Deep Learning with Python

[Class Materials]

http://bit.ly/dlp_2021

DAY 1 ANN AND CNN

Mr Seow Khee Wei / Dr Jimmy Goh

Introduction of trainer

Mr Seow Khee Wei

Name
Seow Khee Wei

Telegram
@kwseow

Email
seow_khee_wei@rp.edu.sg

Dr Jimmy Goh

Name
Jimmy Goh

Telegram
@jimmygohRP

Email
Jimmy_goh@rp.edu.sg

Programmes

Day 1	Lecture 1 – Artificial Neural Network Activity 1 – Predicting HDB resale price Activity 2 – Classifying Greyscale Handwritten Digits Activity 3 – Classifying Fashion Lecture 2 – Image Convolution	Activity 4 – Classifying Fashion with CNN Activity 5 – Classifying Flowers with CNN Lecture 3 – Pre-trained model and transfer learning Activity 6 – Classifying Flowers with pretrained VGG16 Activity 7 – Classifying Flowers with transfer Learning using pretrained VGG16
Day 2	Lecture 1 – Recurrent Neural Network & Long Short-Term Memory Network Activity 1 – Time Series Prediction with MLP Activity 2 – Time Series Prediction with LSTM Activity 3 – Sequence Classification of Movie Review	Lecture 2 – Generative Adversarial Networks Activity 4 – Develop a 1D GAN Activity 5 – Develop a DCGAN for Greyscale Handwritten Digits

Prerequisites

Familiar with

- Python Language
- Virtual Environment

We explore different Neural Network Architectures but not specific to optimizing or hyperparameter tuning.

Workshop materials: bit.ly/dlwp_2021

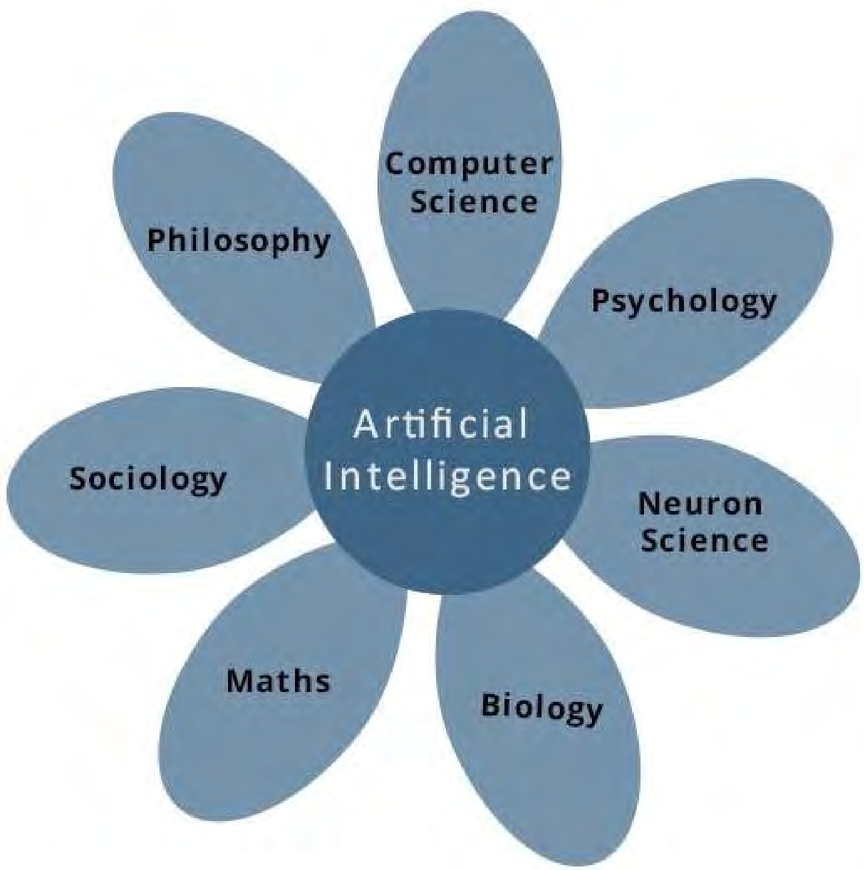
Artificial Neural Network

Artificial Intelligence

- Artificial Intelligence (AI) is the science and engineering of making **intelligent machines**, especially intelligent computer programs.
  - John McCarthy, Stanford University, “Father of AI”, 1956
- "A computer would deserve to be called intelligent if it could **deceive a human into believing that it was human**.“
- "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.“
  - Alan Turing

