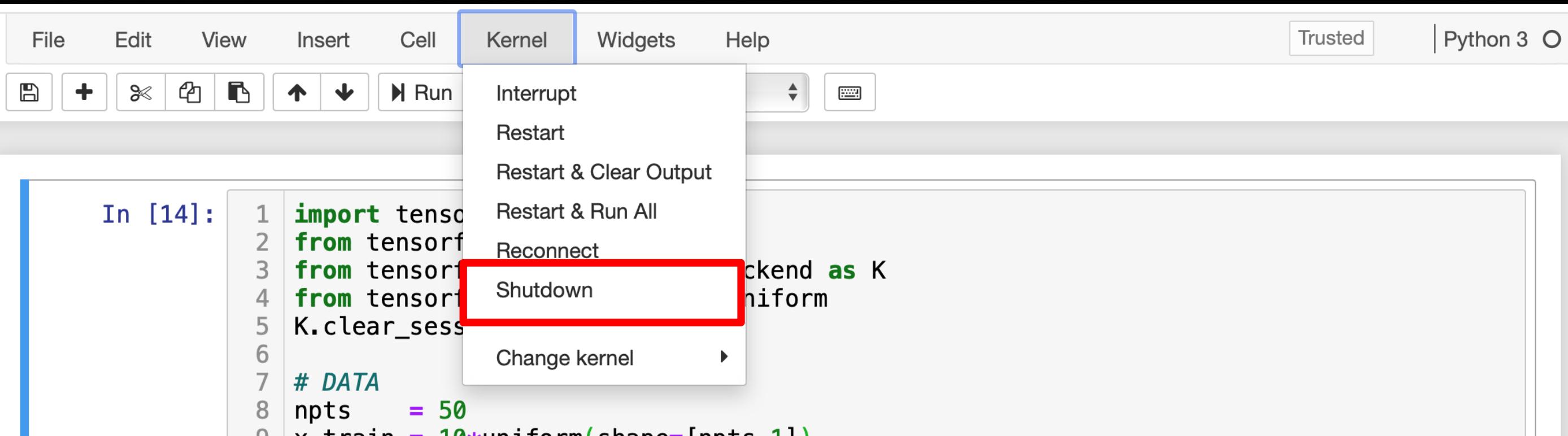
Step through the notebook on your own (shift + enter on each cell)

The following contents will be covered during the Bootcamp :

- CNN Primer and Keras 101 (Intro to DL/Part 2.ipynb)
- Tropical Cyclone Intensity Estimation using Deep Convolution Neural Networks.

CLICK HERE

Shutdown the kernel before clicking on “Next Notebook” to free up the GPU memory



TROPICAL CYCLONE COMPETITION

Can you make a better prediction?

Go to last notebook in the TC section

The following contents will be covered during the Bootcamp :

- CNN Primer and Keras 101 (Intro to DL/Part 2.ipynb)
- Tropical Cyclone Intensity Estimation using Deep Convolution Neural Networks.

See if you can improve the accuracy.

Suggestions:

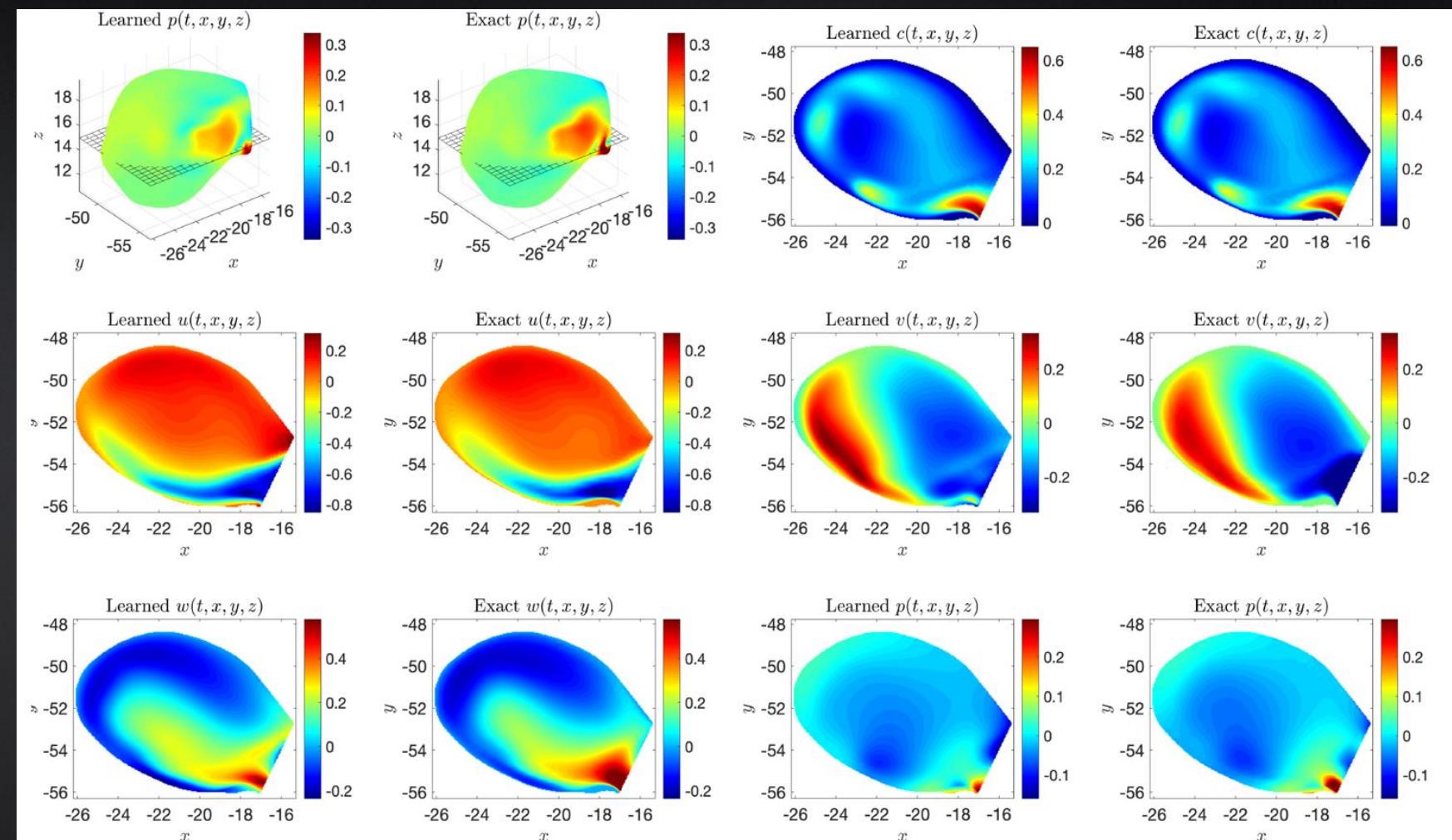
- Improve the data balance
- Tweak the model hyperparameters
- Try different optimizers

Bug in the lab: when doing the test/train split on time-series data, the data should not be shuffled!

- Try maximizing validation accuracy on both shuffled and un-shuffled validation sets

STEADY STATE FLOW WITH NEURAL NETWORKS

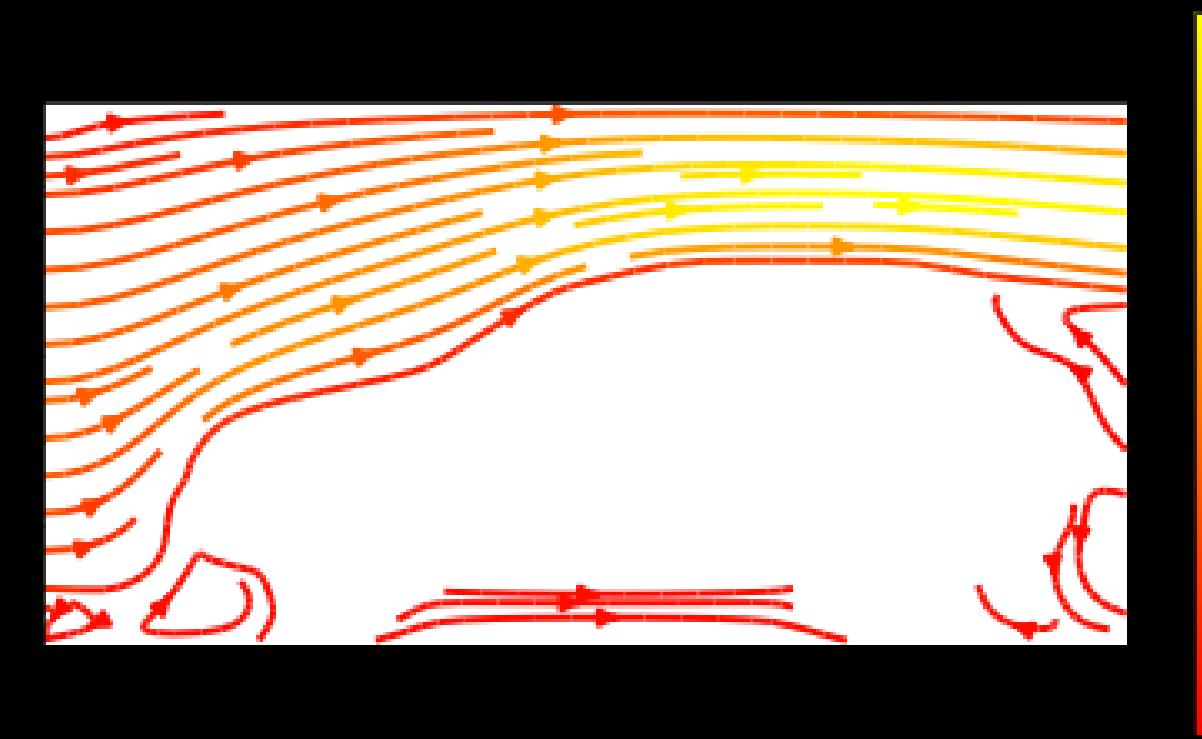
Flow fields are simulated using computational fluid dynamics (CFD) solvers



- CFD simulation is usually a computationally expensive, memory demanding and time-consuming iterative process
- *CFD limit opportunities for design space exploration and forbid interactive design*

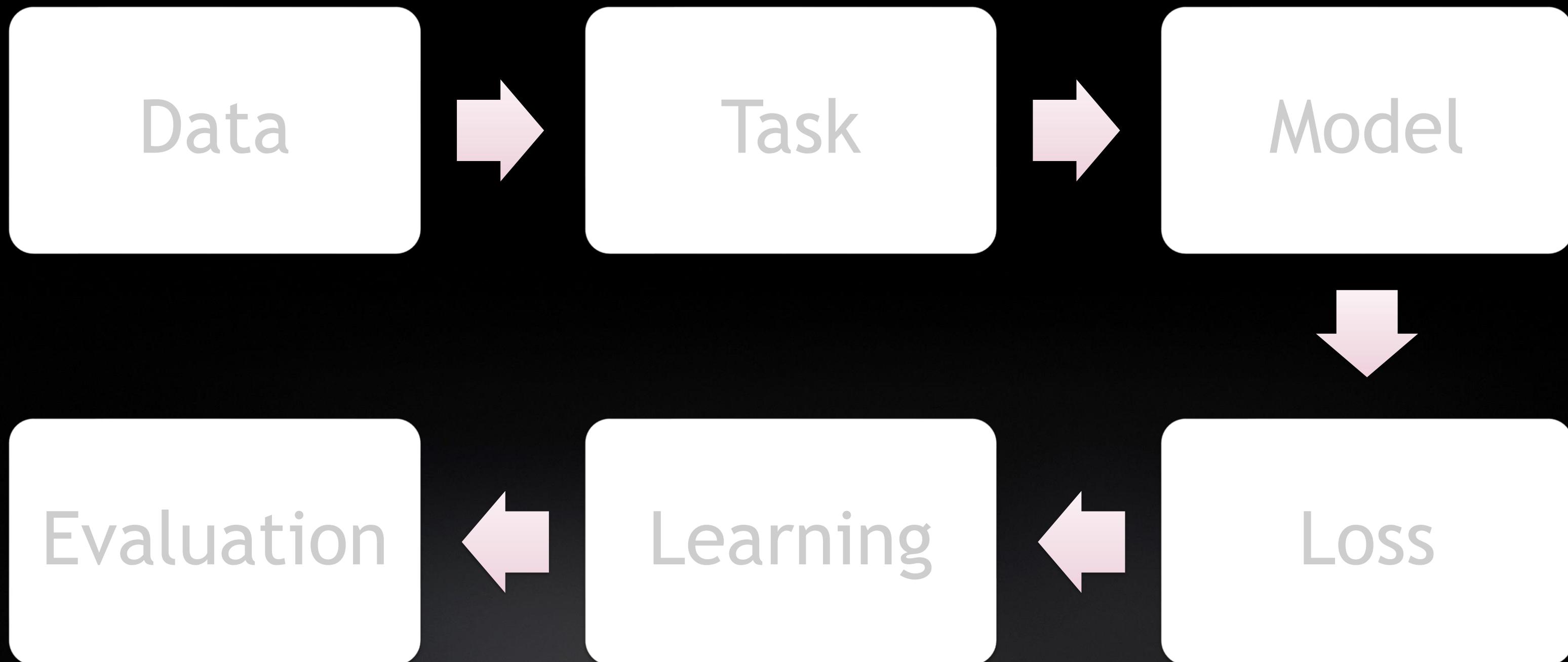
STEADY STATE FLOW WITH NEURAL NETWORKS

Our aim is to predict 2D flow around objects. The input is the boundary around which we want to calculate the flow. Here is an example of input data and the corresponding flow that was calculated using the Lattice Boltzmann method. ([Mechsys](#)).

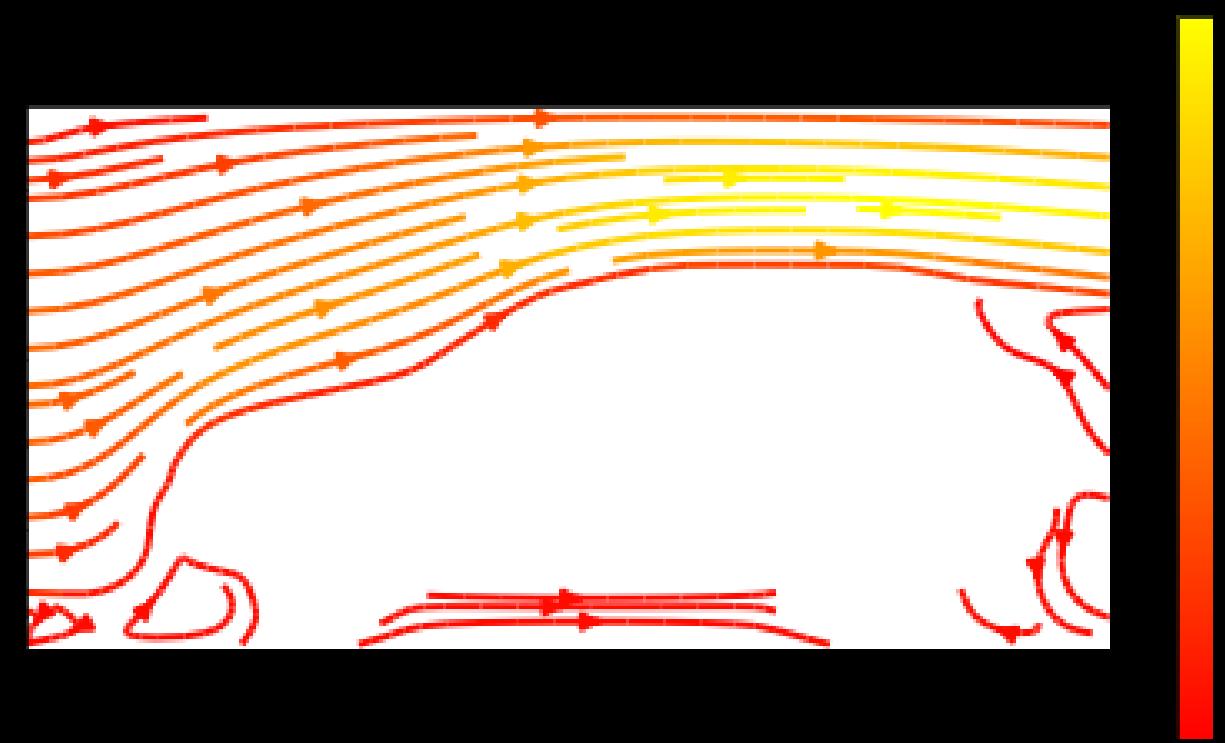


6 STEPS APPROACH

Steps to follow while solving a Machine Learning problem



DATA AND TASK



Predict the velocity vectors of both the x and y channels from our model.

MODEL

We will be building the following Models and benchmarking them as we proceed :

- Simple Fully Connected Networks

3 Layer Network

5 Layer Network

- Convolution Neural Networks

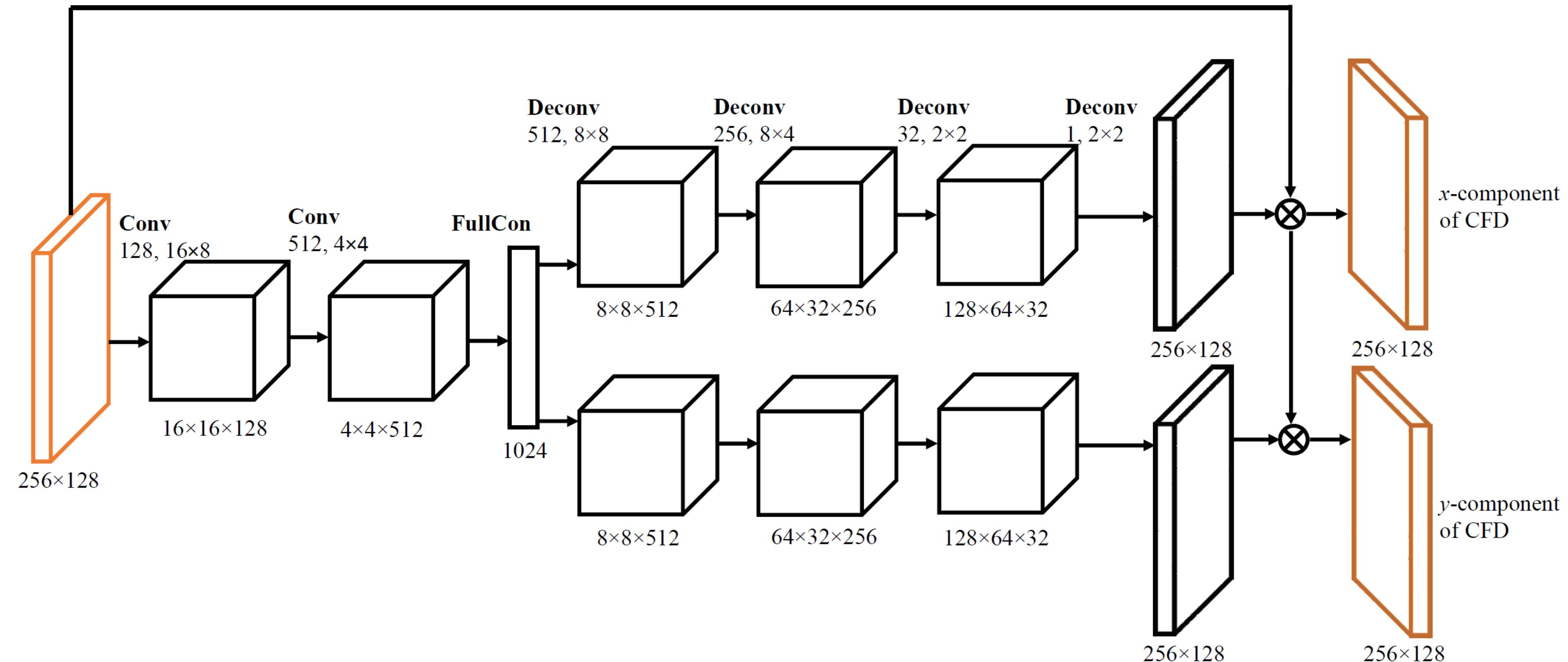
Binary Boundary

Signed Distance Function

- Advanced Networks

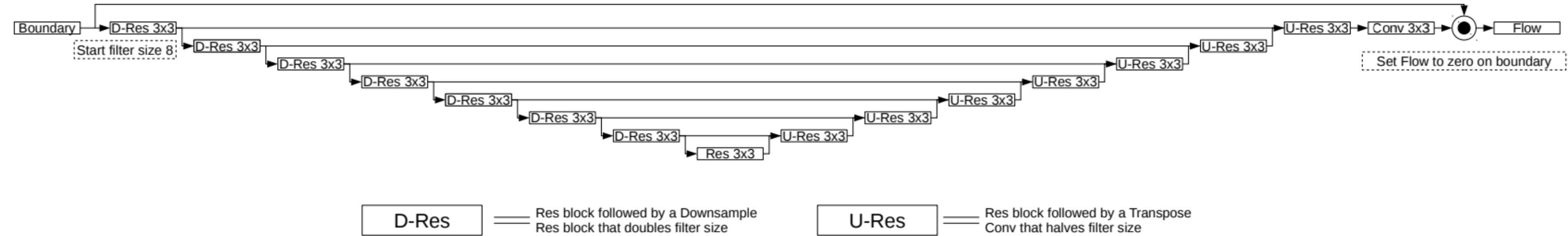
Gated Residual Network

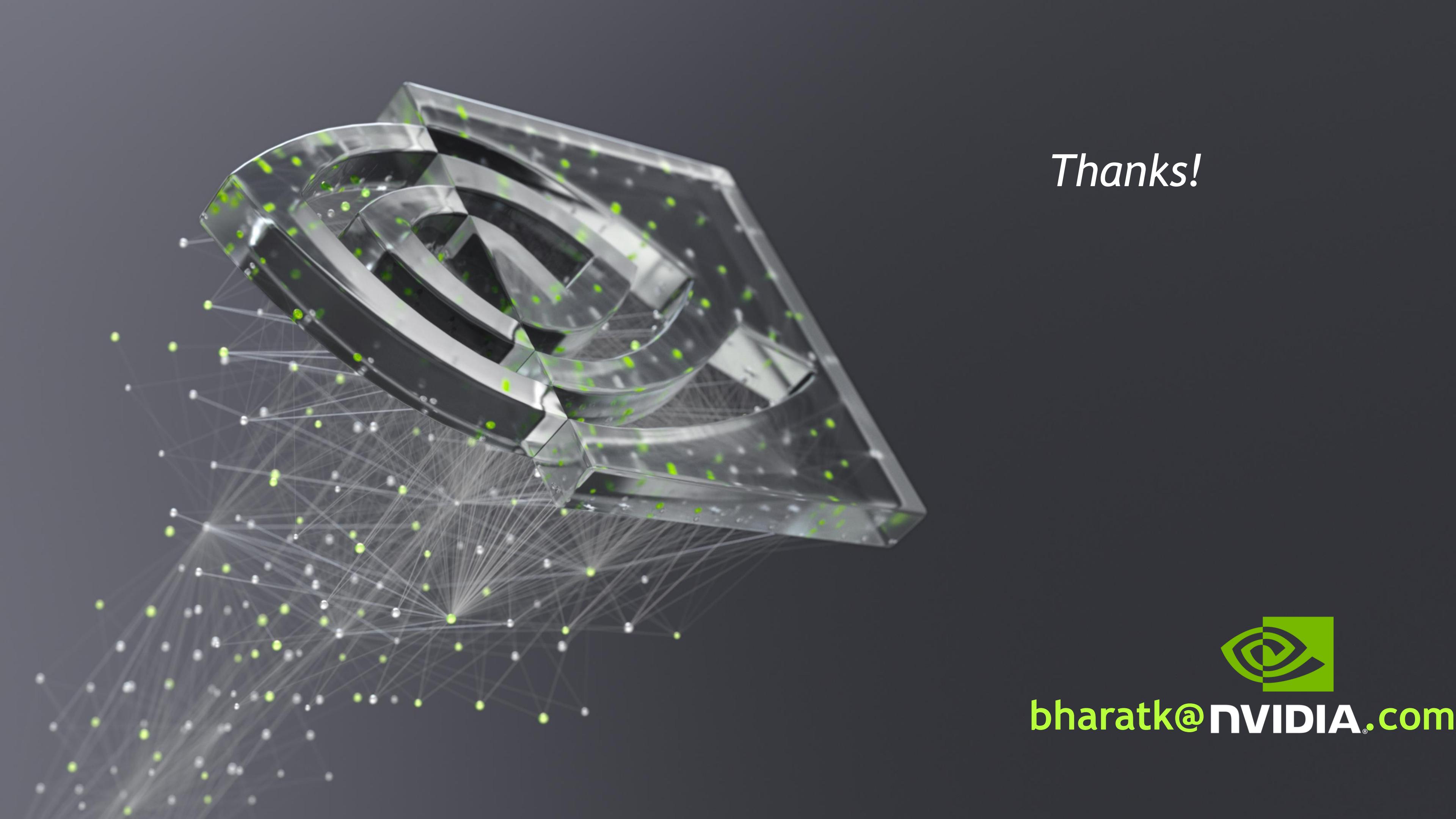
Non-Gated Residual Network



U GATED NETWORK

2D Flow Prediction Network





Thanks!



bharatk@**NVIDIA**.com