



Deep Learning in the world of little ponies

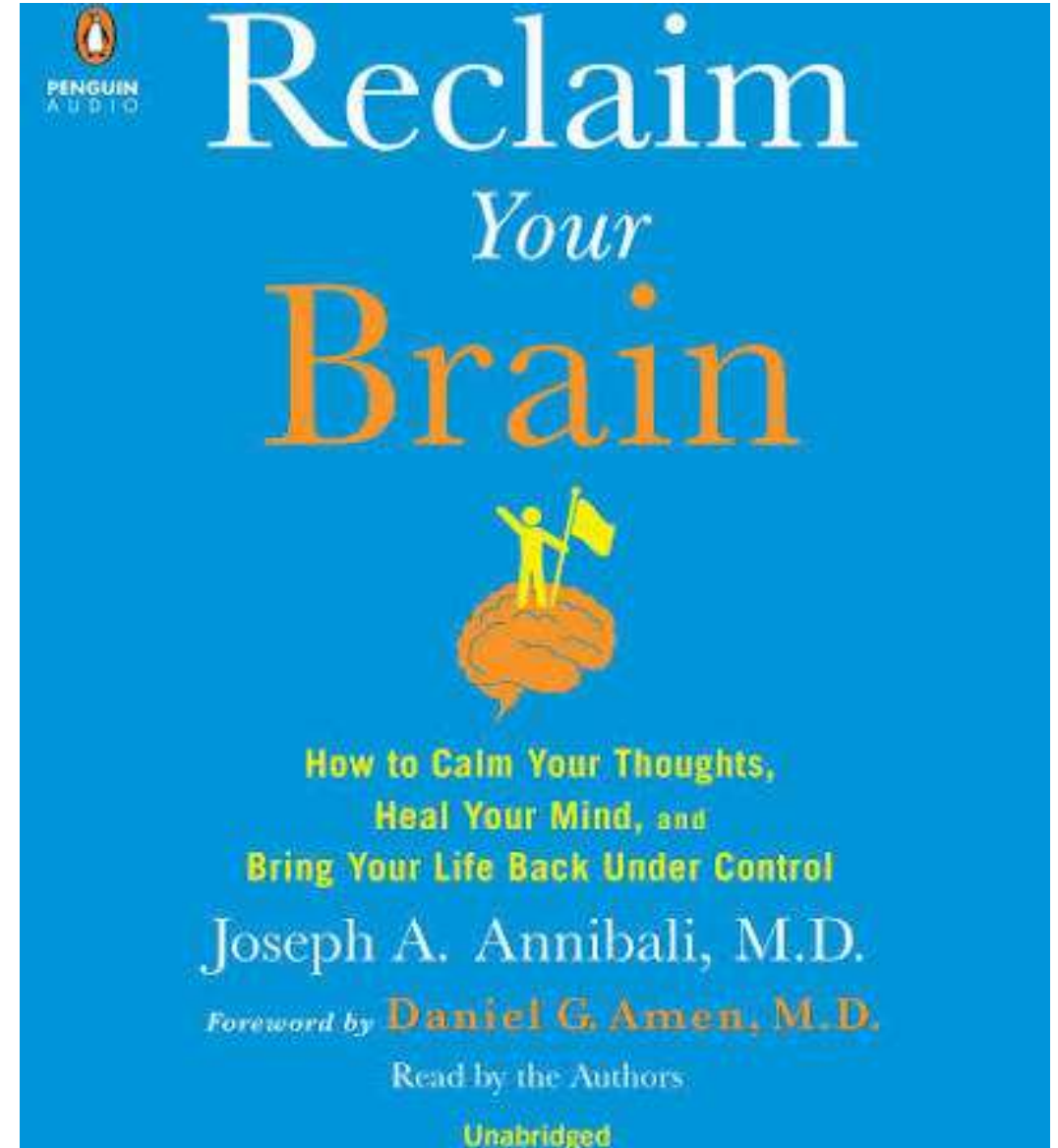
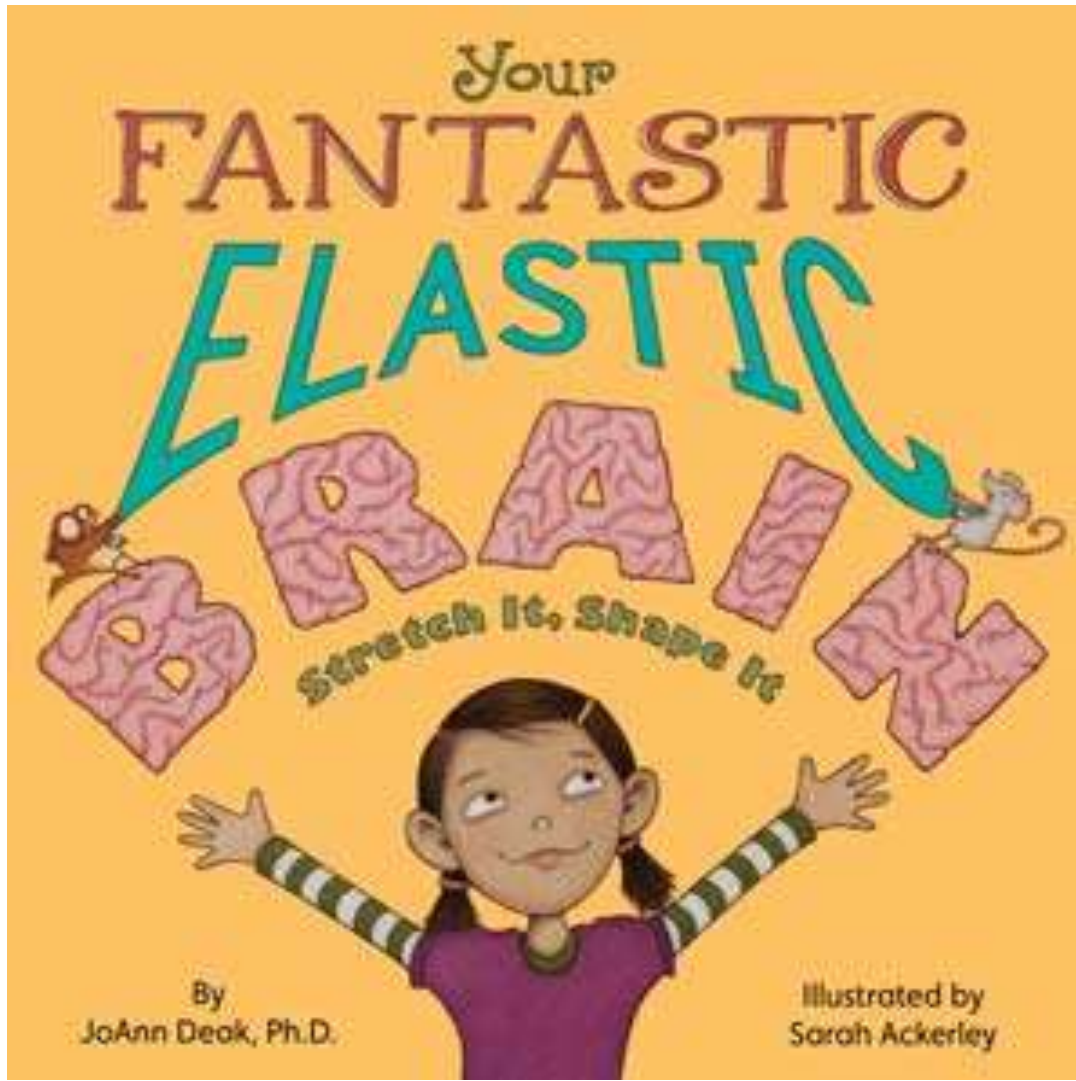
By Galiya Warrior

Galiya Warriar:
Cloud Solution Architect,
Advanced Analytics & AI,
Microsoft,
UK



Maya Warriar:
4.5 years old,
started reception
class 2 weeks ago

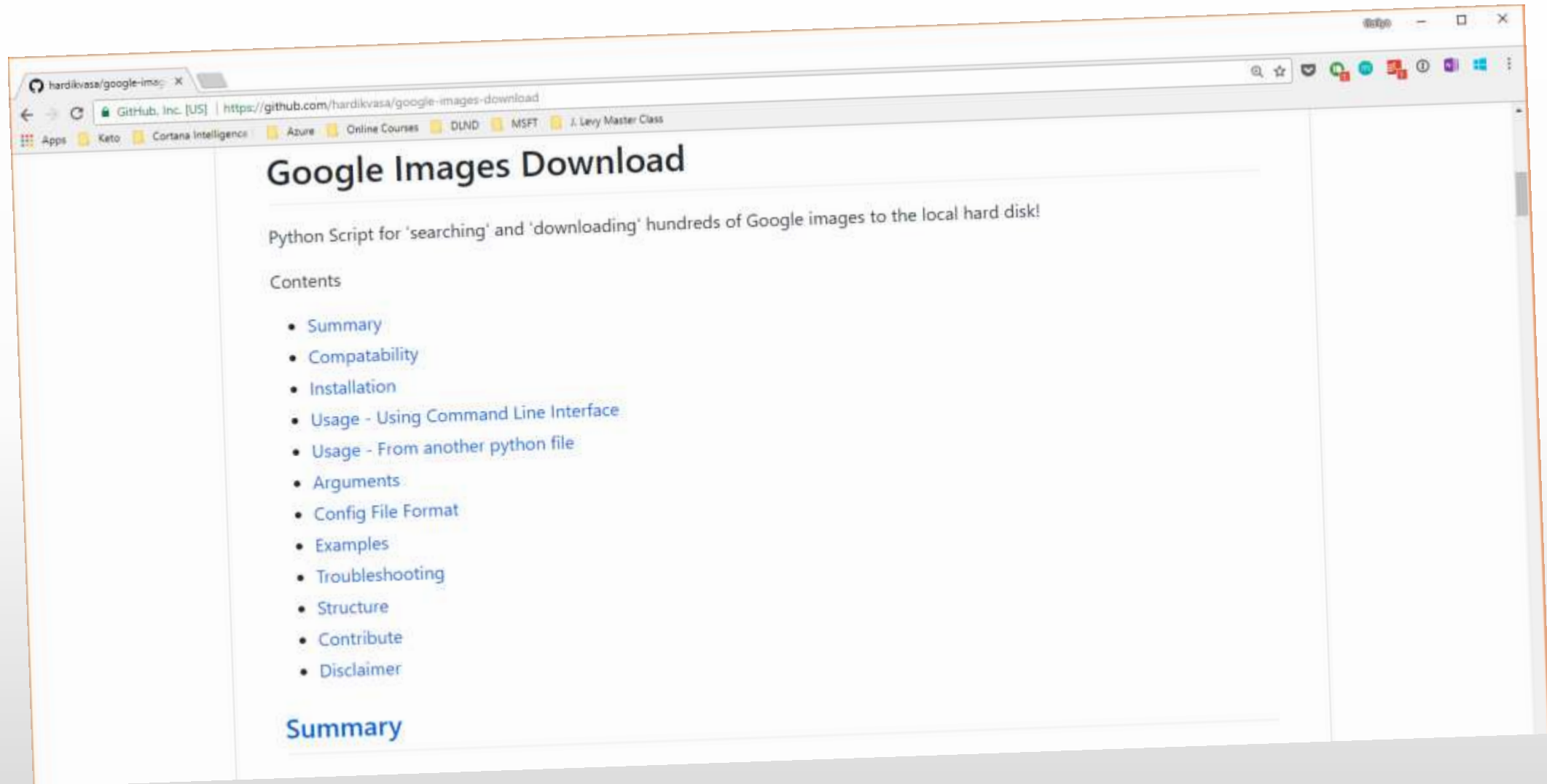






But before starting anything ... DO I HAVE DATA?

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



What should I do now?



Cognitive Services

Use AI to solve business problems



Vision

Image-processing algorithms to smartly identify, caption and moderate your pictures.



Speech

Convert spoken audio into text, use voice for verification or add speaker recognition to your app.



Knowledge

Map complex information and data in order to solve tasks such as intelligent recommendations and semantic search.



Search

Add Bing Search APIs to your apps and harness the ability to comb billions of web pages, images, videos and news with a single API call.

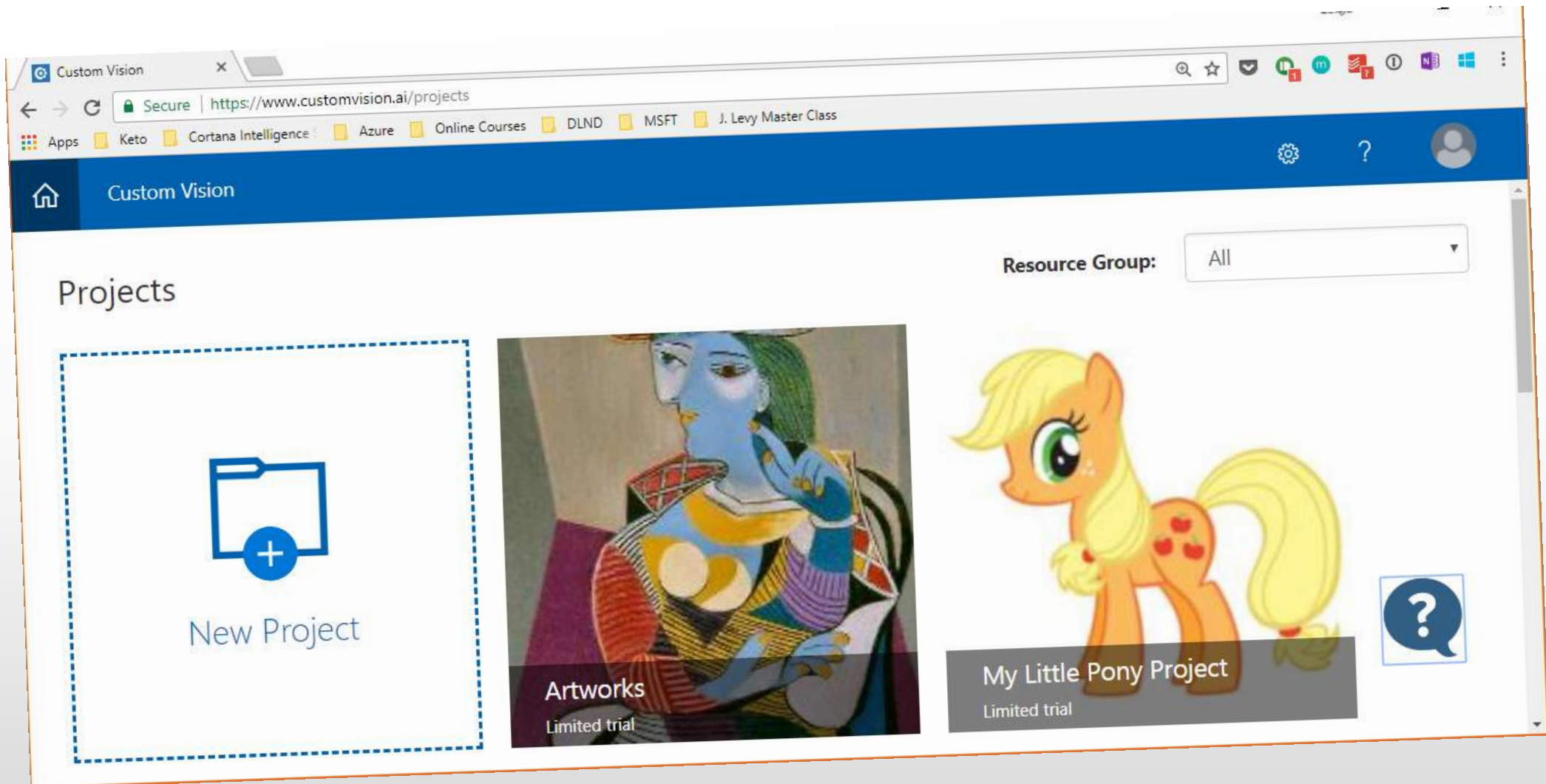


Language

Allow your apps to process natural language with pre-built scripts, evaluate sentiment and learn how to recognise what users want.



<http://aka.ms/cognitive>

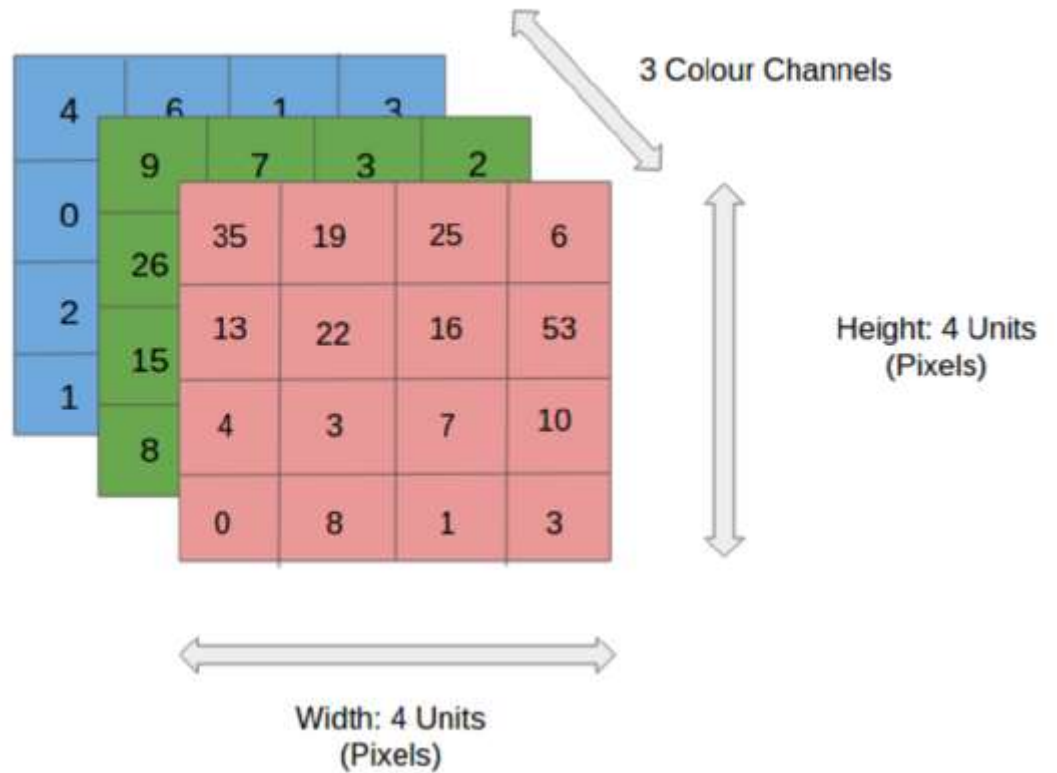






But how does image classification
works under the hood?

What is an image?





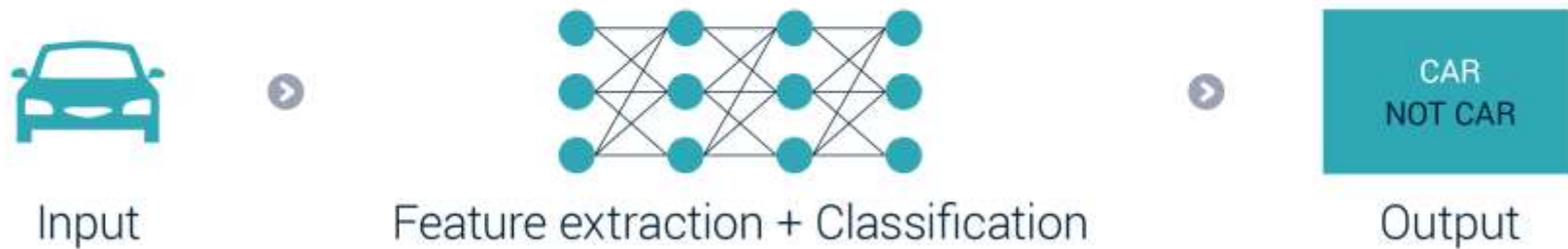
ML: Feature engineering

Classic ML vs. Deep Learning

Machine Learning



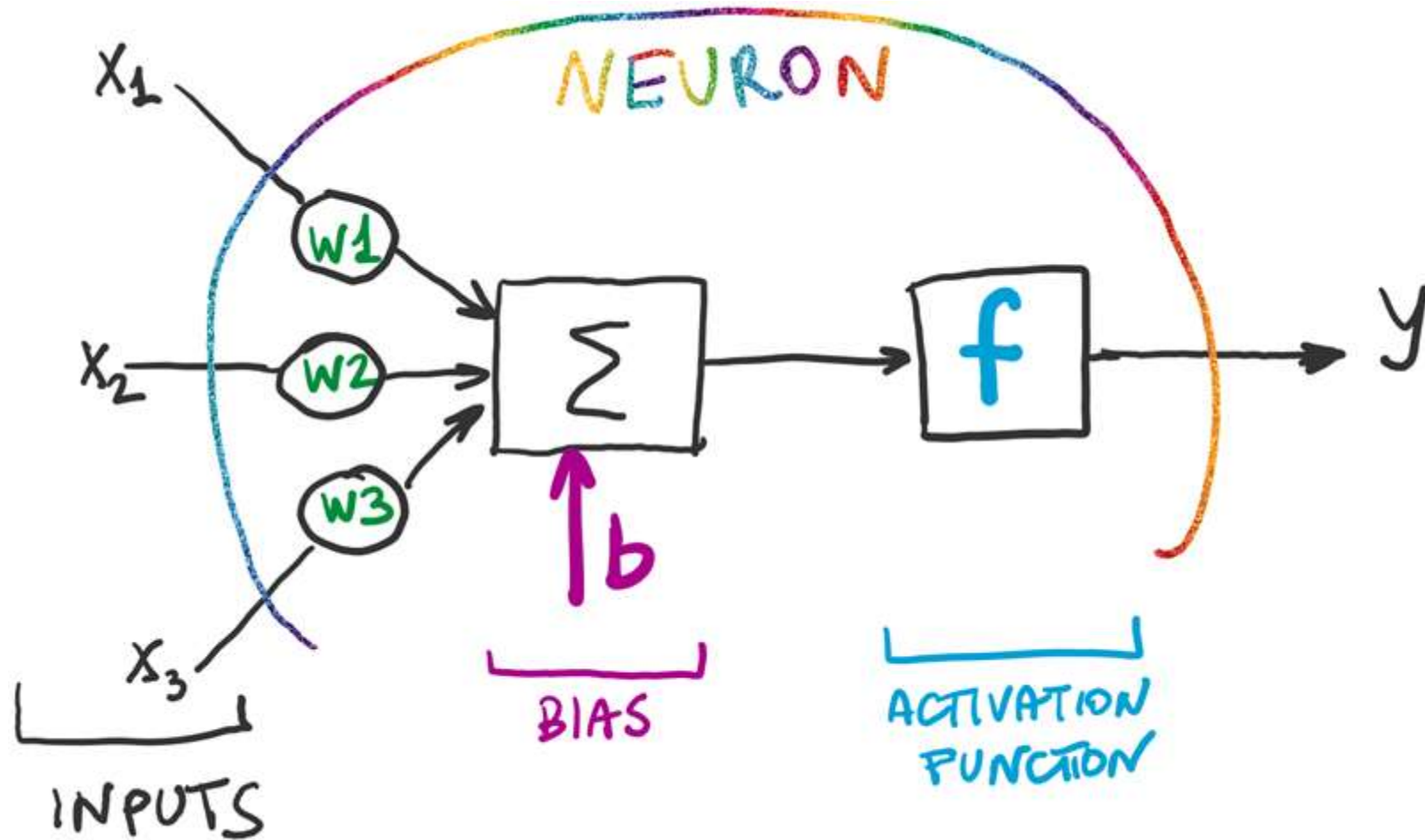
Deep Learning



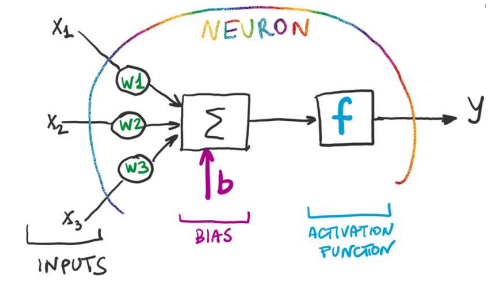


Representations

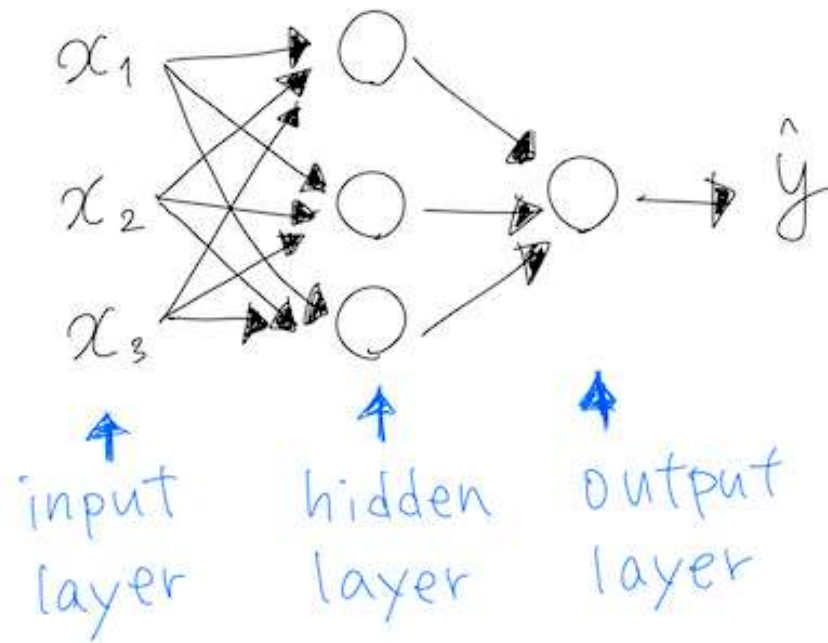
What is a Neuron?



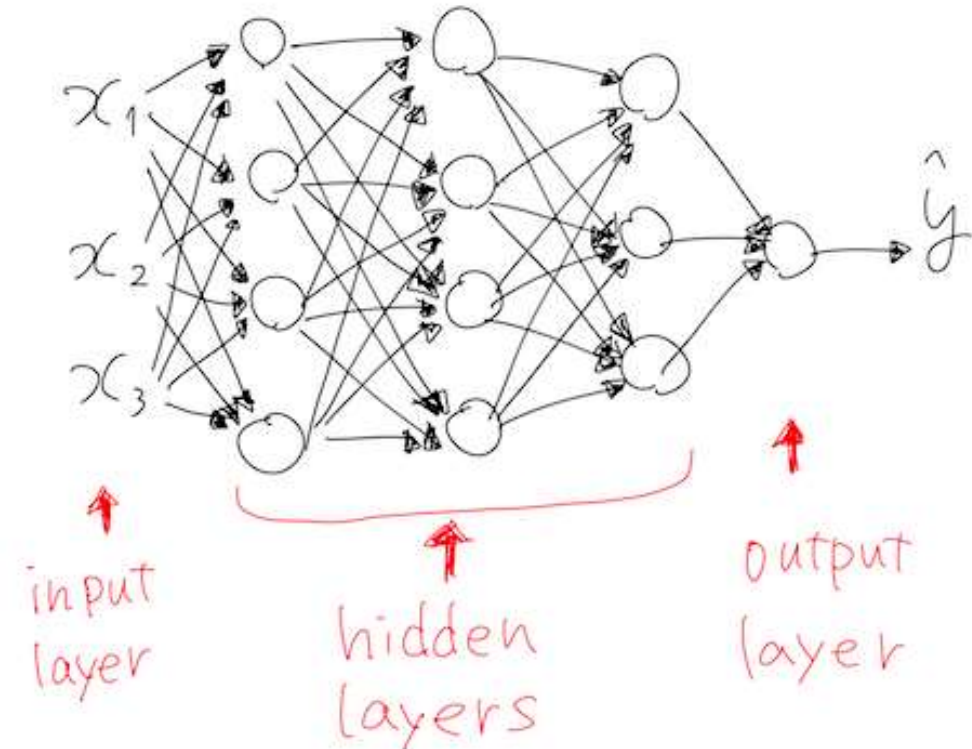
Layers in Neural Network



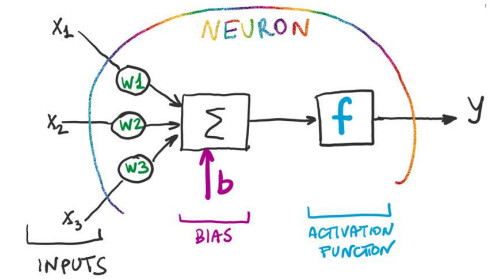
Shallow Neural Network



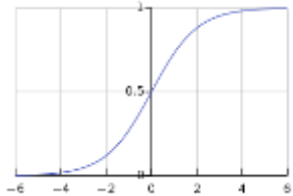
Deep Neural Network



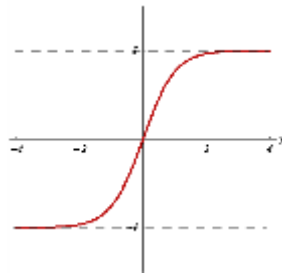
Activation functions



- Good ones are continuous & infinite in domain
- Usually monotonous and don't change direction
- Good ones are usually “non-linear”
- These functions (and their derivatives) should be efficiently computable



Sigmoid



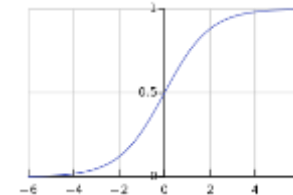
Tanh



ReLU
(Rectified Linear Unit)

NONE

Predict raw data values



Sigmoid
(Unrelated probs)

SOFTMAX

Softmax
 (“which one” prob)

Hidden layer activation functions

Output layer activation functions

Keras: The Python Deep Learning library



Keras

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.



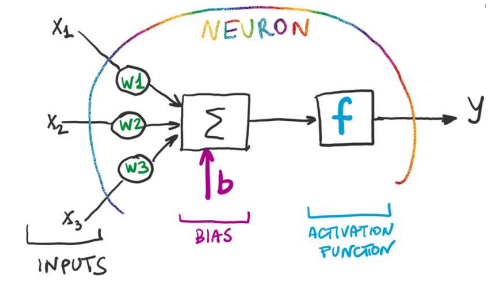
Neural Network Architectures

Densely Connected Neural Networks

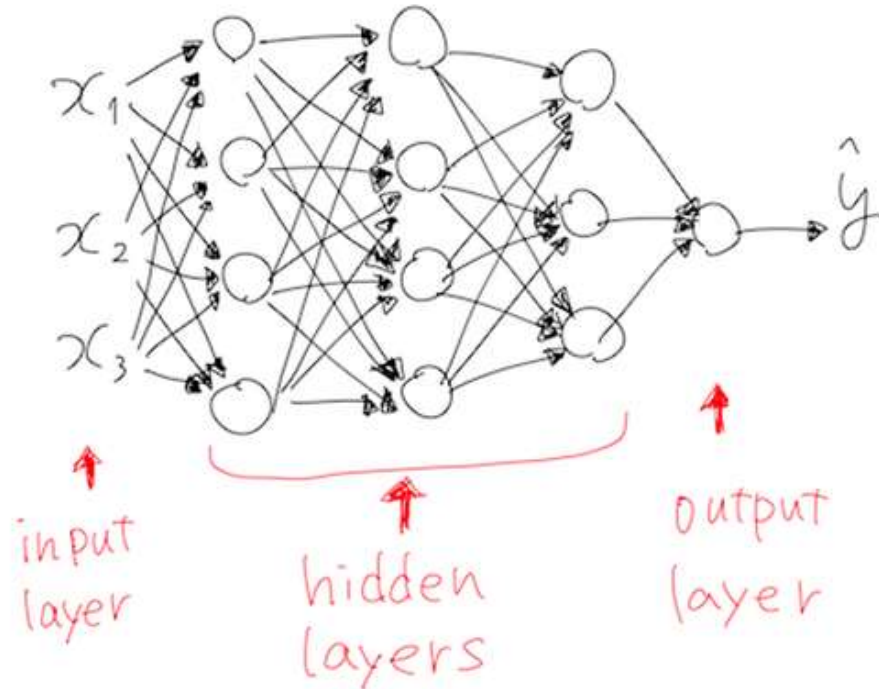
Convolutional Neural Networks
Spatial relationships (i.e. vision)

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

Densely Connected Neural Networks



Deep Neural Network



An example of 4-layer dense, or fully connected, neural network

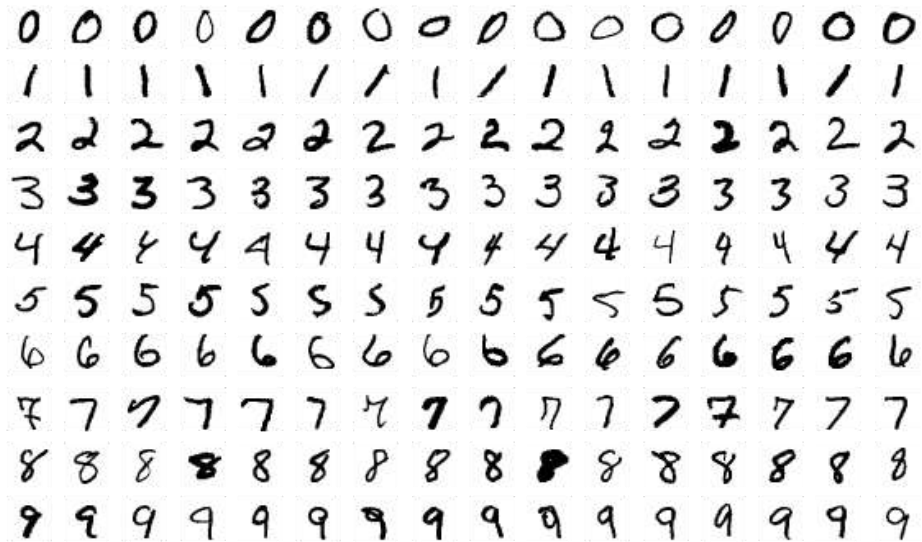
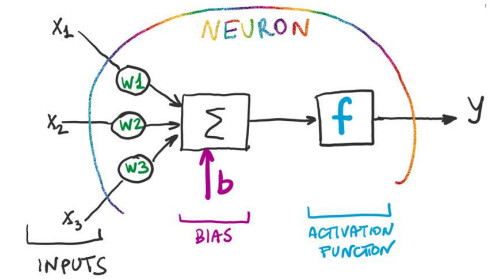
All neurons in any hidden layer have full connections to all activations in the previous layer

Number of parameters on each layer is huge

Model's capacity = number of learnable parameters



Densely Connected Neural Networks: MNIST Classification



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

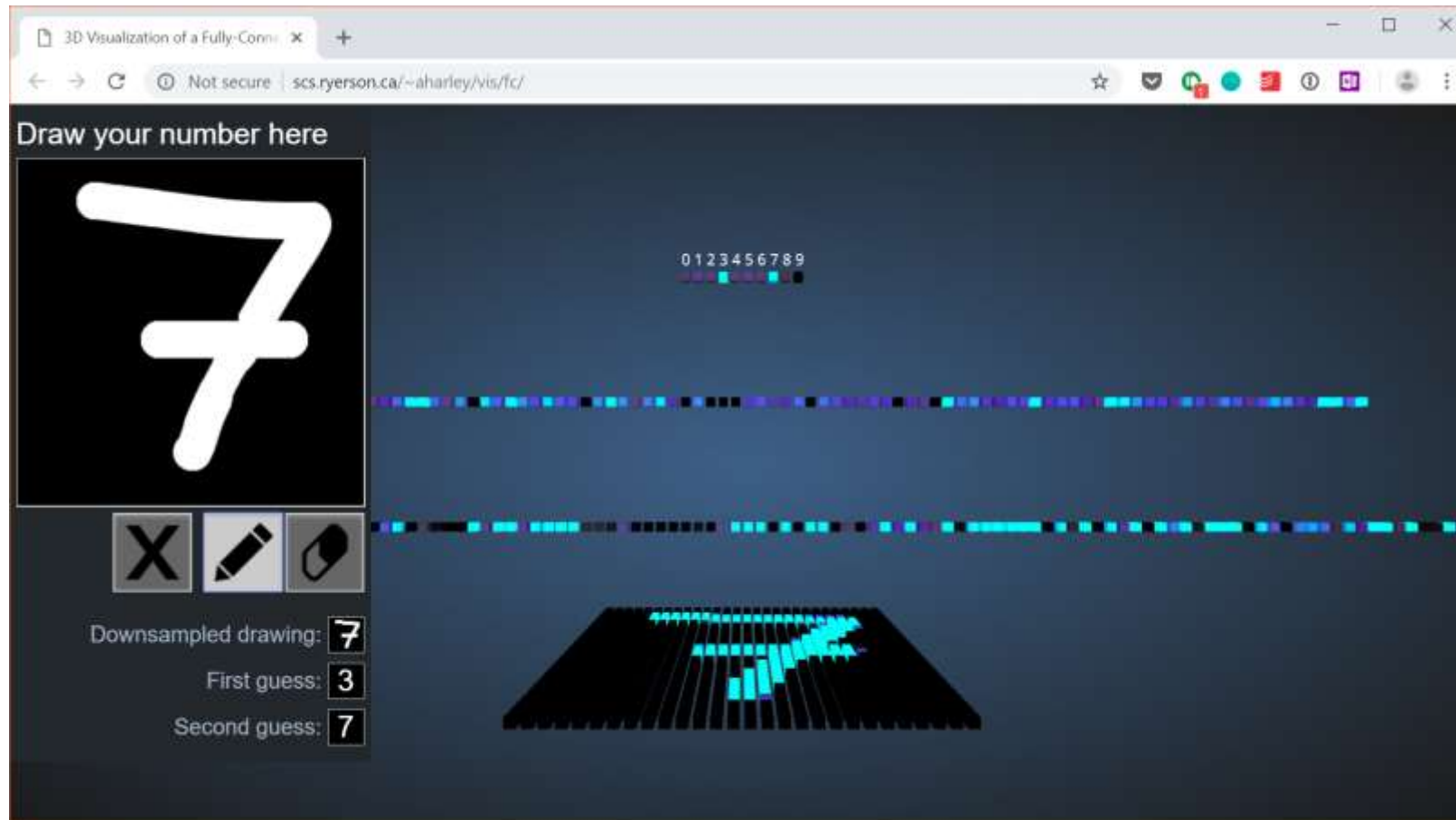
Large database of handwritten digits (28x28)

60,000 training and 10,000 testing images

“Hello World” of computer vision tasks



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





Biological inspiration:

- * Visual cortex in animals and humans
- * Cells are sensitive to small subregions of the input

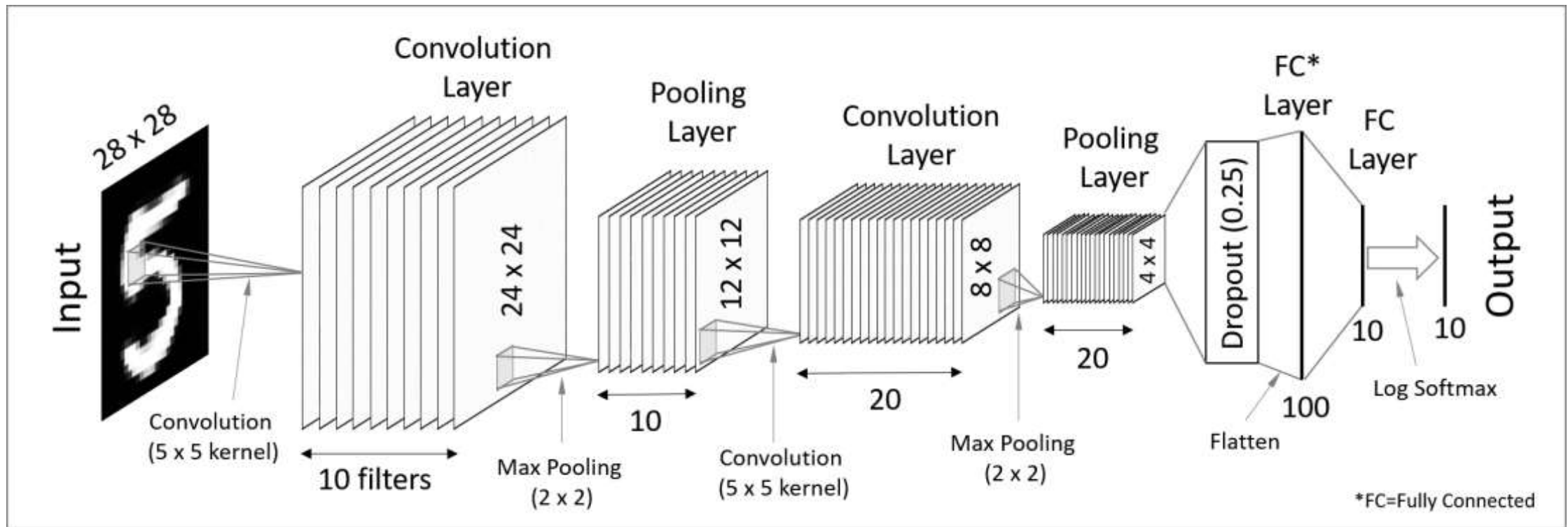
Important for CNNs:

- * Structure to input data
- * Spatial relationships
- * Repeated patterns

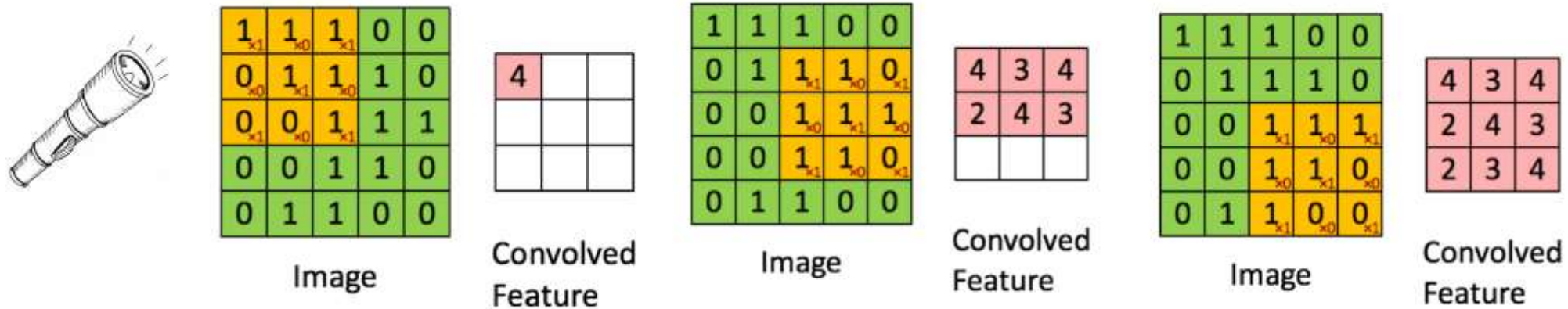
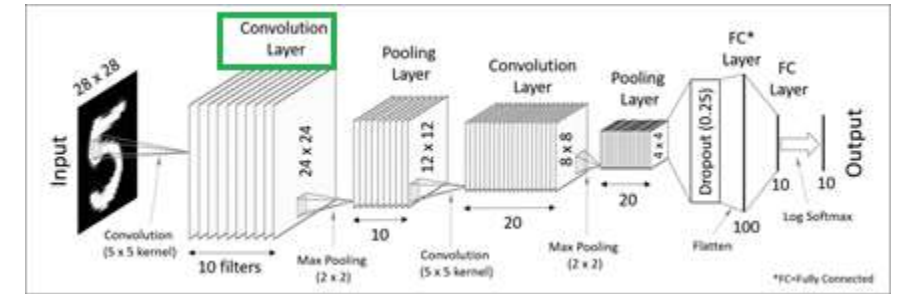
Commonly used for computer vision applications

Why Convolutional Neural Networks?

CNNs: Typical architecture



CNNs: Convolution Layer

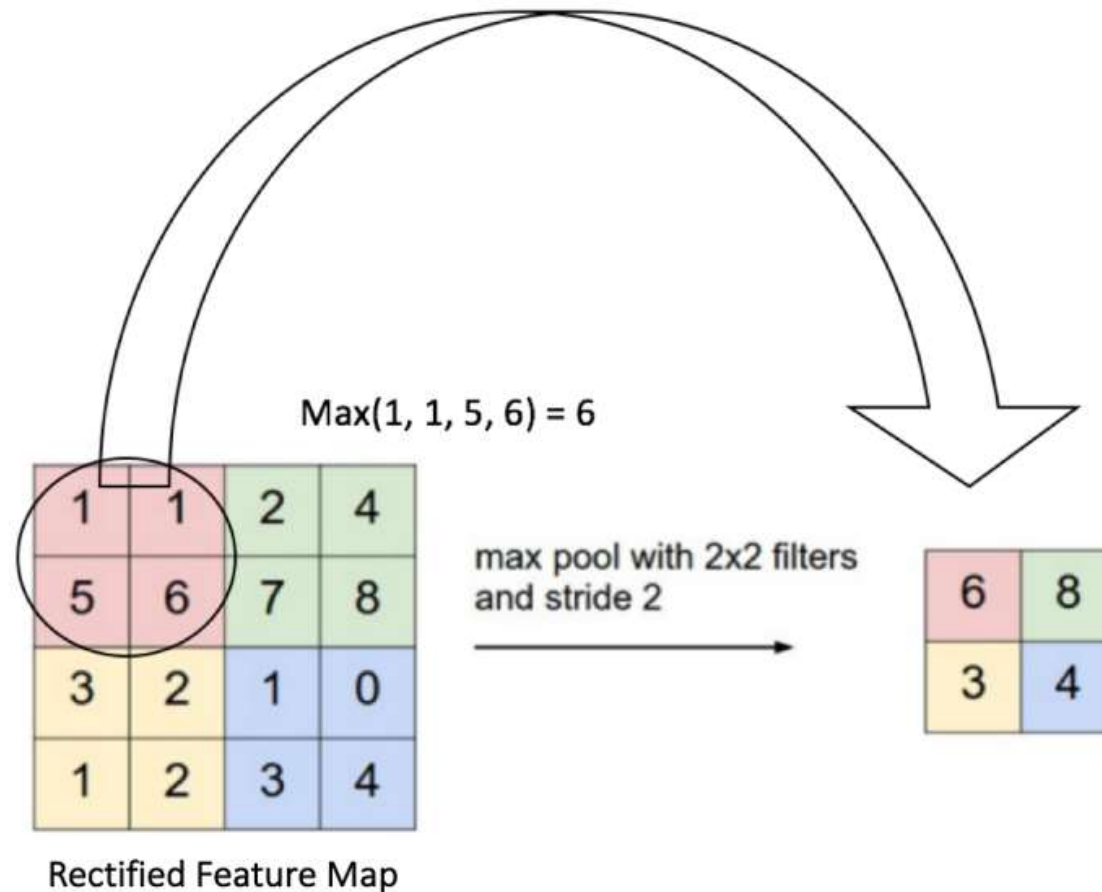
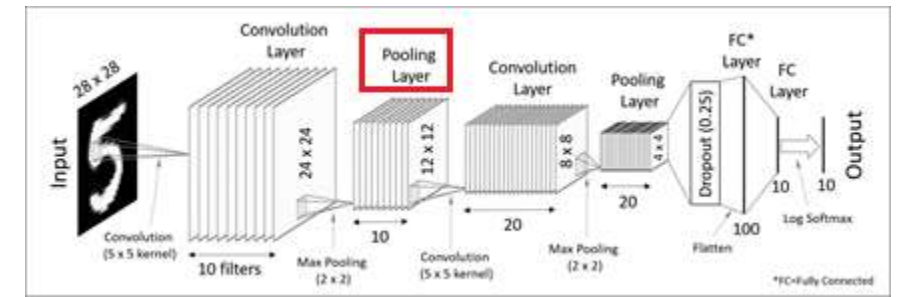


Extract features from input image

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

Multiple filters

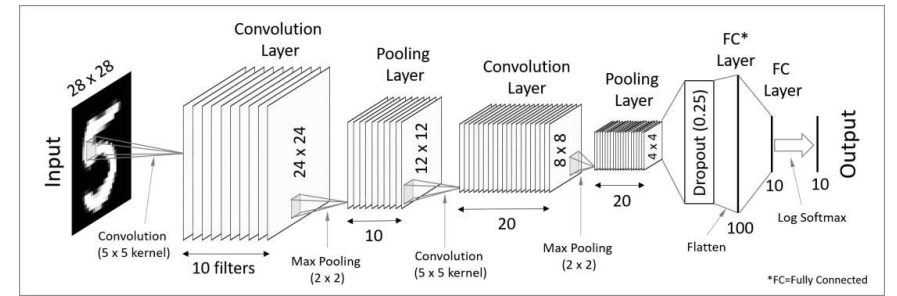
CNNs: Pooling layer



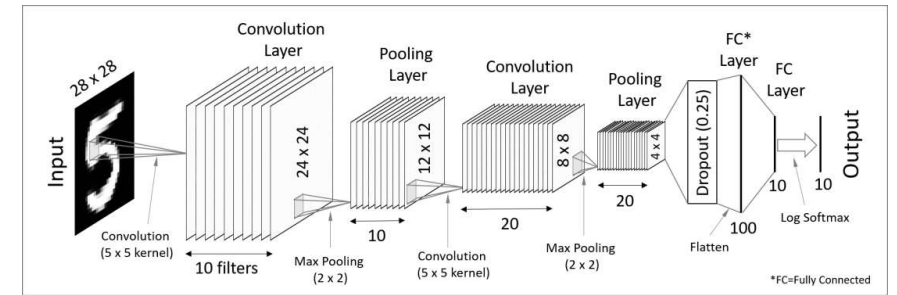
Reduce dimensionality BUT
retain most important information

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]



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?

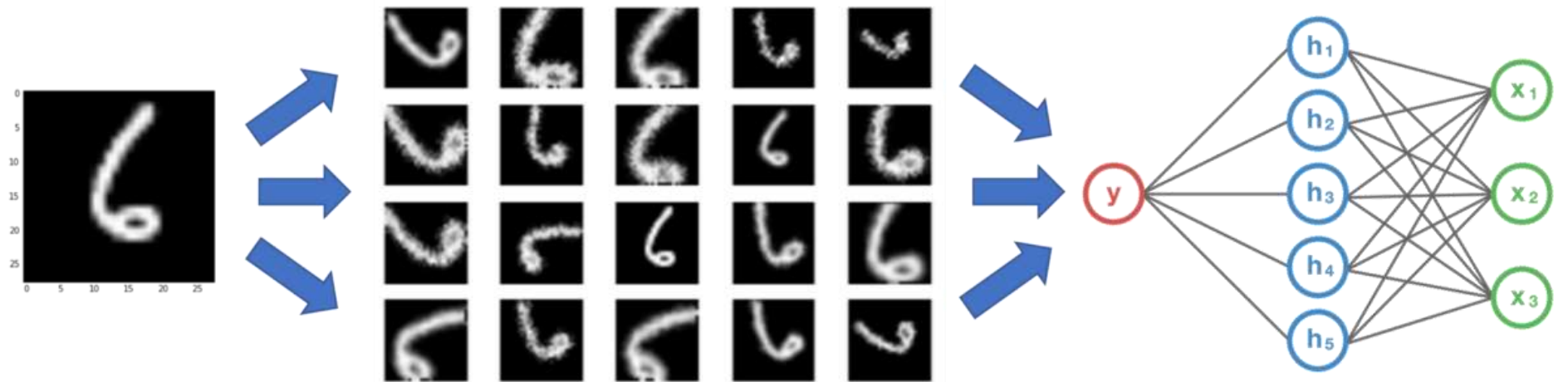
More training data always helps to reduce “memorisation”

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



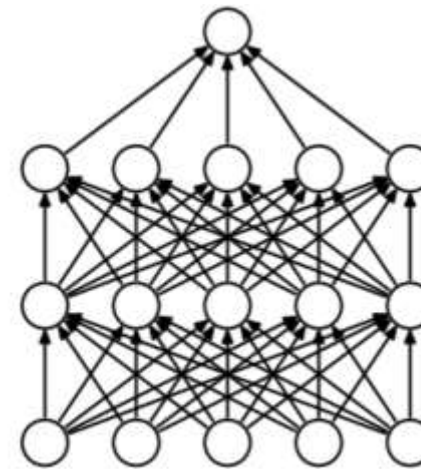
Additional techniques: Dropout

“Learn from less to learn better”

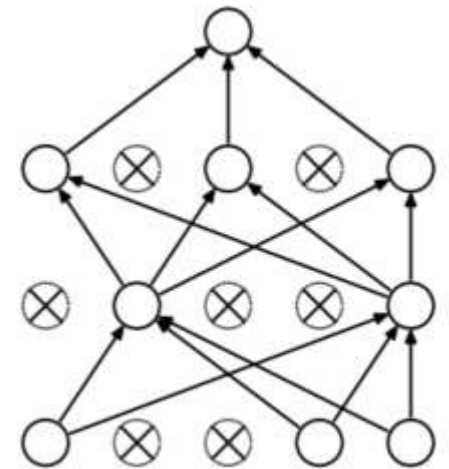
Reduces overfitting

Learn more robust features

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

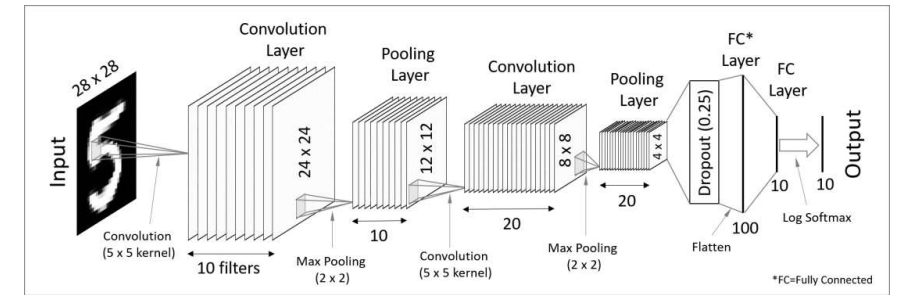


(a) Standard Neural Net



(b) After applying dropout.

Demo: Build custom pony classifier from scratch



Input

A folder with 6 subfolders of images

150px * 150px each

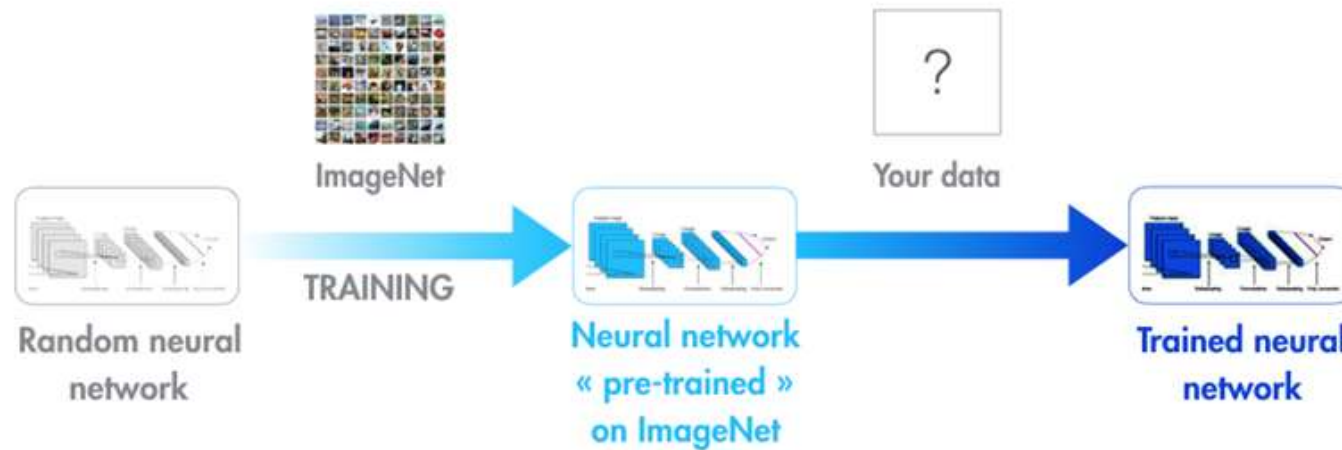
Output

Number of classes – 6

Names: (on the right)



Transfer learning



Lots of existing pre-trained networks are available:

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

Demo: Use pre-trained models for classification task

Input

A number of personal images

Various sizes

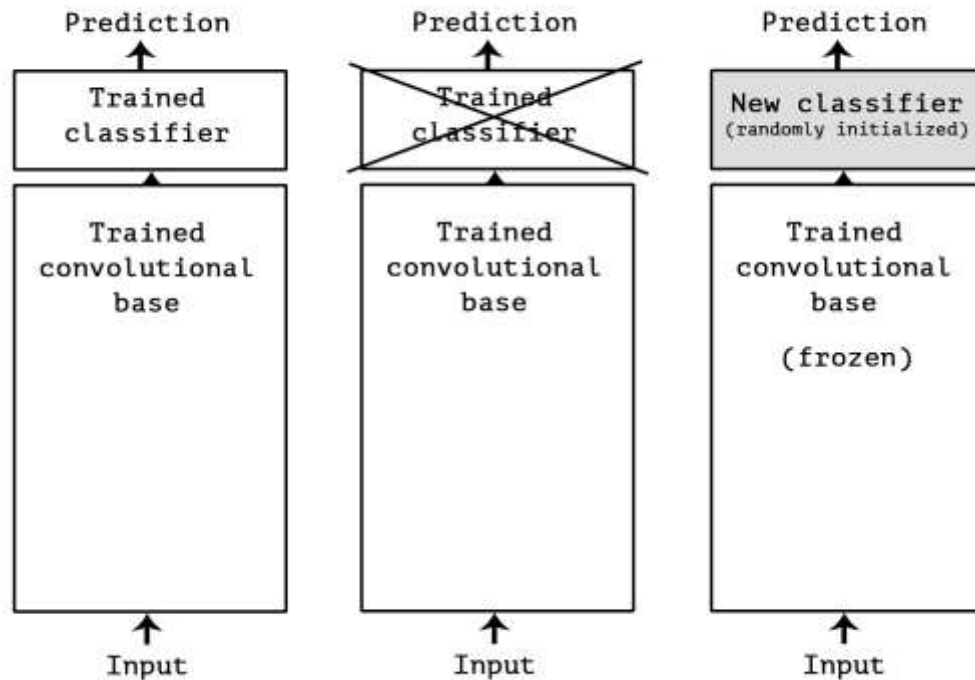
Output

Image class (as per pre-trained model)

Probability



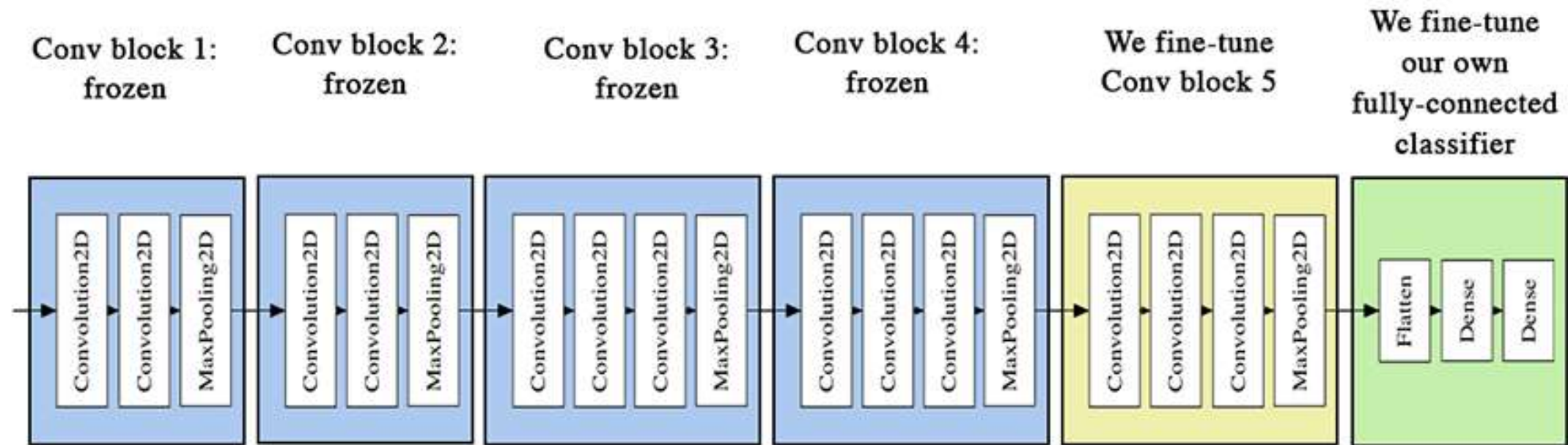
Pre-trained models: Feature extraction



```
1 from keras.preprocessing import image
2 from keras.applications.vgg16 import preprocess_input, decode_predictions, VGG16
3 import numpy as np
4 import os
5
6 conv_base = VGG16(weights = 'imagenet', include_top = False, input_shape=(150,150,3))
```

```
1 from keras import models
2 from keras import layers
3 conv_base.trainable = False
4 model = models.Sequential()
5 model.add(conv_base)
6 model.add(layers.Flatten())
7 model.add(layers.Dense(256, activation='relu'))
8 model.add(layers.Dense(3, activation='softmax'))
```

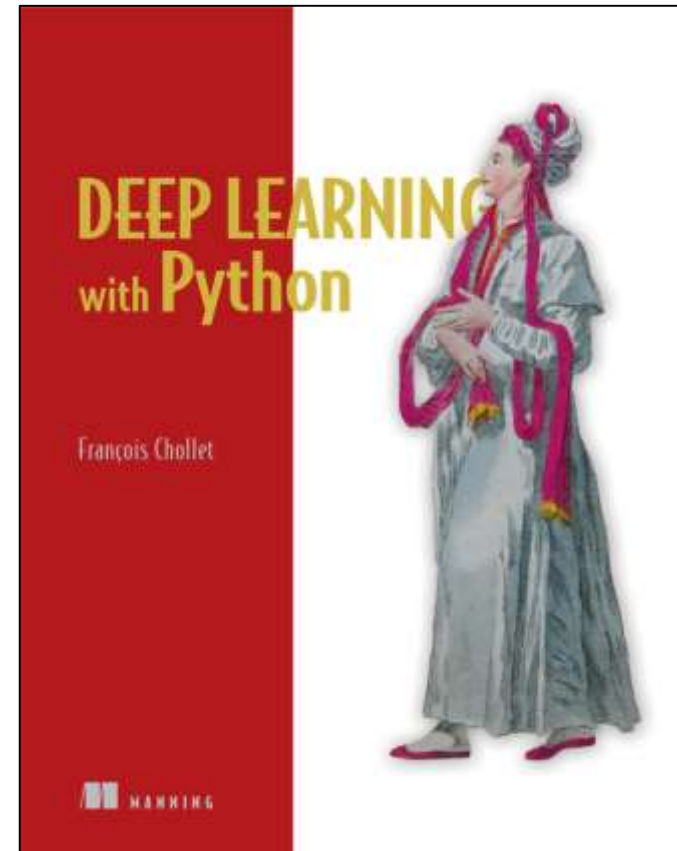
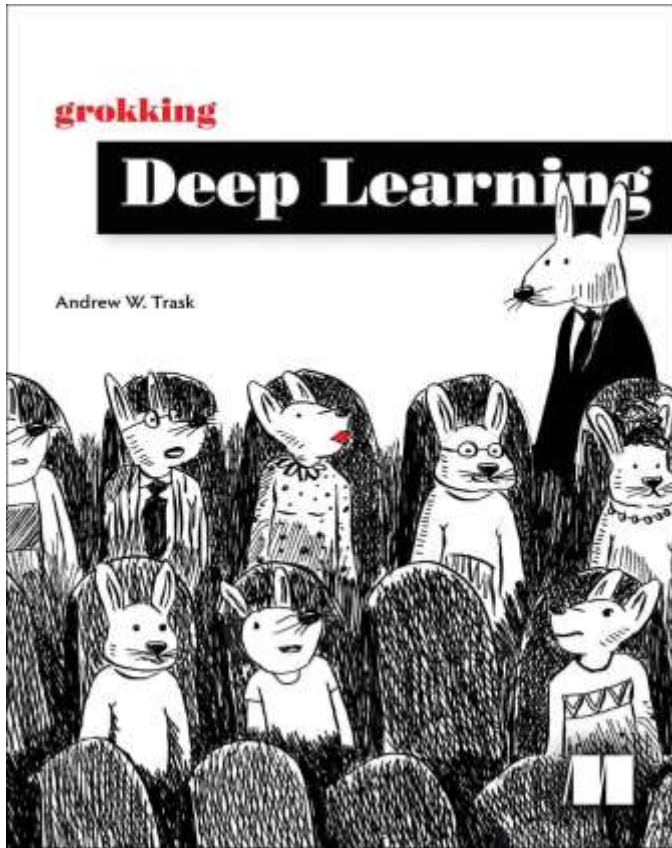
Pre-trained models: Fine-tuning



Takeaways

- ◉ Only interested in API? [Custom Vision Service](#) from Cognitive Services suite
- ◉ Want to build your own classifier?
 - ◉ Have good dataset to work with (with lots of samples!)
 - ◉ Use [Keras](#) to implement CNNs
 - ◉ Need access to GPUs? Use [Azure Databricks](#) (ML Runtime)
- ◉ 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



What's next?



Thank you!

And enjoy #NDCSydney!