Artificial Intelligence

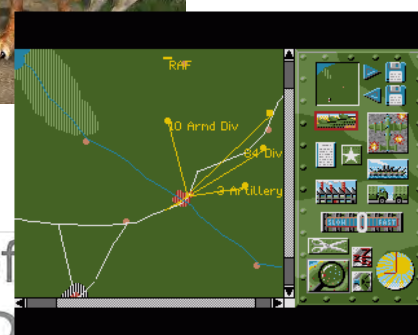
- Contribution to AI

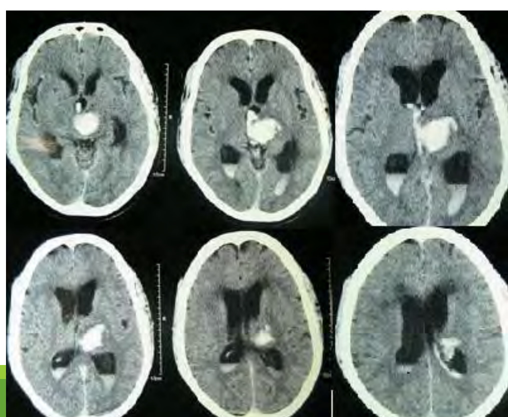


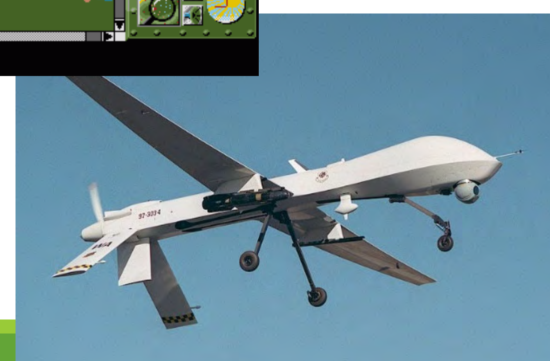
Ref: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_overview.htm

Artificial Intelligence

- Applications of AI
  - Gaming
  - Natural Language Processing
  - Expert Systems
  - Vision Systems
  - Speech Recognition
  - Handwriting Recognition
  - Intelligent Robots
  - Reasoning and awareness
  - Date Analytic

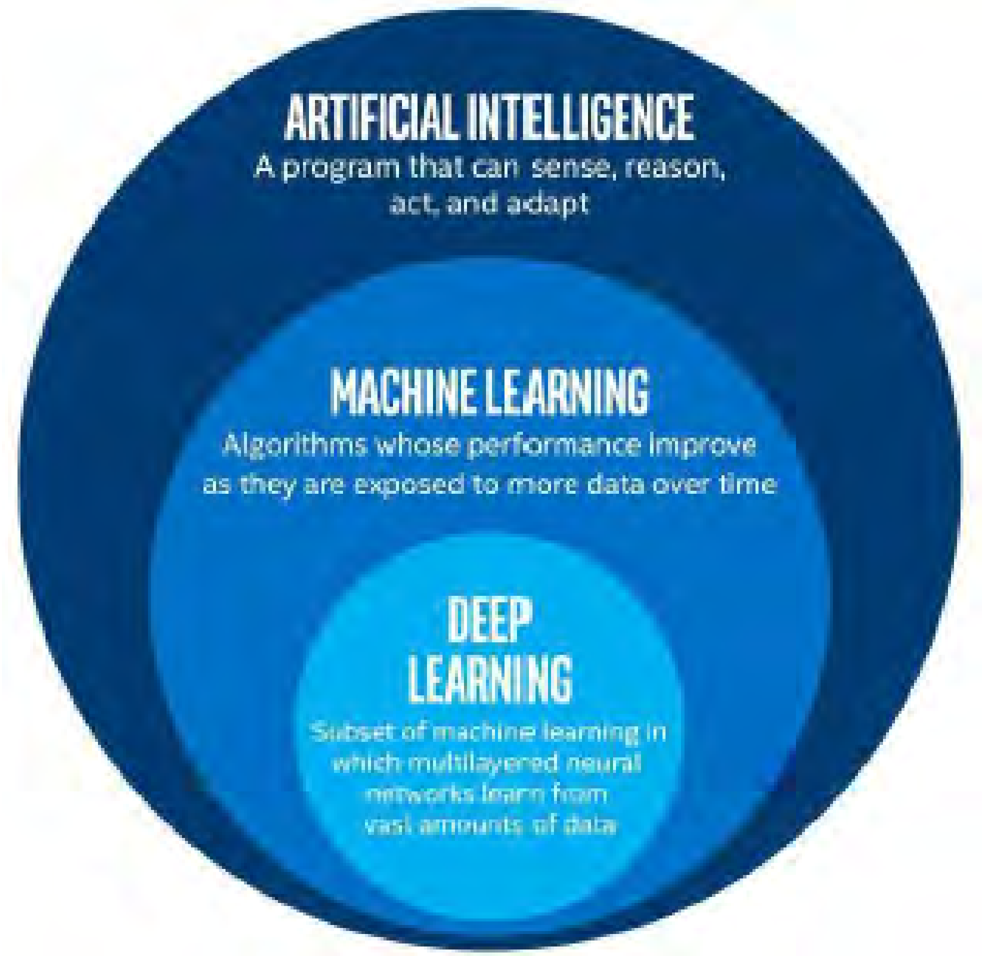






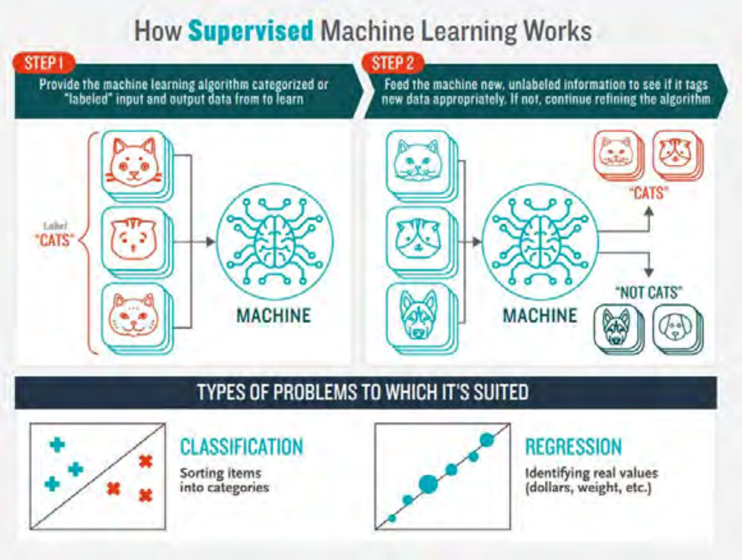
Machine Learning

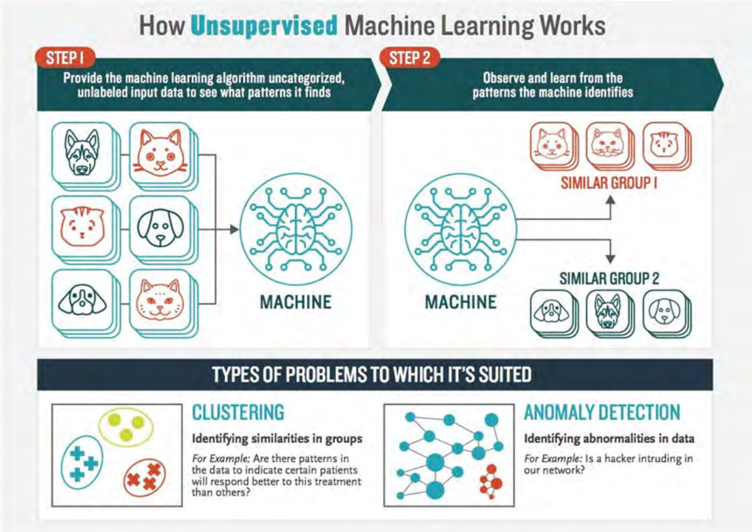
- These programs learn from repeatedly seeing data, rather than being explicitly programmed by humans



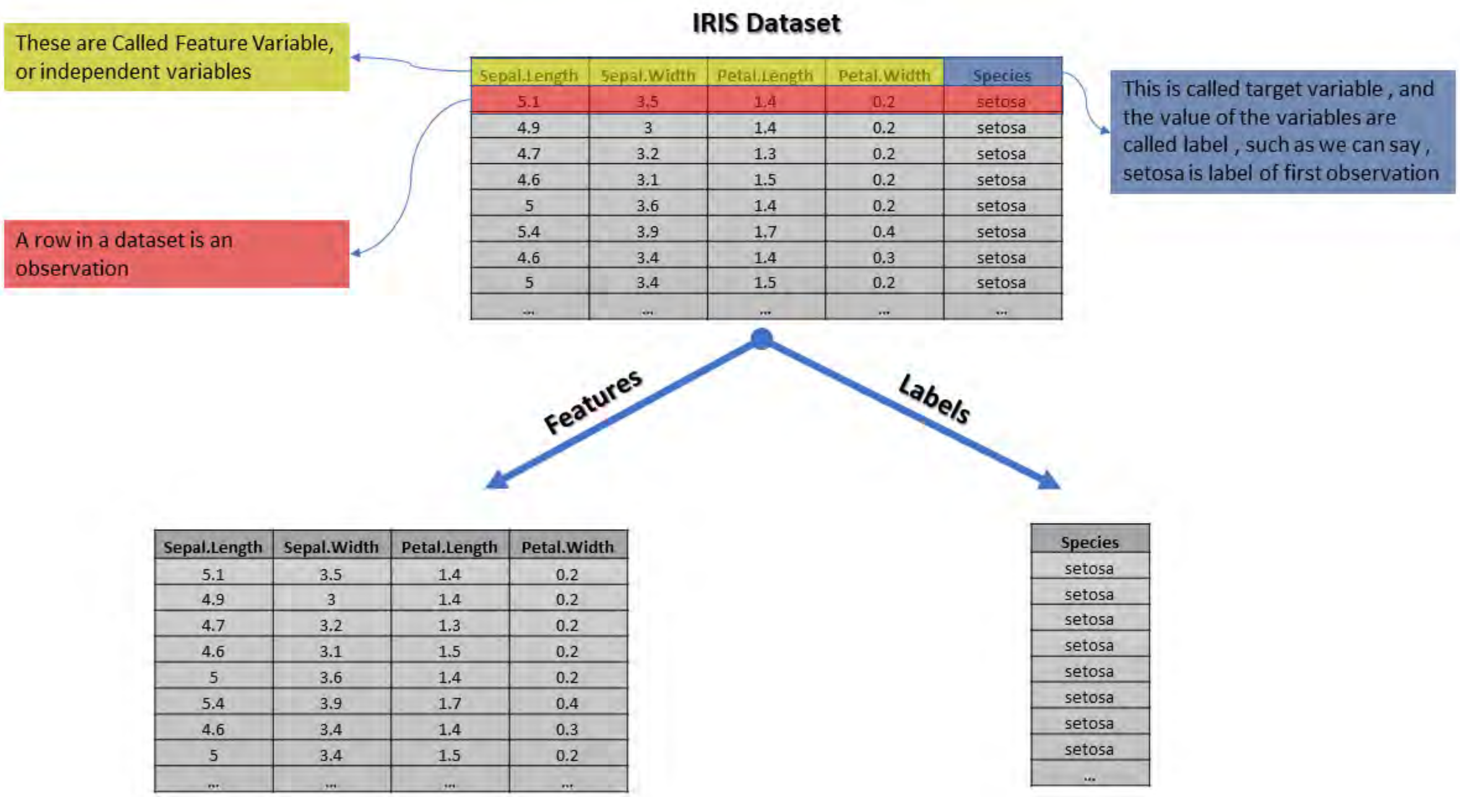
Machine Learning

- **Two main types of learning**
  - Supervised Learning
    - Data points have known outcome
    - Goal is to make predictions - Classify and Regression
  - Unsupervised Learning
    - Data points have unknown outcome
    - Goal is to find structure within the data – Clustering
- **Other types of learning**
  - Reinforcement Learning
  - Genetic Algorithm





Supervised Learning example



Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
...	...	...	...	...

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5	3.4	1.5	0.2
...	...	...	...

Species
setosa
setosa
setosa
setosa
setosa
setosa
setosa
setosa
...

https://www.iroblack.com/2020-07-07-supervised-learning-with-scikit-learn/

Machine Learning

- Applications in our daily lives

Spam Filtering	Web Search	Postal Mail Routing
Fraud Detection	Movie Recommendations	Vehicle Driver Assistance
Web Advertisements	Social Networks	Speech Recognition

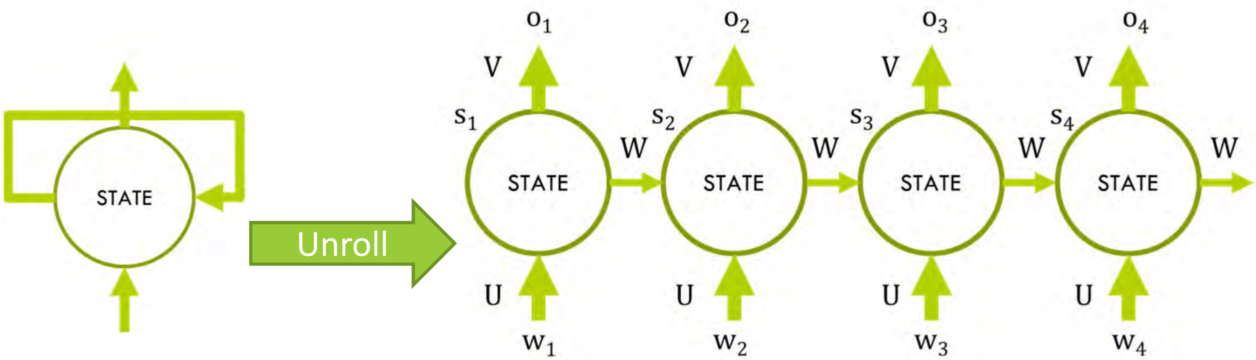
Deep Learning

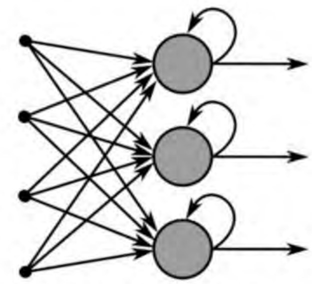
- Deep learning is a class of machine learning algorithms that:
  - use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
  - learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
  - learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

Ref: https://en.wikipedia.org/wiki/Deep_learning#Deep_learning_revolution

Deep Learning

- Deep Neural Network
    - Convolutional Neural Network (CNN)
    - Recurrent Neural Network (RNN)

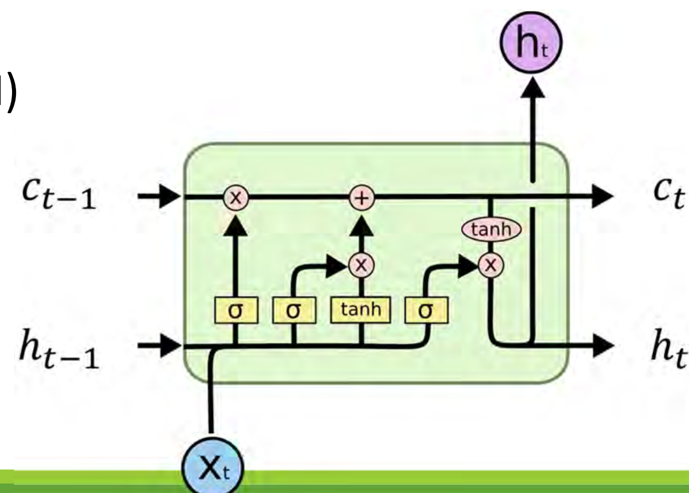




Recurrent Neural Network

https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5

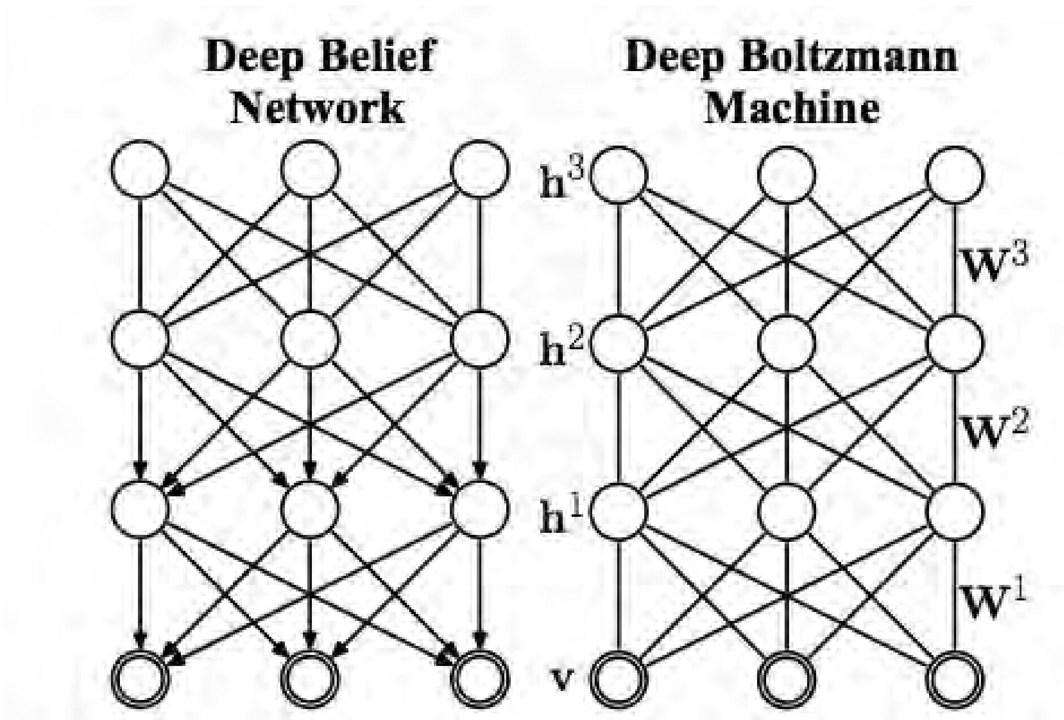
- Long-Short Term Memory (LSTM)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Deep Learning

- Deep Believe Network and Deep Boltzmann Network
  - Based on Restricted Boltzmann Machine (RBM)
  - RMB applied Bayes Theorems into ANN.
  - Unsupervised Learning

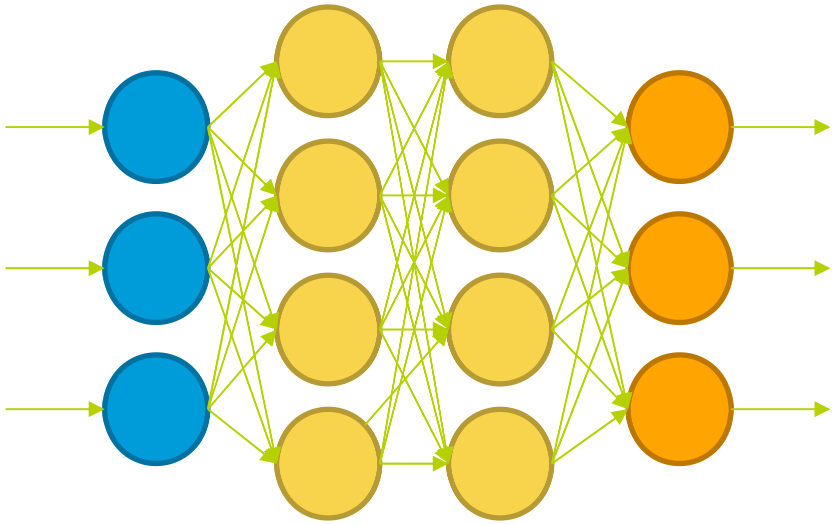


https://www.youtube.com/watch?v=MnGXXDjGNd0

https://www.slideshare.net/zukun/p05-deep-boltzmann-machines-cvpr2012-deep-learning-methods-for-vision

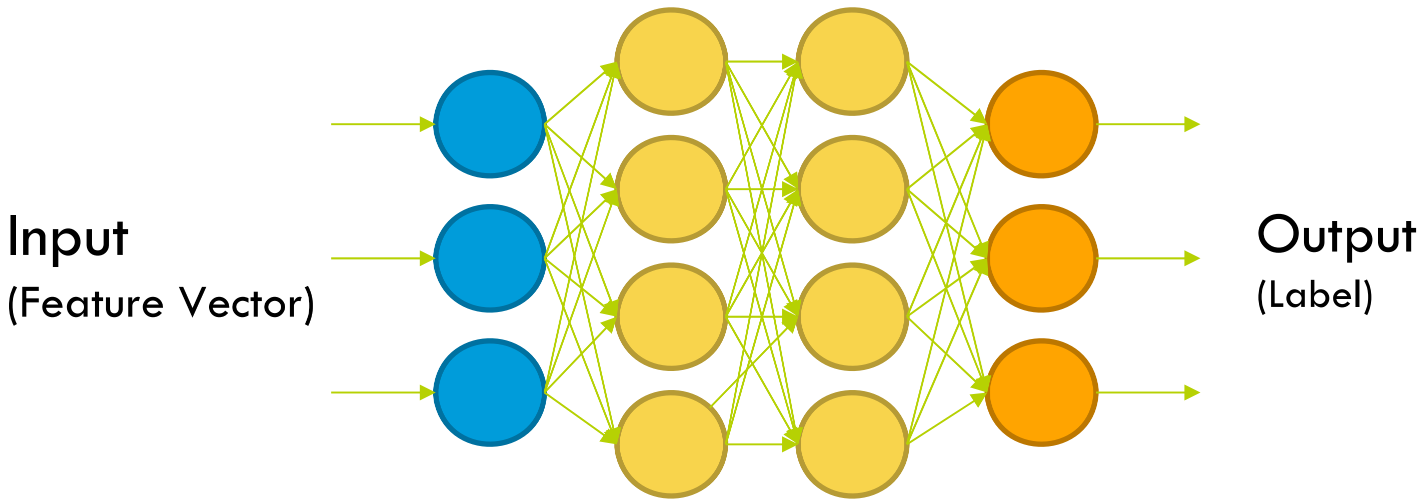
ANN

- Motivation for Neural Nets
  - Use biology as inspiration for mathematical model
  - