#NDCLondon

Deep Learning in the world of little ponies

@GaliyaWarrier

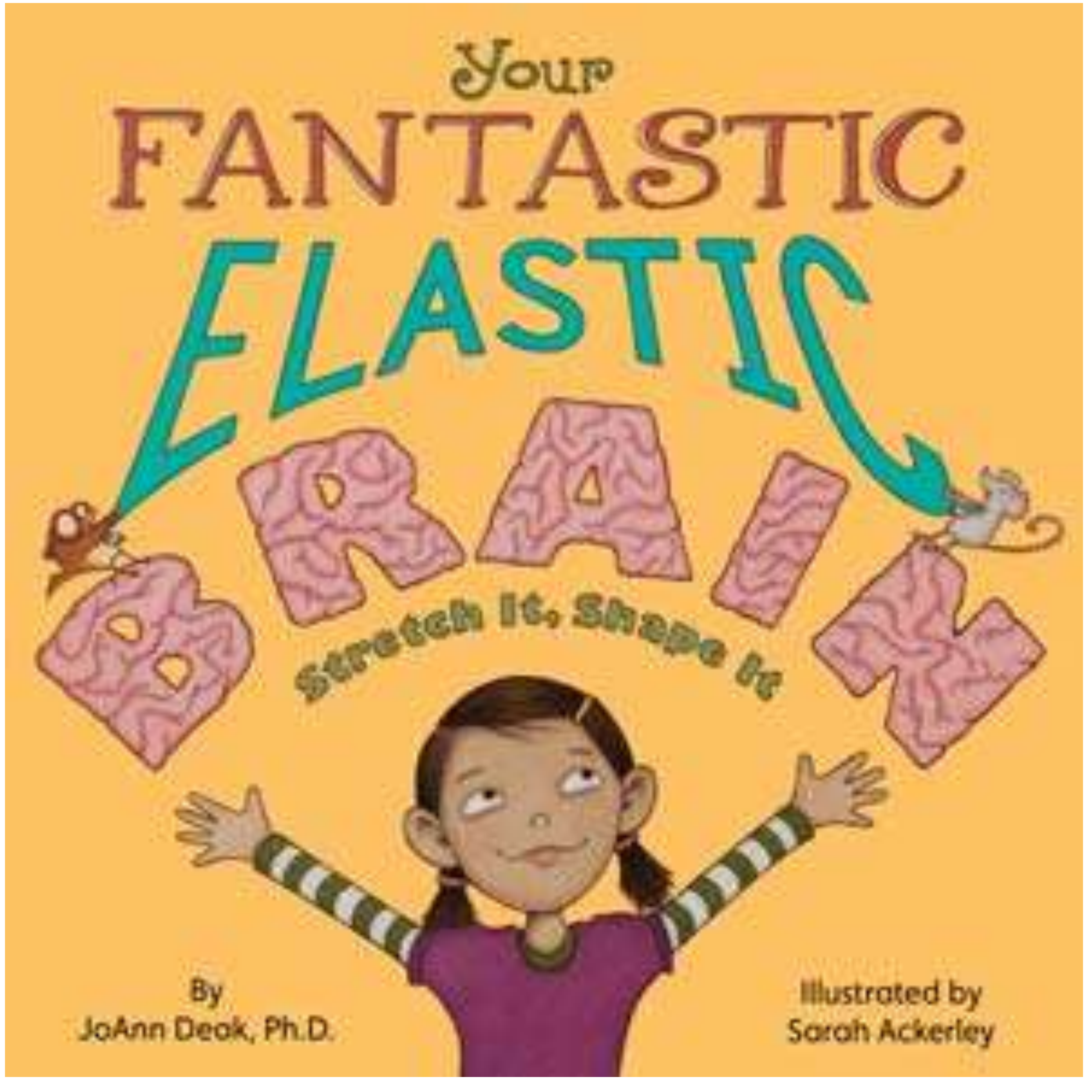
Galiya Warrier

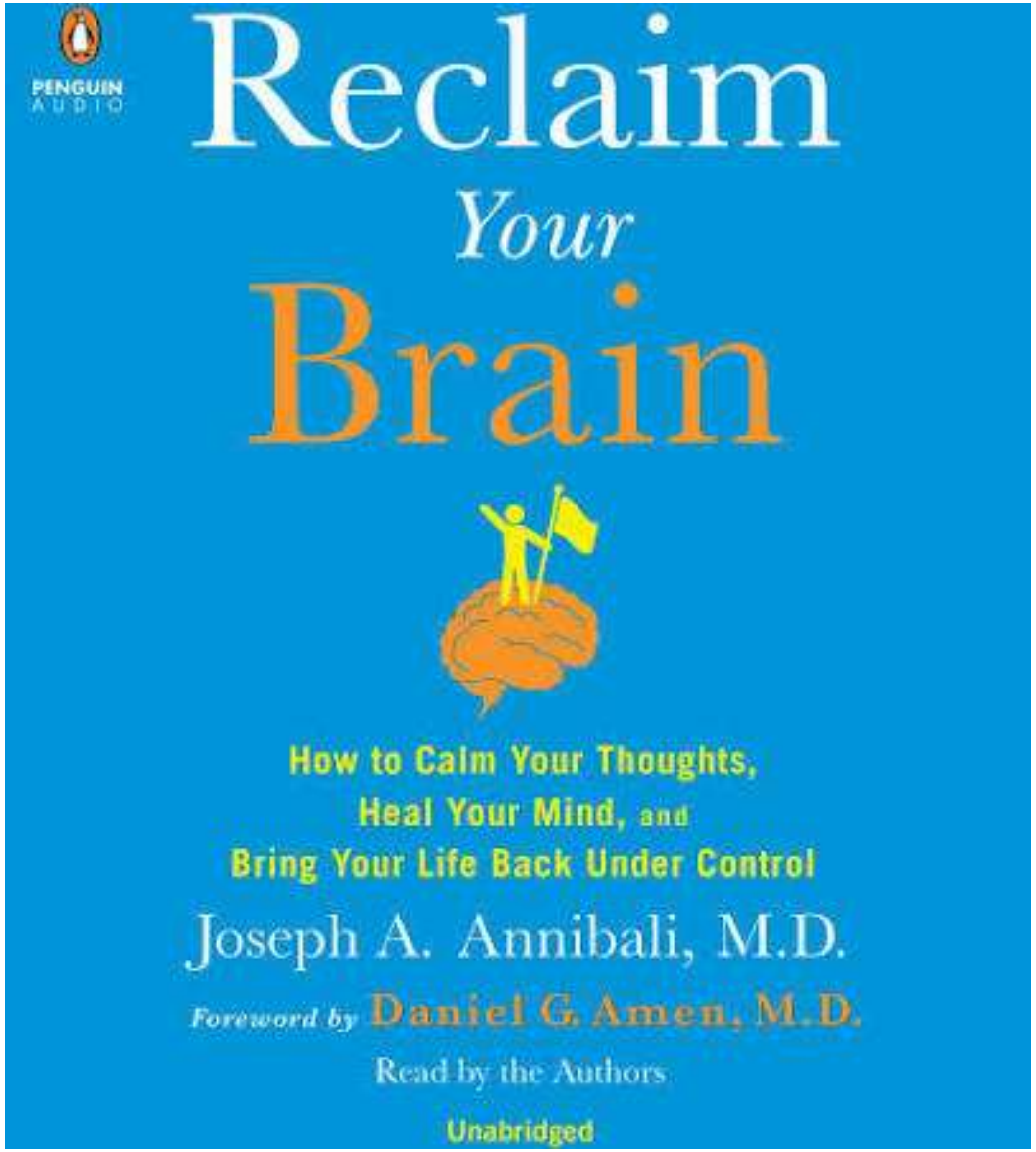
Cloud Solution Architect,
Advanced Analytics & AI,
Microsoft,
UK

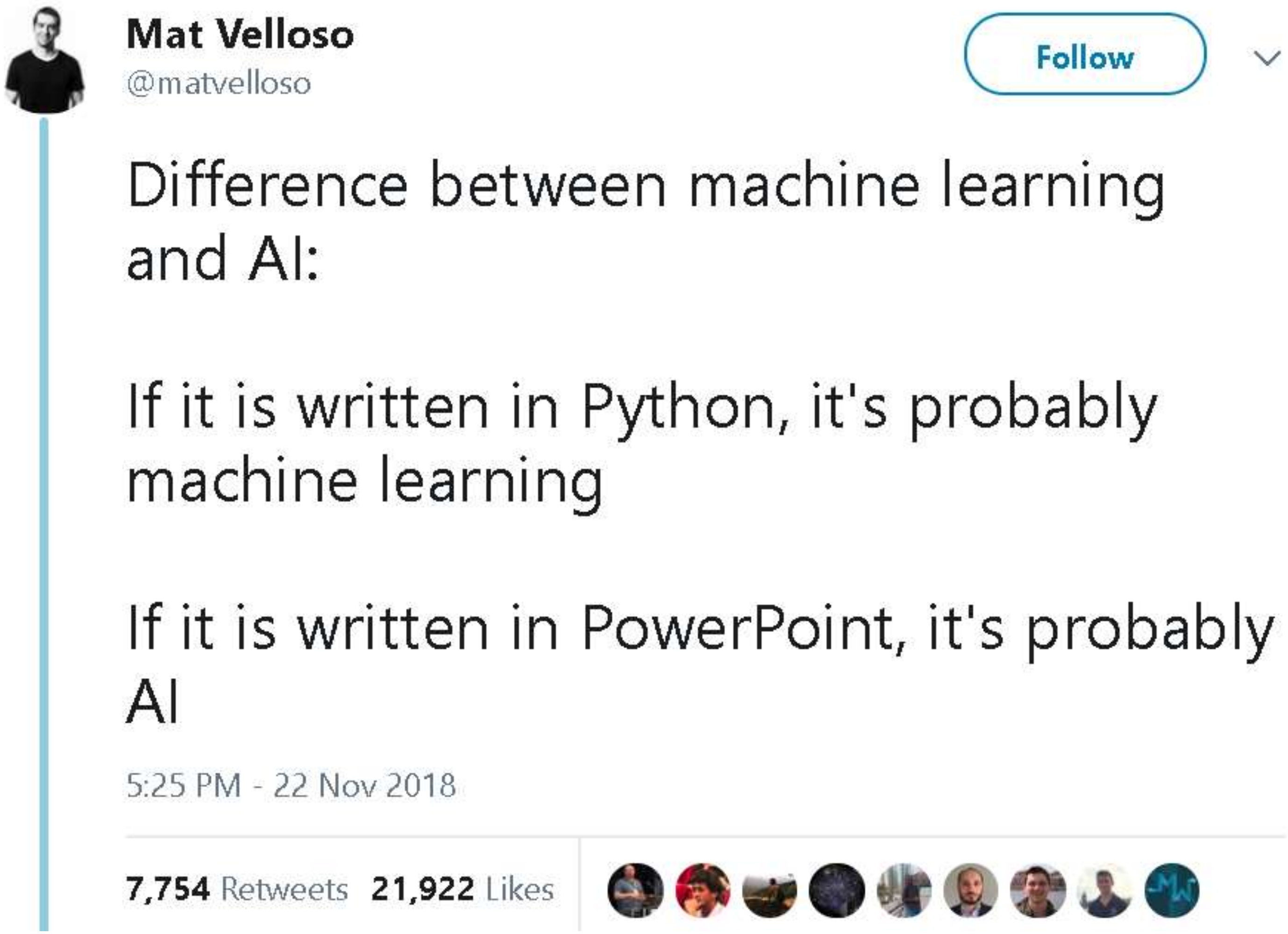
Maya Warrier

4 years old,
reception class

Twilight's Doll Smarty Pants
Twilight Sparkle
Starlight
Owlowiscious
Diamond Tiara
Silver Spoon
Rarity
Sweetiebelle
Octavia
Vinyl Scratch
Gilda
Shining Armour
Lightshine
Pumpkin Cake
Pound Cake
Twist
Braeburn
Babs Seed
Applebloom
Ironwill
Spike
Docter Whooves
Derpy
Pinkie Pie
Rainbow Dash
Scootaloo
Thunderlane
Rumble
Cheerilee
Big Macintosh
Applejack
Snips
Snails
Lyra
Fluttershy
Princess Celestia
Princess Luna
Princess Cadence
Zaucrua
The Great of Powerful Trixie
Snowflake
Bonbon
my LITTLE PONY
FRIENDSHIP is MAGIC


Your
FANTASTIC
ELASTIC
BRAIN
Stretch It, Shape It
By
JoAnn Deak, Ph.D.
Illustrated by
Sarah Ackerley


PENGUIN AUDIO
Reclaim
Your
Brain
How to Calm Your Thoughts,
Heal Your Mind, and
Bring Your Life Back Under Control
Joseph A. Annibali, M.D.
Foreword by Daniel G. Amen, M.D.
Read by the Authors
Unabridged



Mat Velloso
@matvelloso

Difference between machine learning and AI:

If it is written in Python, it's probably machine learning

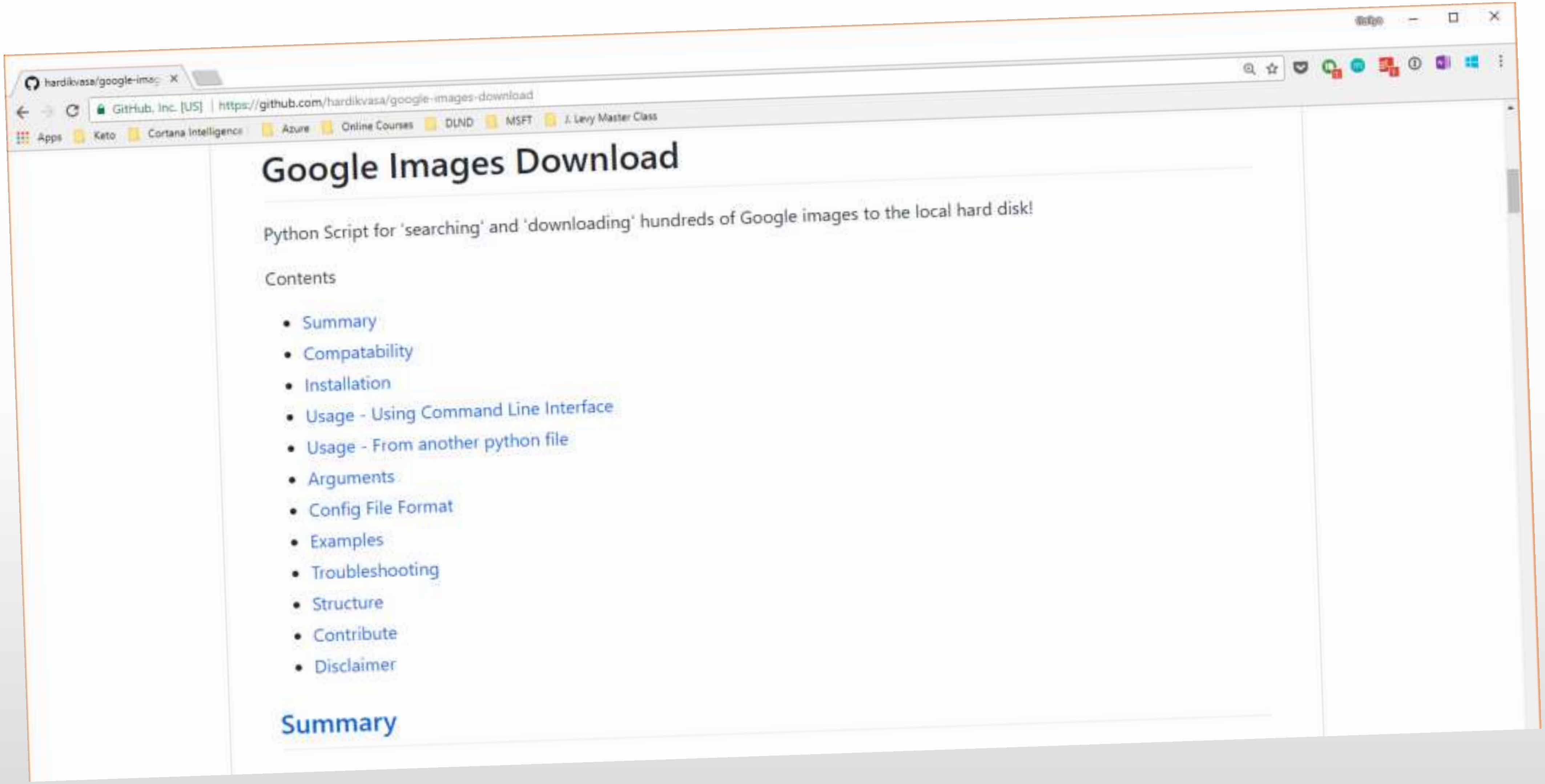
If it is written in PowerPoint, it's probably AI

5:25 PM - 22 Nov 2018

7,754 Retweets 21,922 Likes

But before starting with anything …
Do we have data?

https://github.com/hardikvasa/google-images-download



Google Images Download

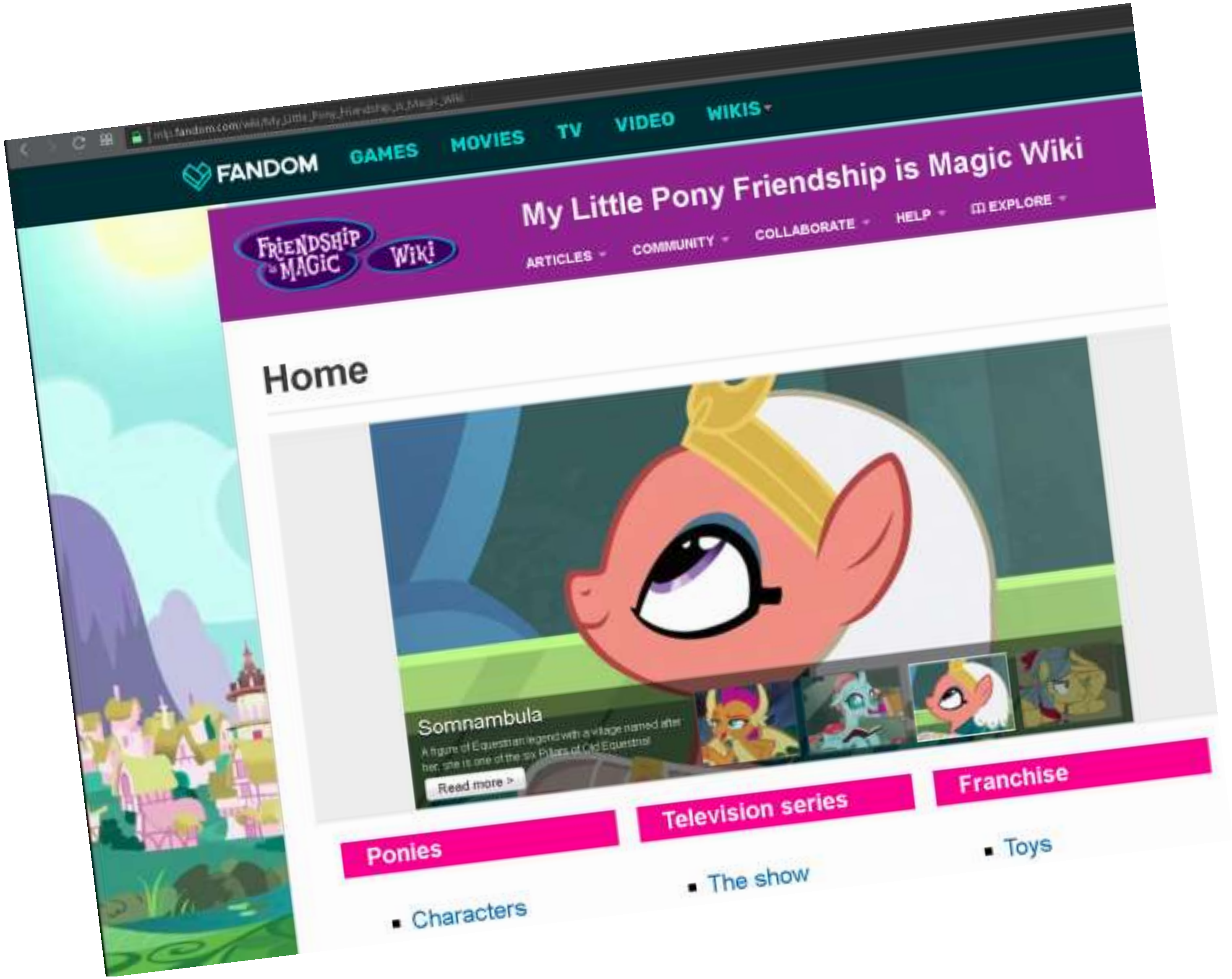
Python Script for 'searching' and 'downloading' hundreds of Google images to the local hard disk!

Contents

- Summary
- Compatability
- Installation
- Usage - Using Command Line Interface
- Usage - From another python file
- Arguments
- Config File Format
- Examples
- Troubleshooting
- Structure
- Contribute
- Disclaimer

Summary

https://mlp.fandom.com/wiki/My_Little_Pony_Friendship_is_Magic_Wiki



My Little Pony Friendship is Magic Wiki

Home

Somnambula

A figure of Equestrian legend with a village named after her, she is one of the six Pillars of Old Equestria

Ponies

- Characters

Television series

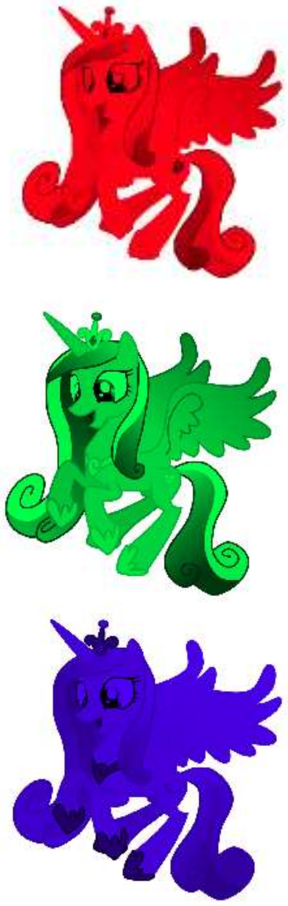
- The show

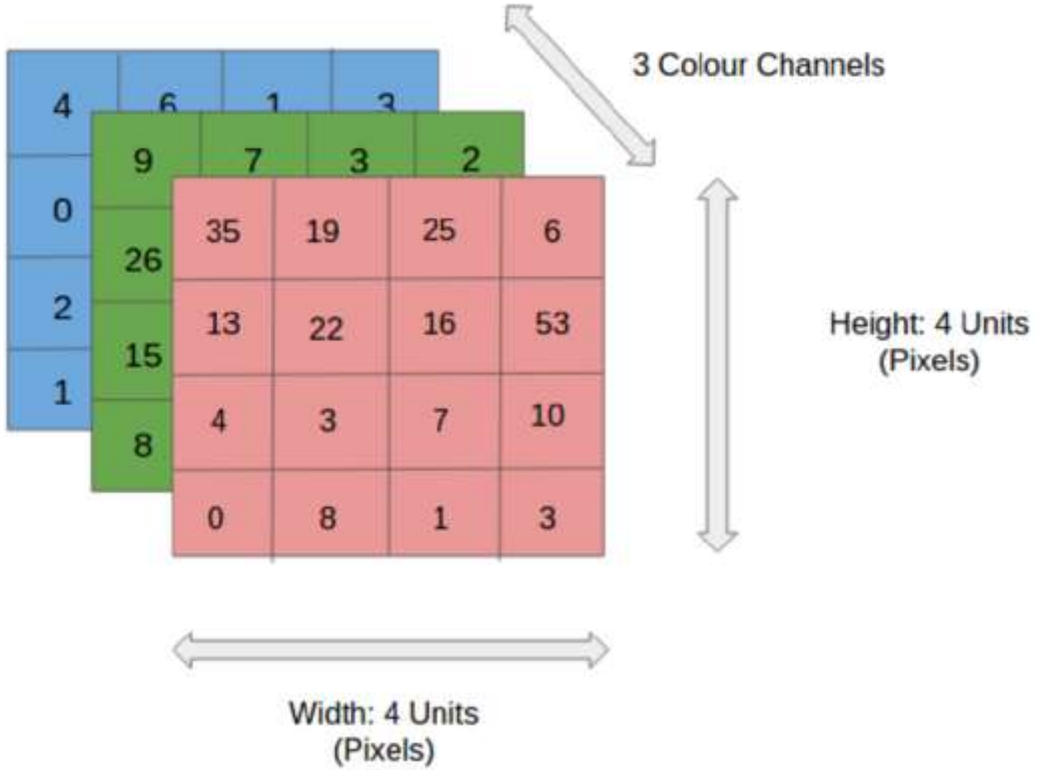
Franchise

- Toys

lorem
Hoy
#i
नमस्ते
Ahoj
DIA
duuit
ПРИВЕТ
cześć

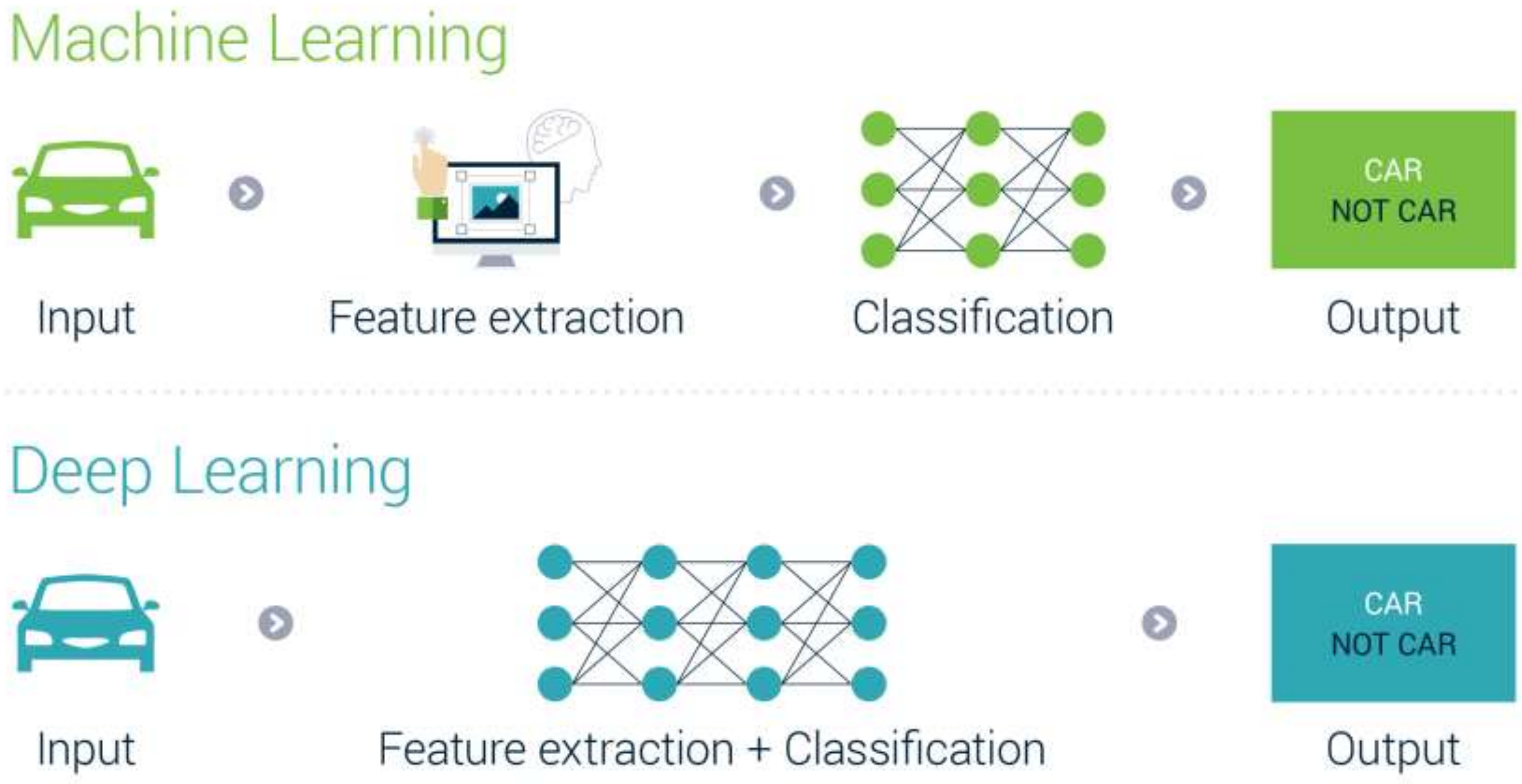
What is an image?





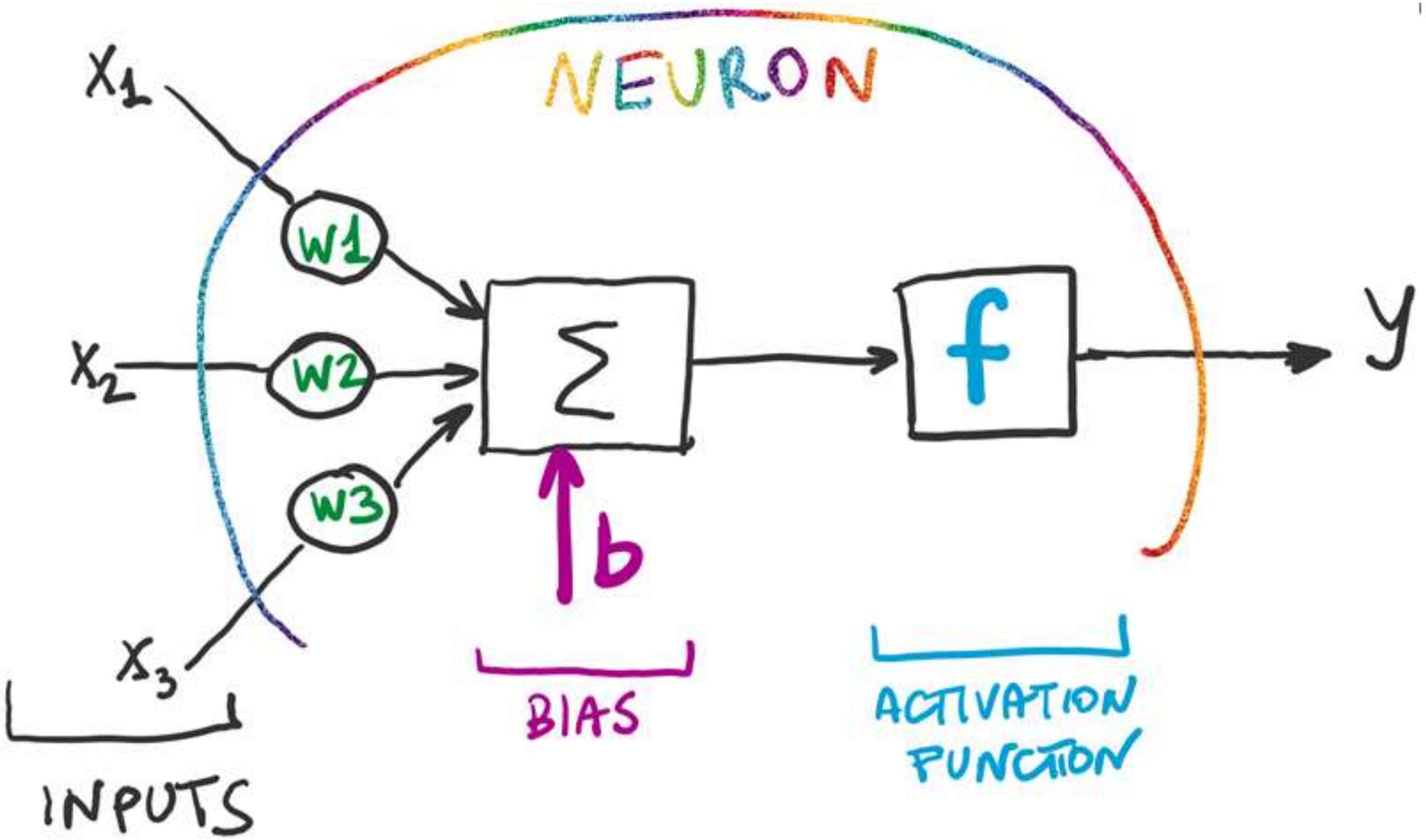
ML: Feature engineering

Classic ML vs. Deep Learning

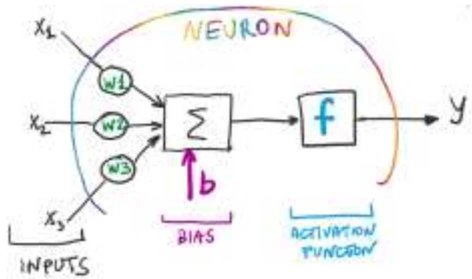


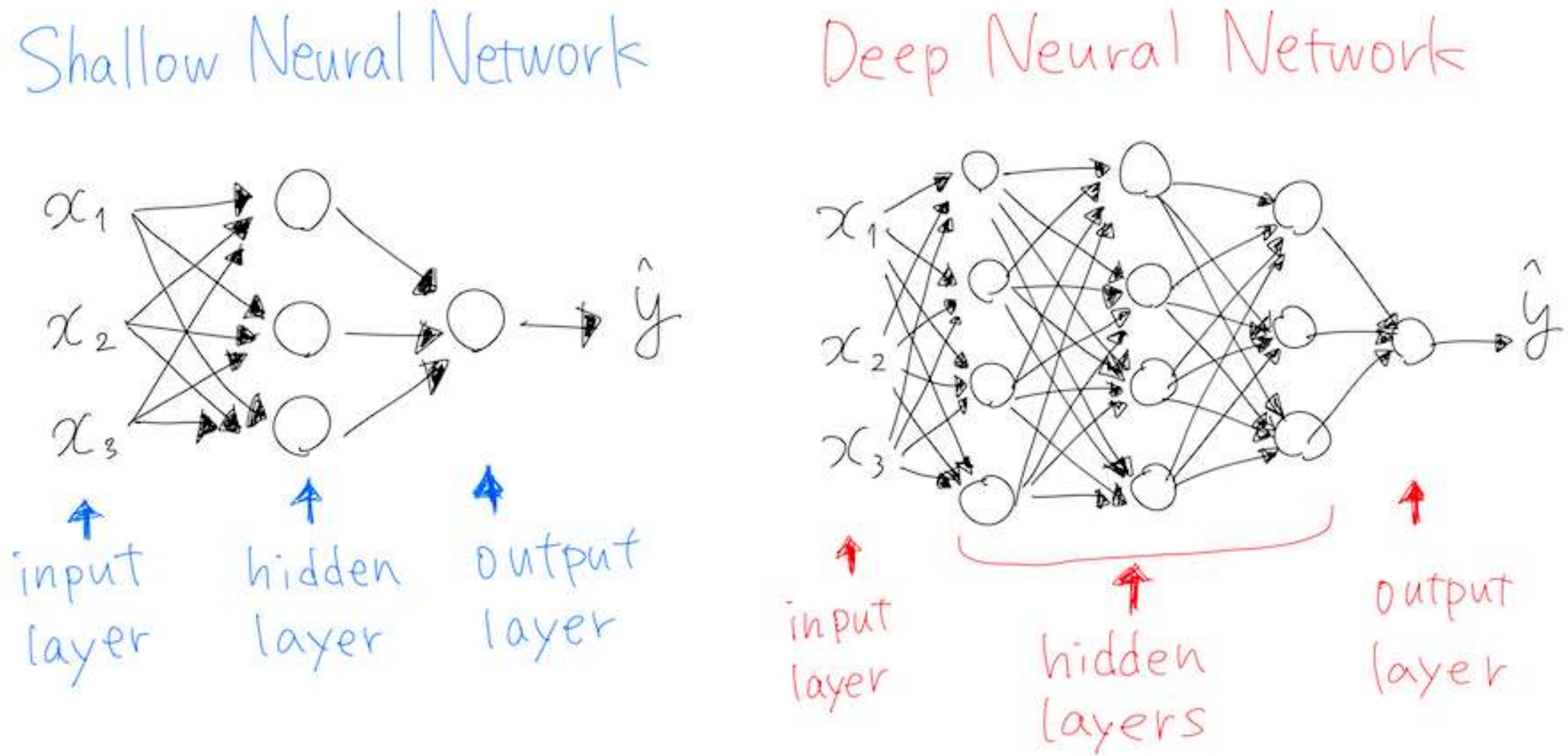
Representations

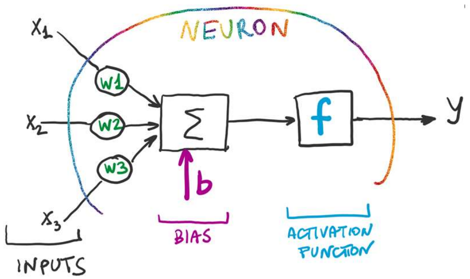
What is a Neuron?



Layers in Neural Network





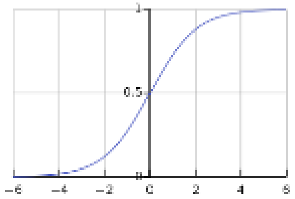


Activation functions

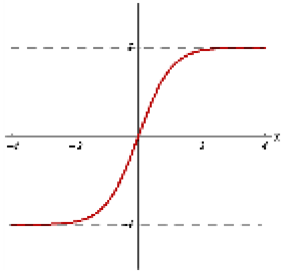
- Good ones are continuous & infinite in domain

- Usually monotonous and don't change direction

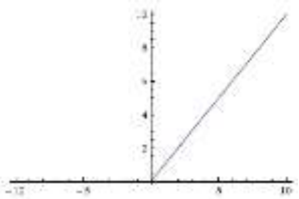
- These functions (and their derivatives) should be efficiently computable



Sigmoid



Tanh

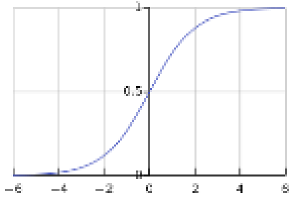


ReLU
(Rectified Linear Unit)

Hidden layer activation functions

NONE

Predict raw data values

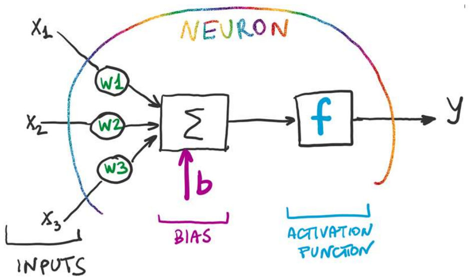


Sigmoid
(Unrelated probs)

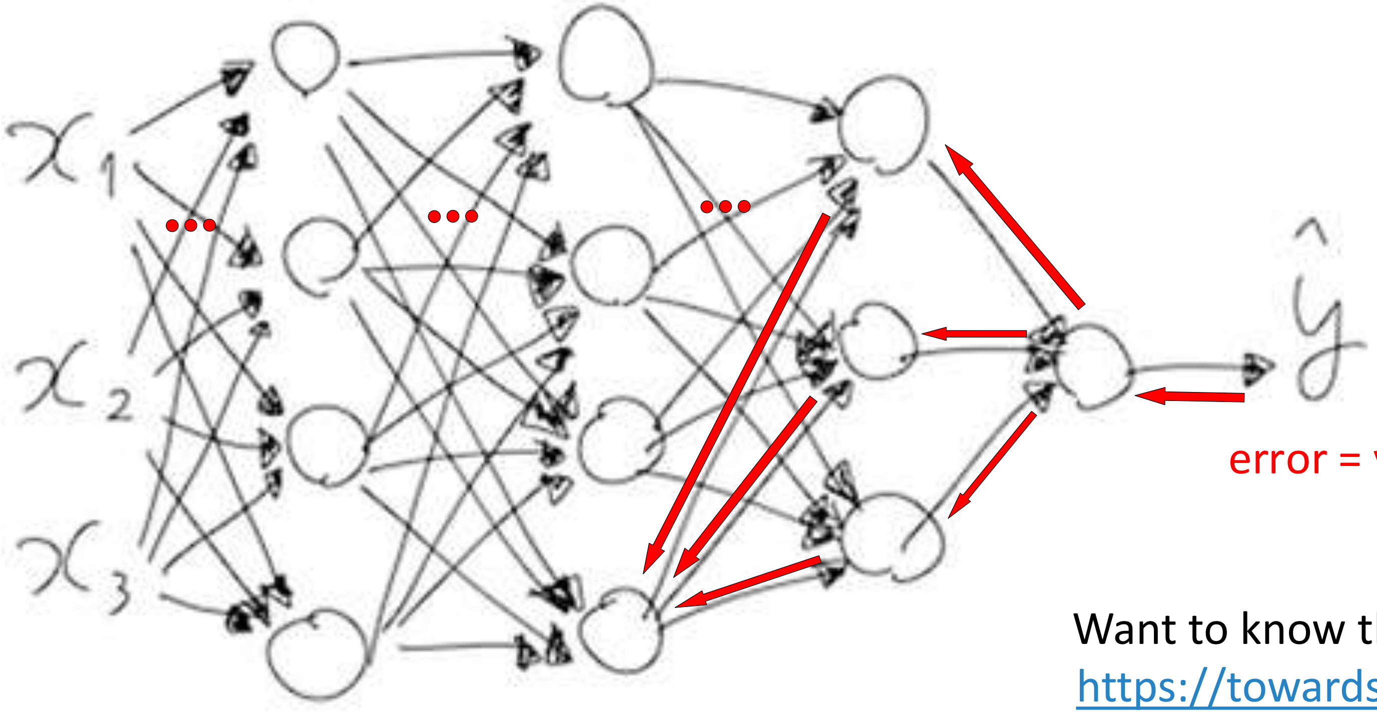
SOFTMAX

Softmax
("which one" prob)

Output layer activation functions



Backpropagation



error = y - ŷ

Want to know the math behind the scene?
https://towardsdatascience.com/learning-backpropagation-from-geoffrey-hinton-619027613f0

Neural Network Architectures

Densely Connected Neural Networks

Convolutional Neural Networks
Spatial relationships (i.e. vision)

Recurrent Neural Networks
Time relationships (i.e. time series, language, speech)

Putting things into practice

TensorFlow

mxnet

GLUON

The Microsoft Cognitive Toolkit

Caffe

torch

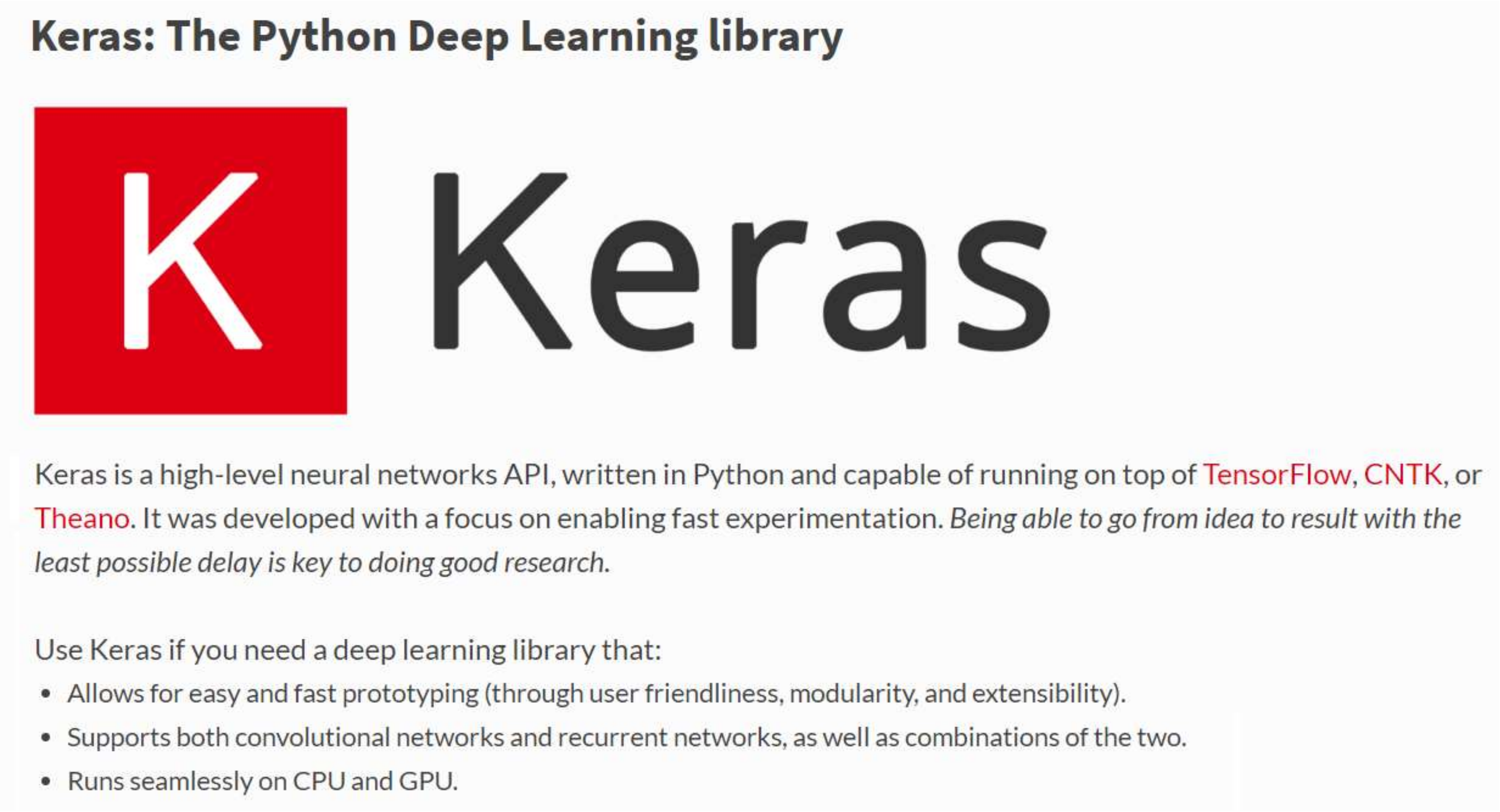
PYTORCH

K

theano

Caffe2

And others …



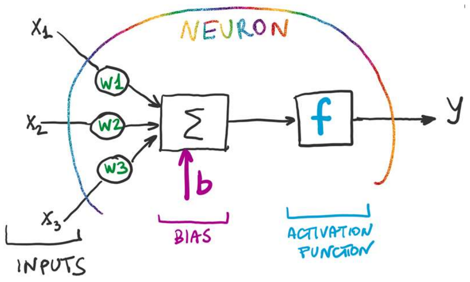
Keras: The Python Deep Learning library

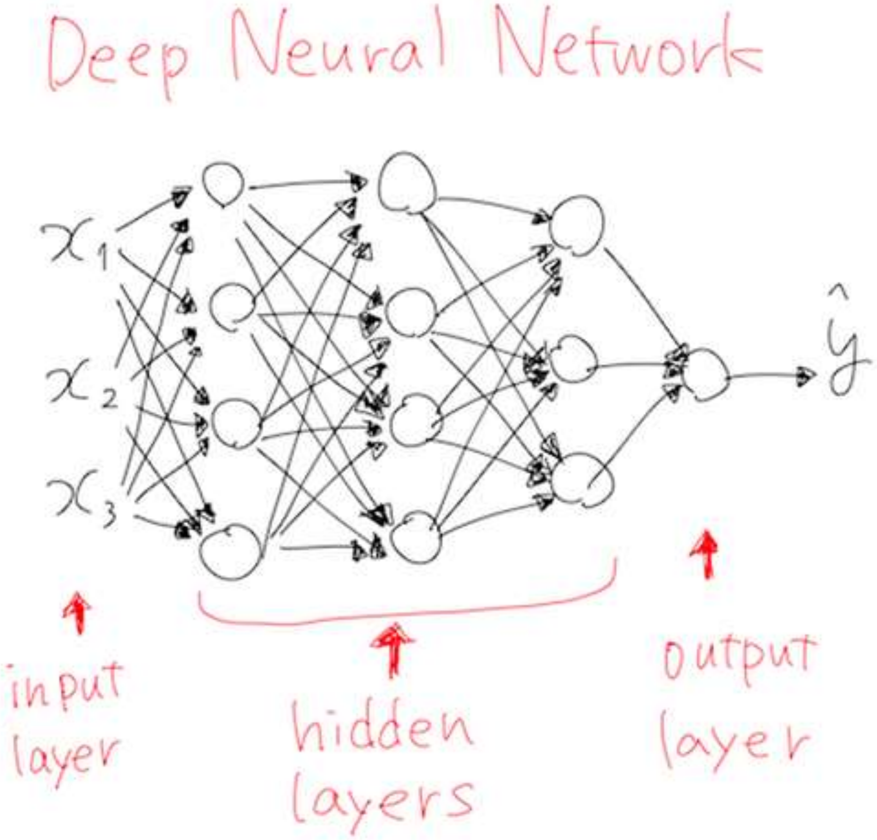
Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Densely Connected Neural Networks





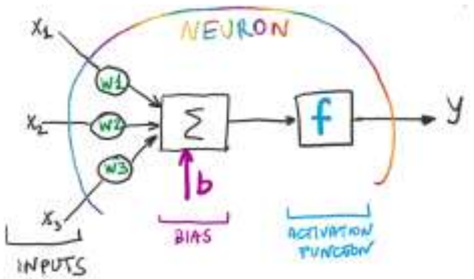
All neurons in any hidden layer have full connections to all activations in the previous layer

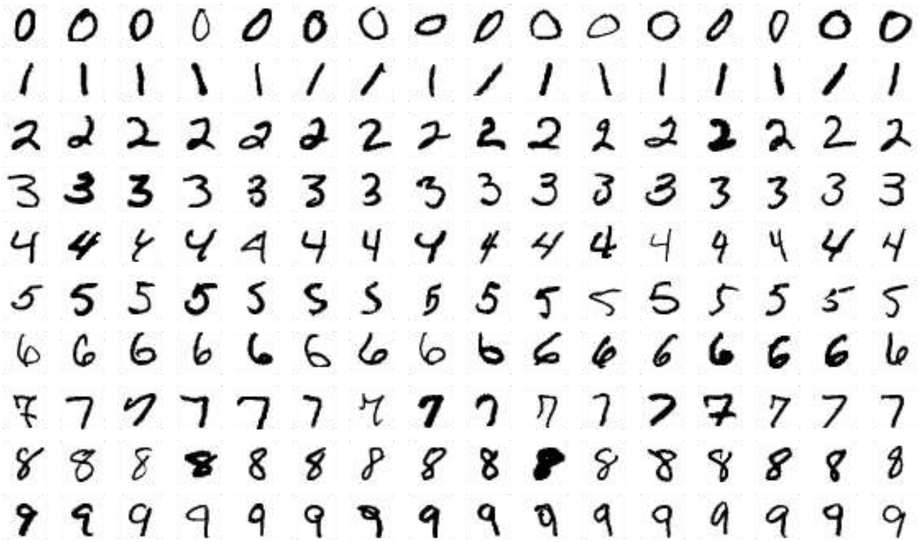
Number of parameters on each layer is huge

Model's capacity = number of learnable parameters

DEMO

Densely Connected Neural Networks: MNIST Classification





MNIST database (Modified National Institute of Standards and Technology database)

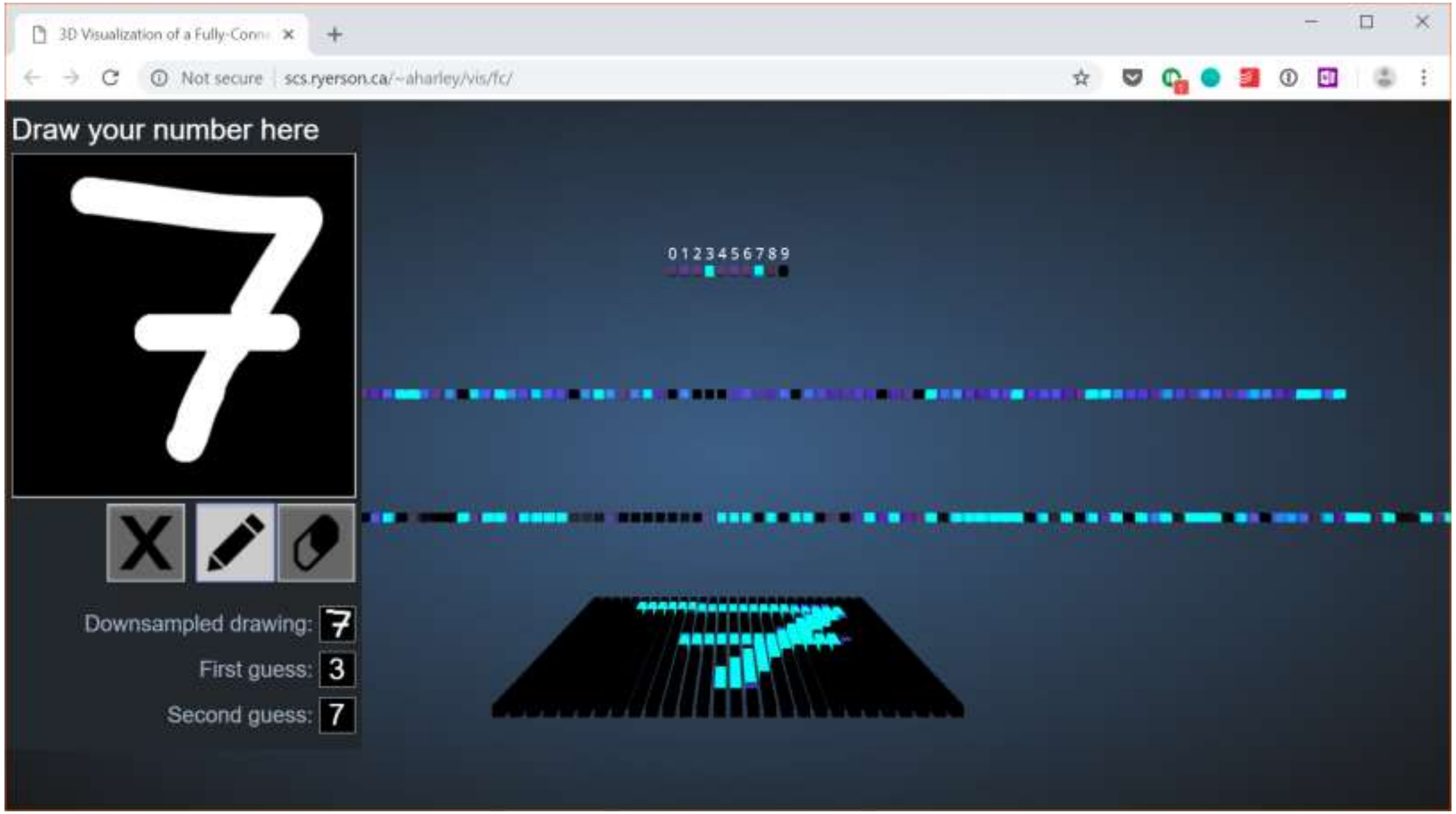
Large database of handwritten digits (28x28)

60,000 training and 10,000 testing images

“Hello World” of computer vision tasks

DEMO

3D Visualization of a Dense Neural Network trained on MNIST dataset

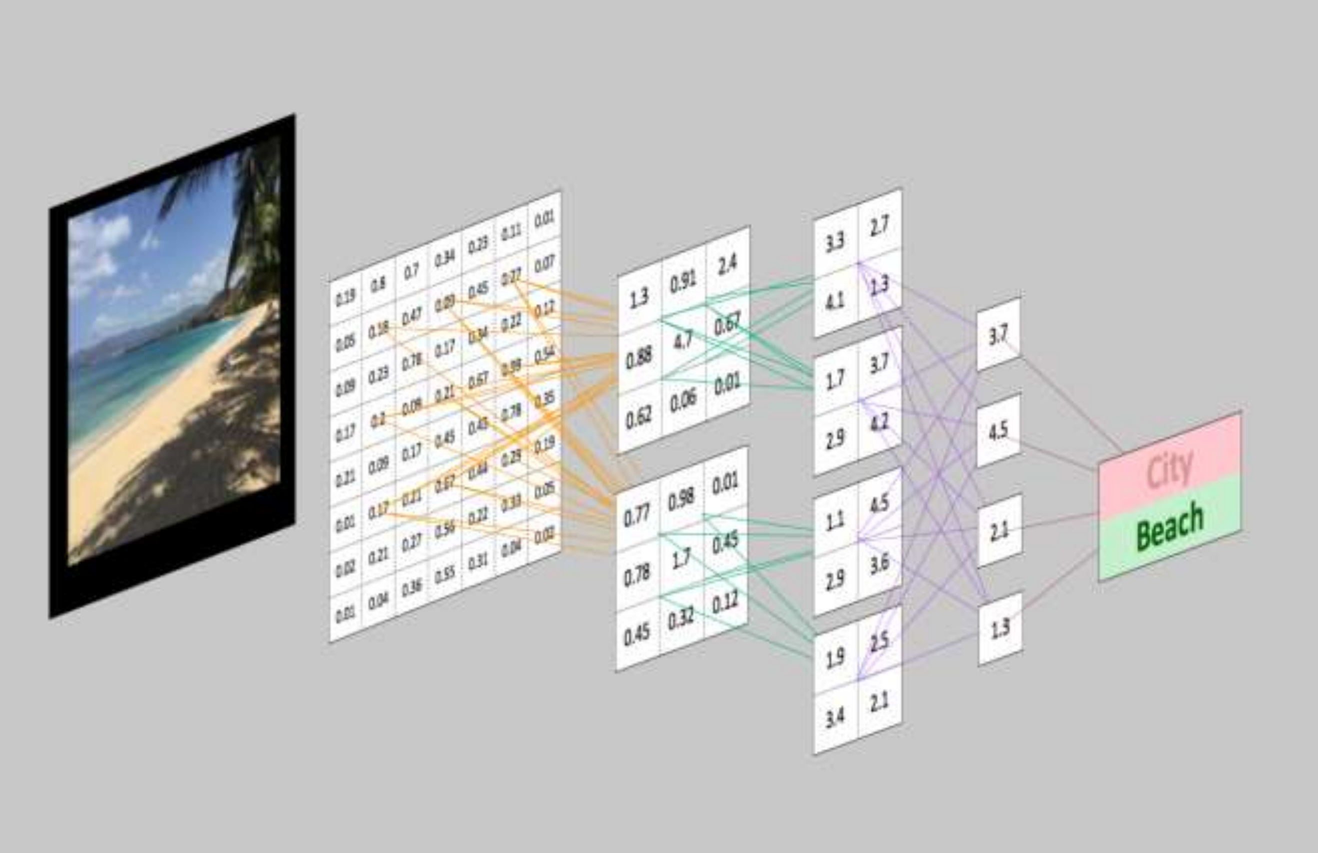

3D Visualization of a Fully-Conn
Not secure
scs.ryerson.ca/~aharley/vis/fc/
Draw your number here
0 1 2 3 4 5 6 7 8 9

Downsampled drawing: 7

First guess: 3

Second guess: 7

http://scs.ryerson.ca/~aharley/vis/fc/



Biological inspiration:

* Visual cortex in animals and humans

* Cells are sensitive to small subregions of the input

Important for CNNs:

* Structure to input data

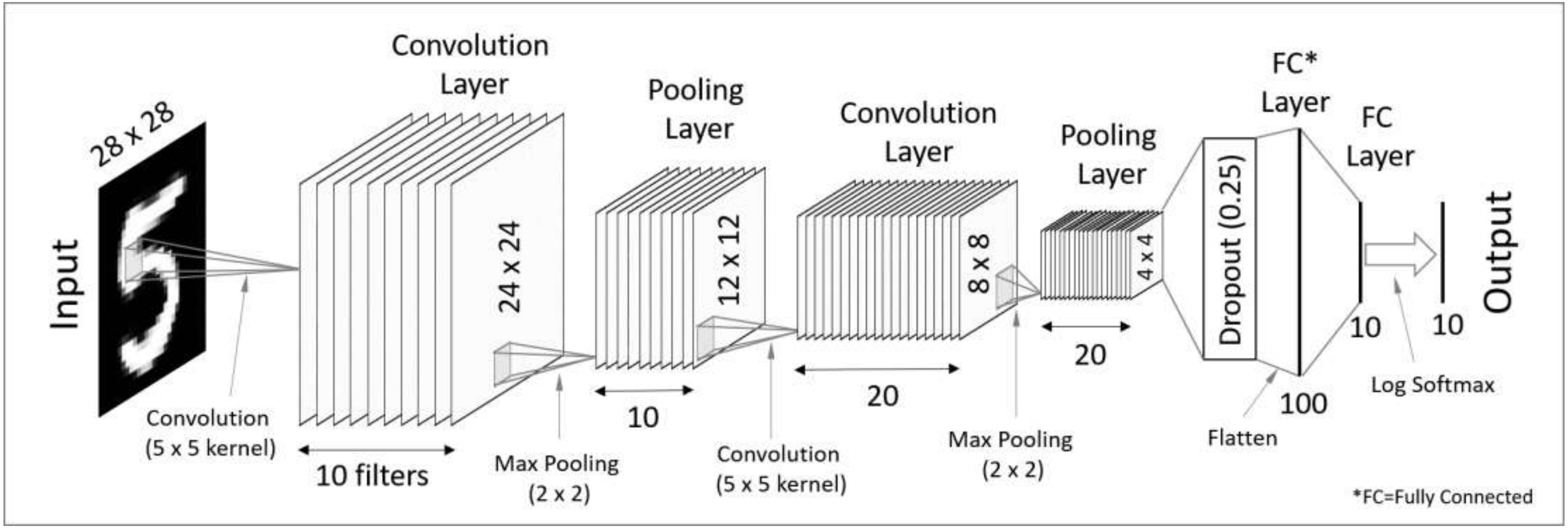
* Spatial relationships

* Repeated patterns

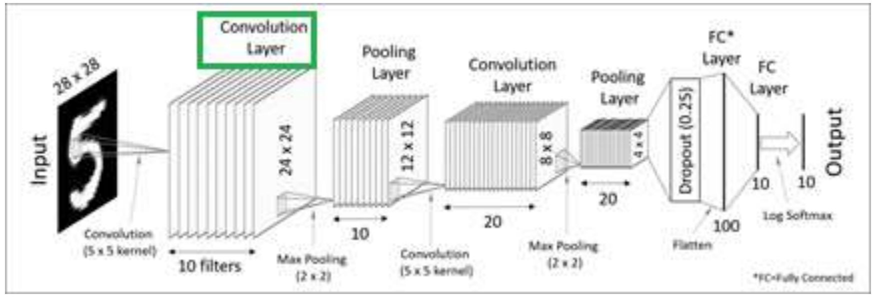
Commonly used for computer vision applications

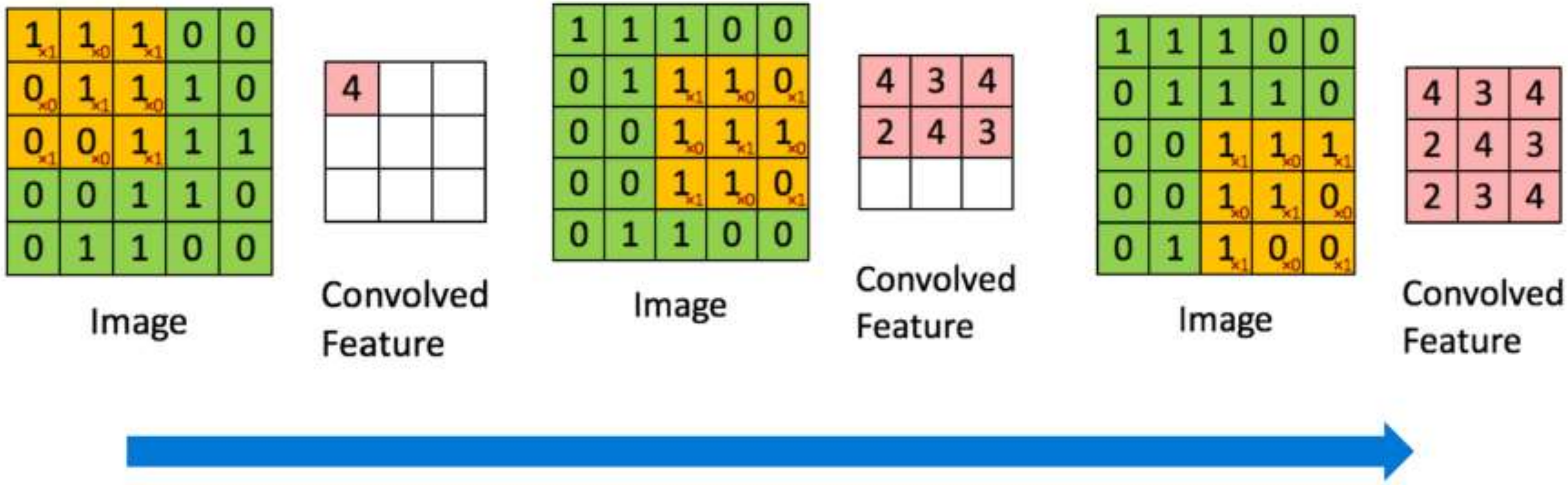
Convolutional Neural Networks (CNNs)

CNNs: Typical architecture



CNNs: Convolution Layer





Image

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Convolved Feature

4		

Image

1	1	1	0	0
0	1	$1_{\times 1}$	$1_{\times 0}$	$0_{\times 1}$
0	0	$1_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$
0	0	$1_{\times 1}$	$1_{\times 0}$	$0_{\times 1}$
0	1	1	0	0

Convolved Feature

4	3	4
2	4	3

Image

1	1	1	0	0
0	1	1	1	0
0	0	$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$
0	0	$1_{\times 0}$	$1_{\times 1}$	$0_{\times 0}$
0	1	$1_{\times 1}$	$0_{\times 0}$	$0_{\times 1}$

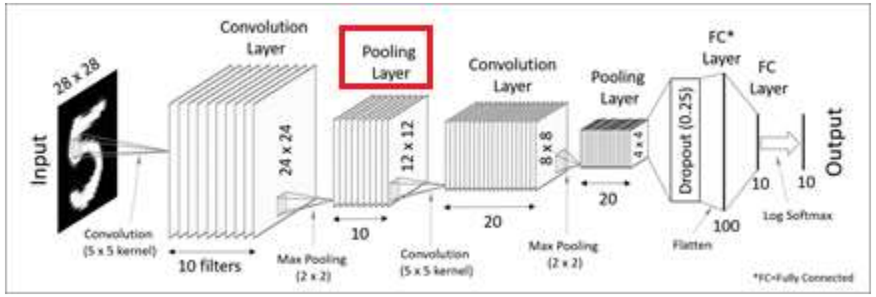
Convolved Feature

4	3	4
2	4	3
2	3	4

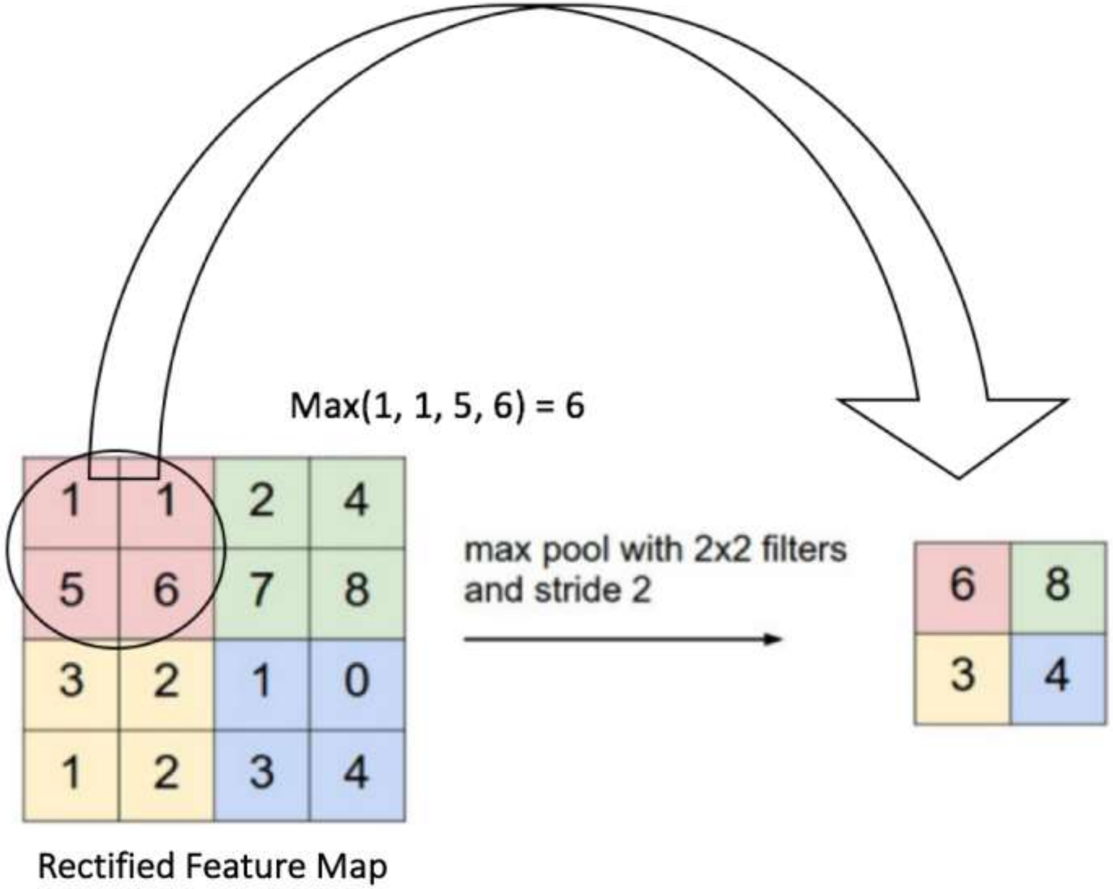
Extract features from input image

Slide filter (3x3, 5x5, etc.) over image to obtain a feature map per filter

Multiple filters



CNNs: Pooling layer



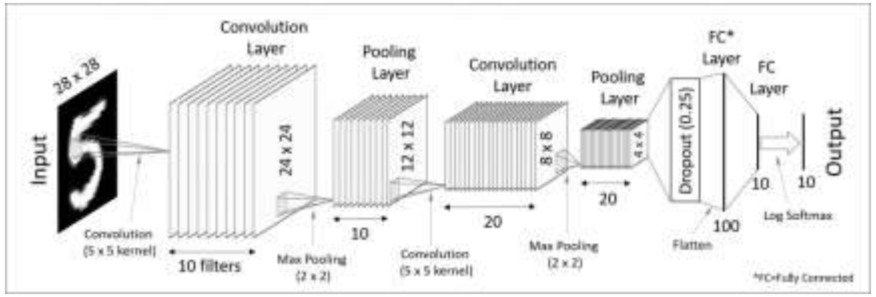
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

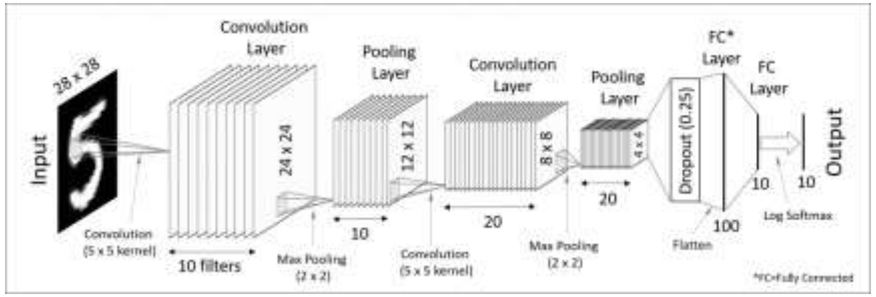
Reduce dimensionality BUT
retain most important information

Can be of different types:
Max, Average, Sum

Information distillation by CNN

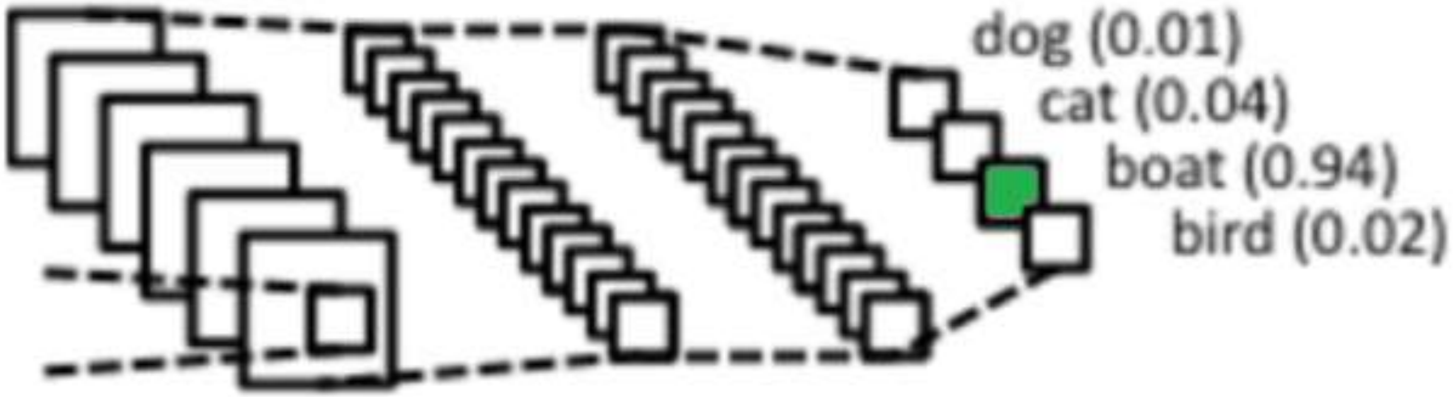


CNNs: Fully connected layer

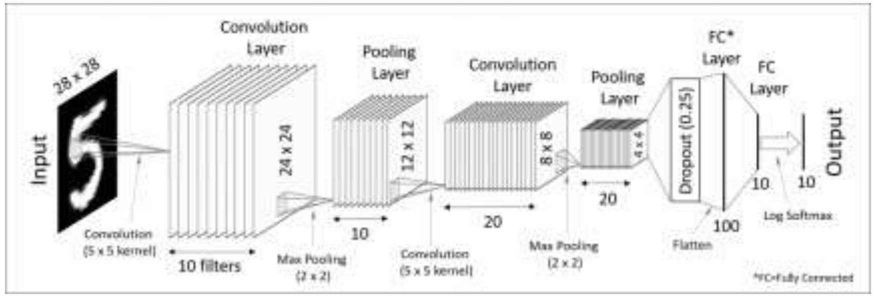


Input: High-Level features of original image

Uses those to classify input image into various classes [based on the training dataset]



Demo: Build custom pony classifier from scratch



Input

A folder with 3 subfolders of images

150px * 150px each

Output

Number of classes – 3

Names: (on the right)

Rarity Twilight sparkle Rainbow Dash

Apple Jack Pinky pie Fluttershy

Overfitting problem

Any ML models can suffer from overfitting

Model can adjust to get the best performance on training dataset (**optimisation**)

Question is how well model **generalise** and perform on test data never seen before?

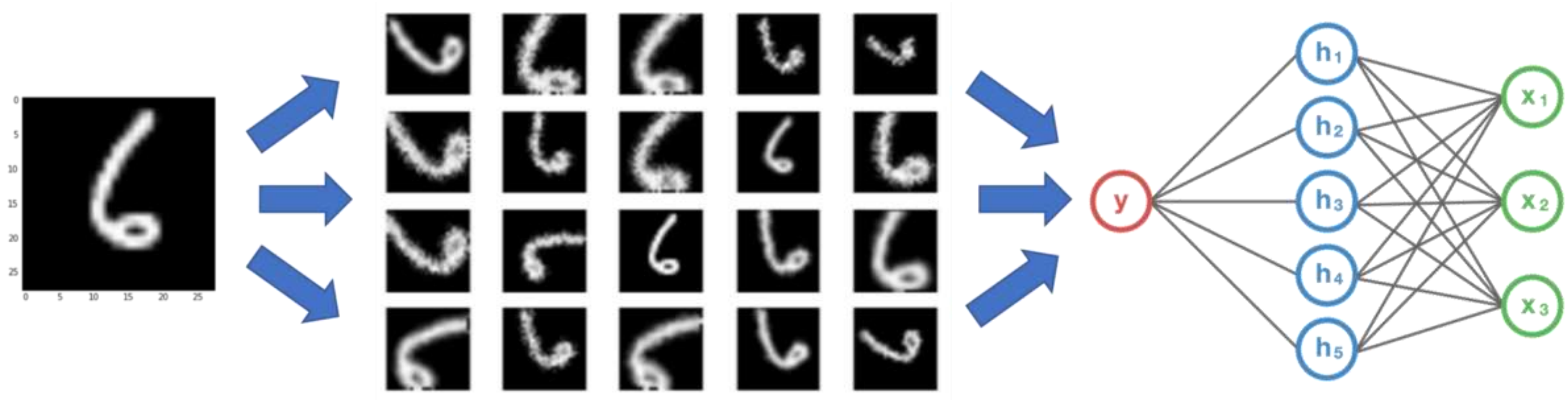
More training data always helps to reduce “memorisation”

Additional techniques: Data augmentation

"Generate more training samples given a small dataset"

Helps to deal with overfitting

Apply random transformations ("augment") to existing training samples to generate new training data



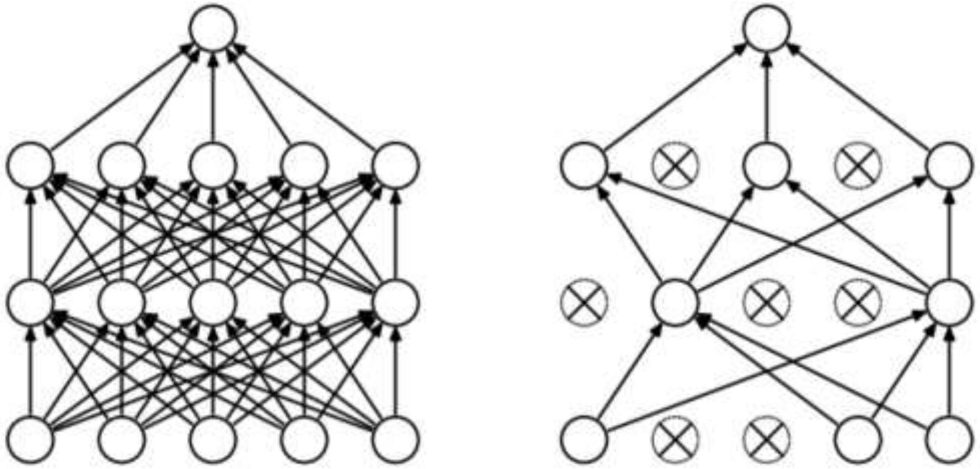
Additional techniques: Dropout

“Learn from less to learn better”

Reduces overfitting

Learn more robust features

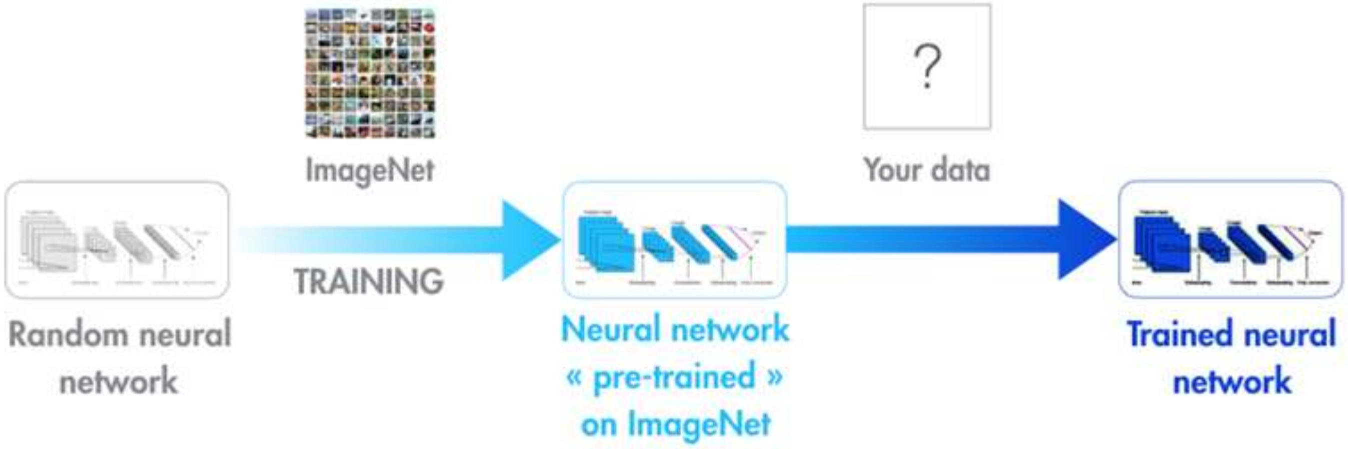
Dropout layer is effective together with data augmentation (fewer original samples)



(a) Standard Neural Net

(b) After applying dropout.

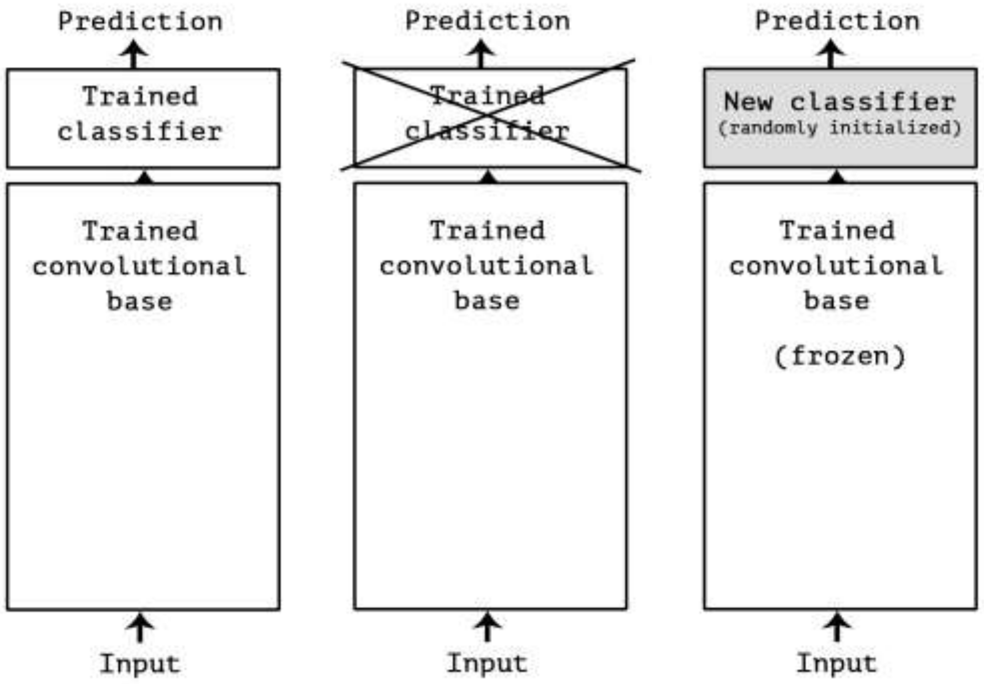
Transfer learning

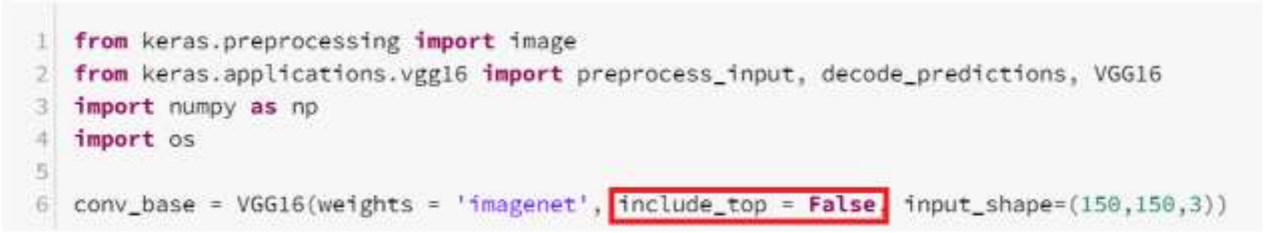


Lots of existing pre-trained networks are available:

- Xception
- InceptionV3
- ResNet50
- VGG16
- VGG19
- MobileNet

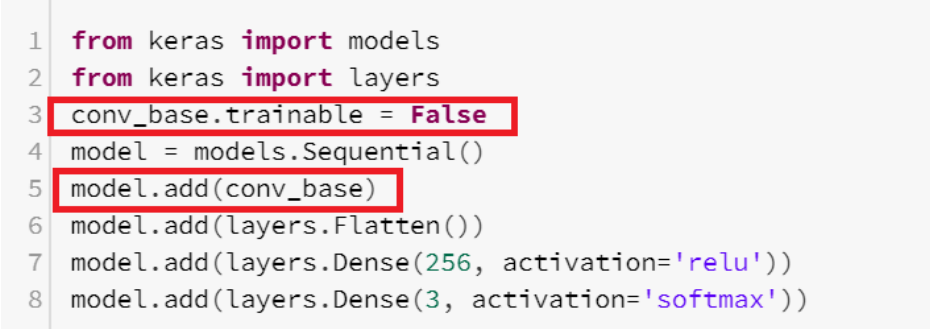
Pre-trained models: Feature extraction





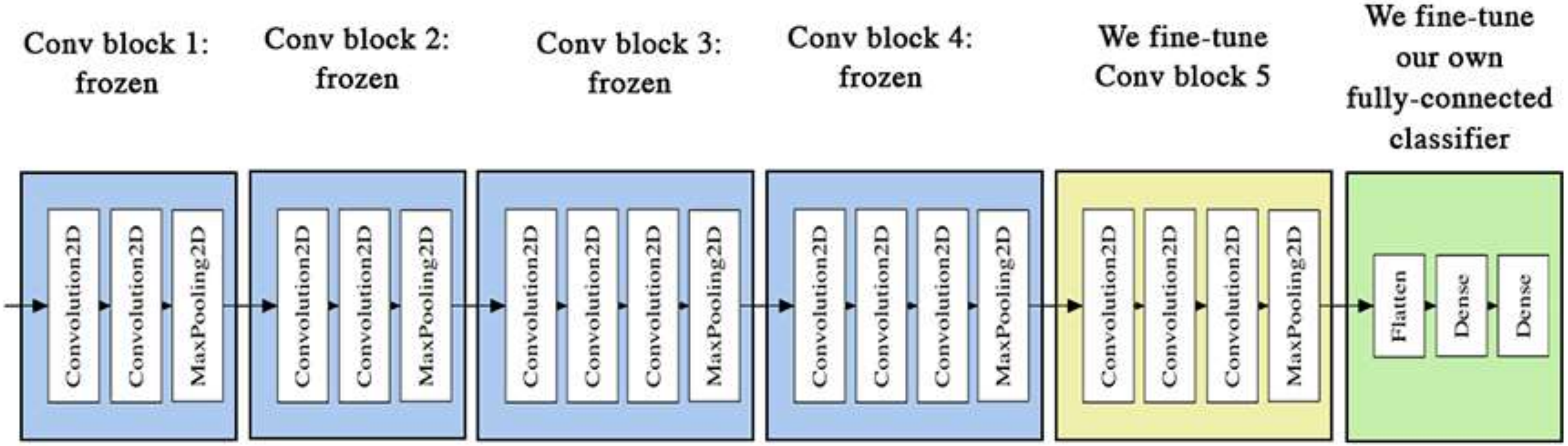
```python
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions, VGG16
import numpy as np
import os

conv_base = VGG16(weights = 'imagenet', include_top = False, input_shape=(150,150,3))
```



```python
from keras import models
from keras import layers
conv_base.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
```

Pre-trained models: Fine-tuning



Demo: Use pre-trained models for classification task

Input

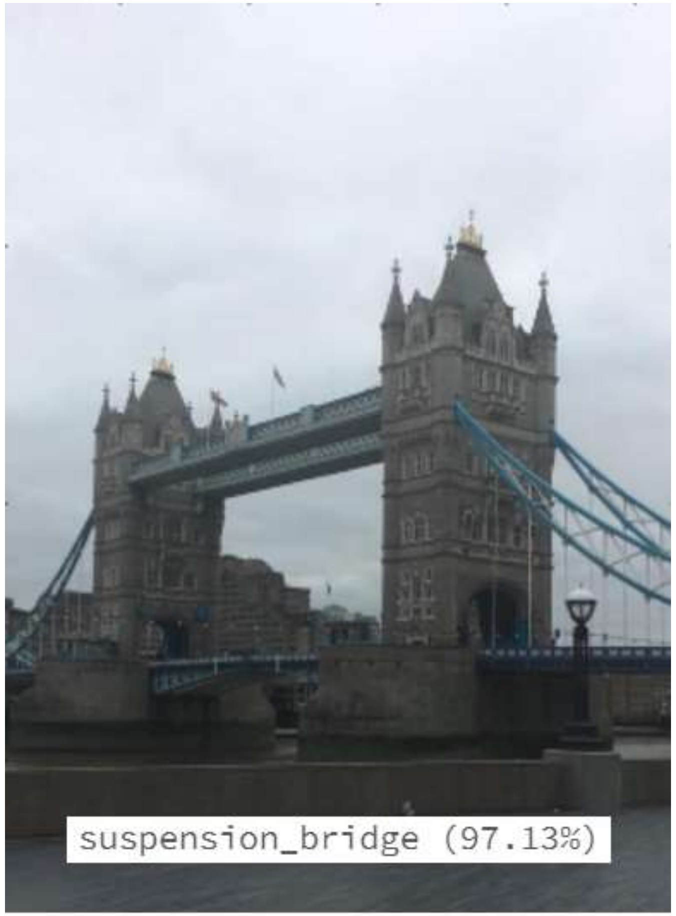
A number of personal images

Various sizes

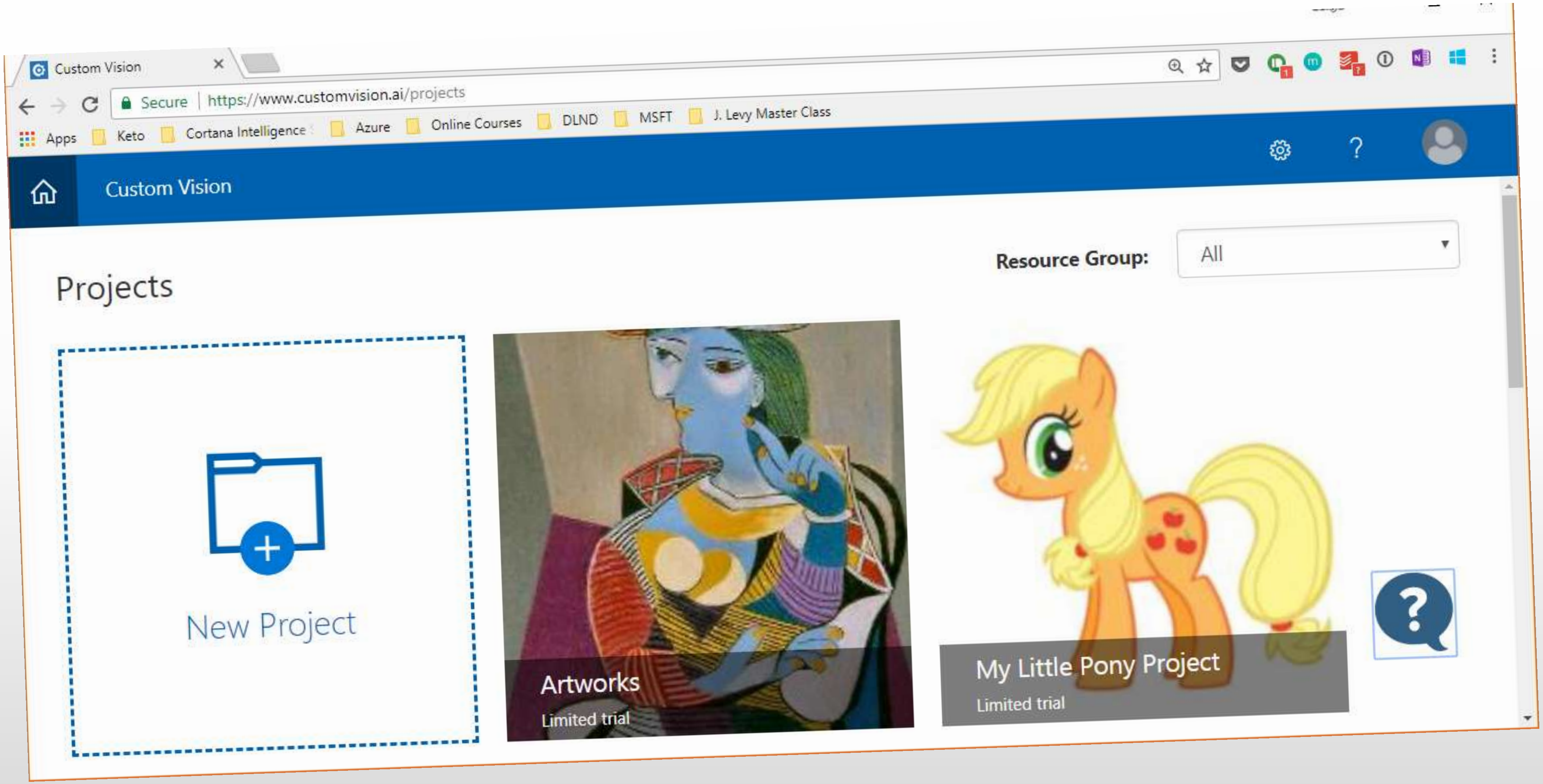
Output

Image class (as per pre-trained model)

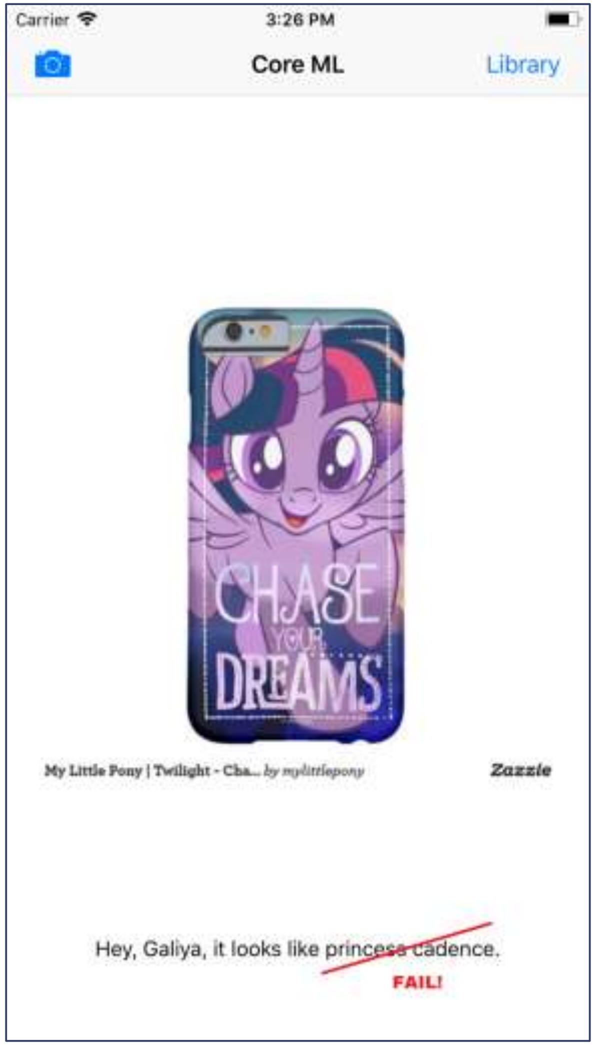
Probability



http://aka.ms/customvision

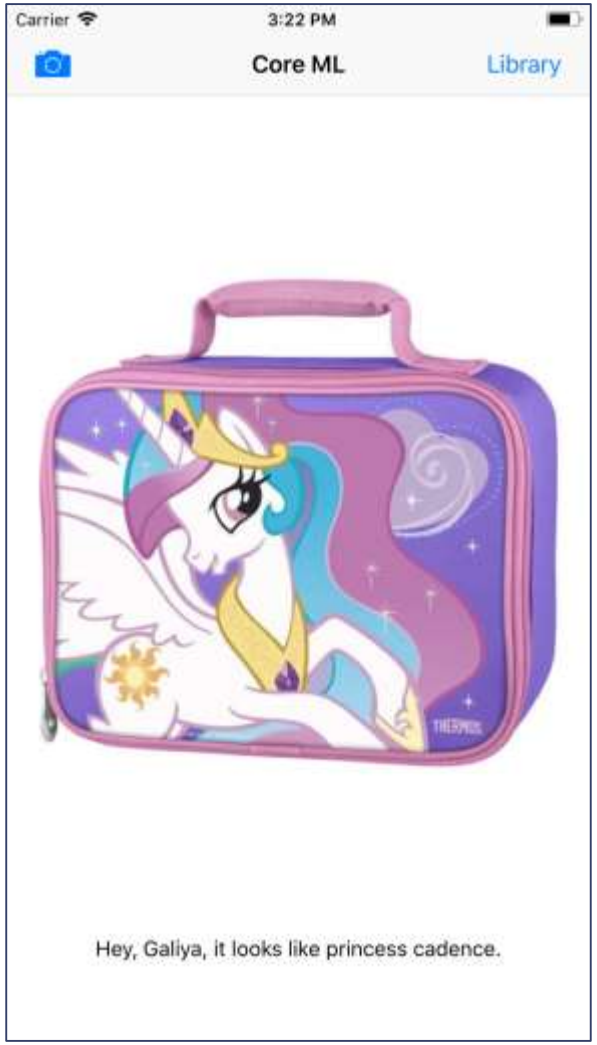


DEMO

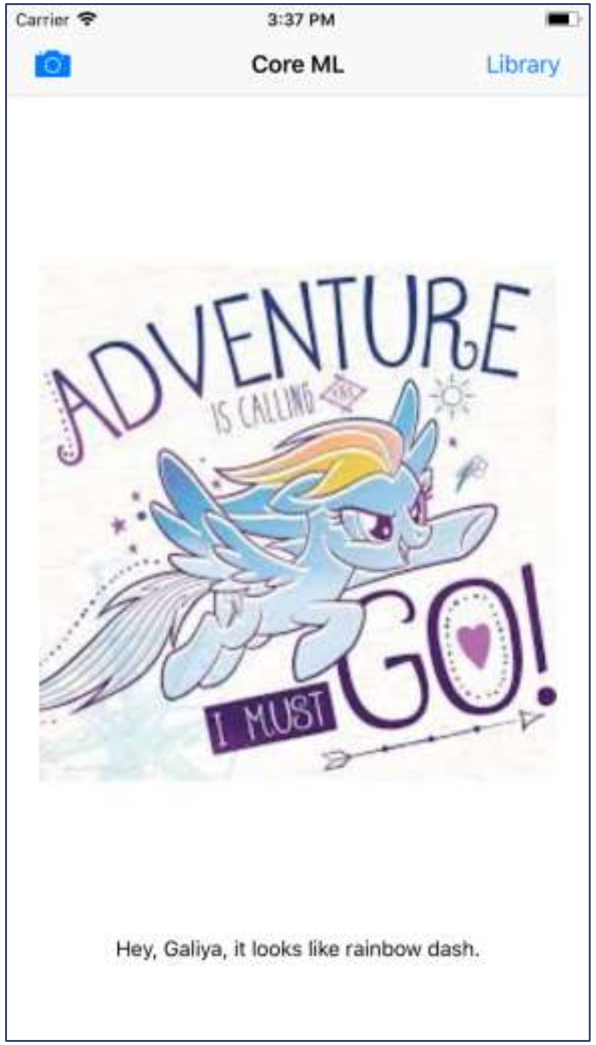


Hey, Galiya, it looks like ~~princess cadence~~.

FAIL!



Hey, Galiya, it looks like princess cadence.

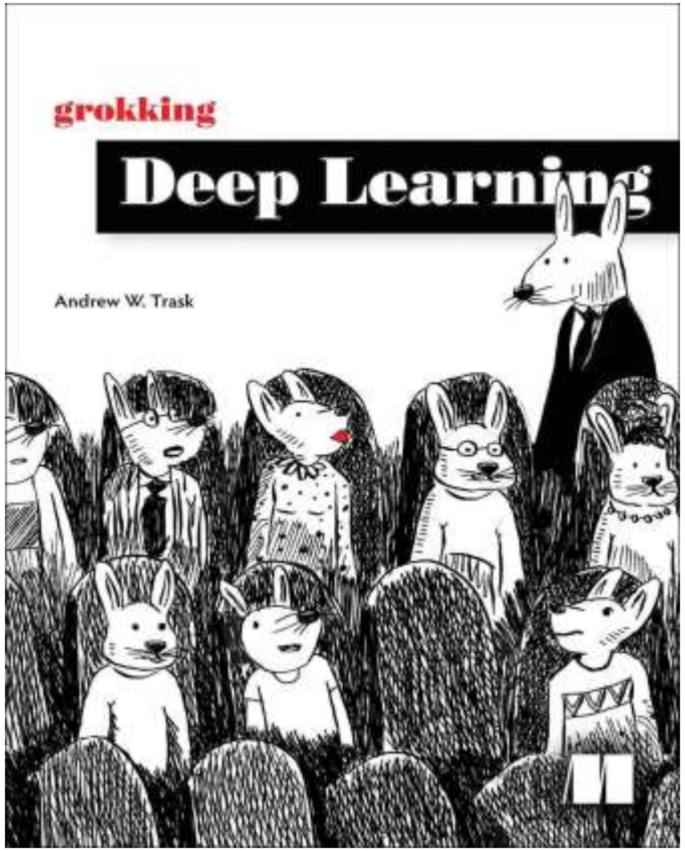


Hey, Galiya, it looks like rainbow dash.

Takeaways

- Only interested in API? Take a look at Custom Vision Service
- Want to build your own classifier?
    - Have good dataset to work with (with lots of samples!)
    - Use Keras to implement CNNs
    - You will need access to GPU (your own / on the cloud)
- Convolutional Neural Networks:
    - Start with naïve approach “from scratch” or use pre-trained models
    - Apply data augmentation to avoid overfitting
    - Introduce dropout layers to avoid overfitting

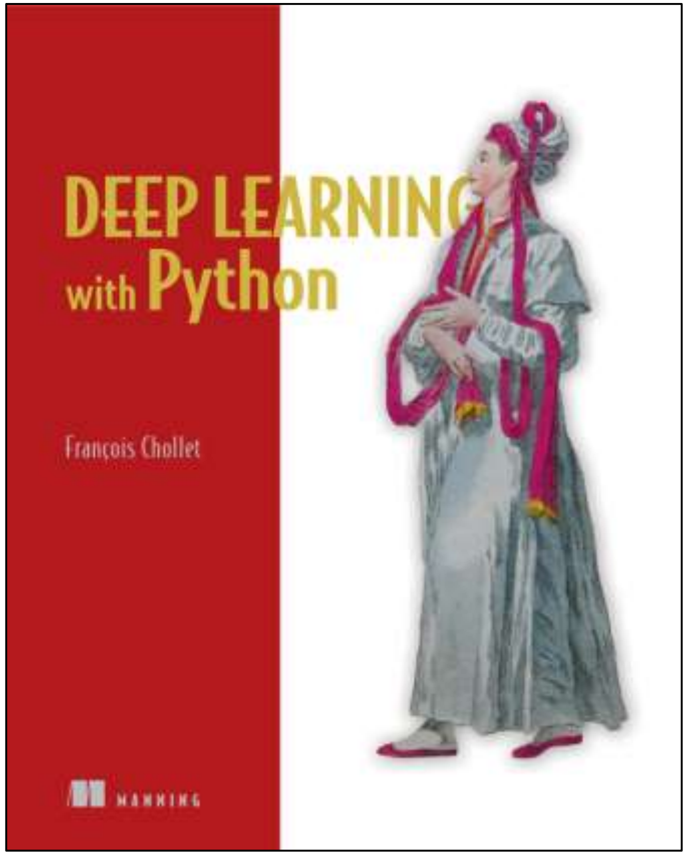
Takeaways: Resources



grokking

Deep Learning

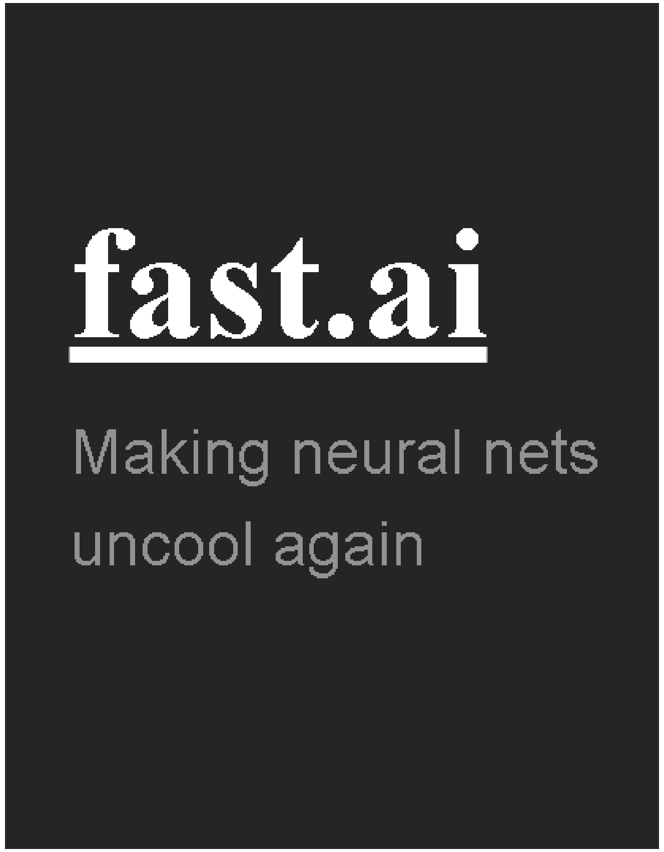
Andrew W. Trask



DEEP LEARNING
with Python

François Chollet

MANNING



fast.ai

Making neural nets uncool again

What’s next?

POKéMON

PAW
PATROL

Thank you!

@galiyawarrier

https://github.com/galiya/NDCLondon2019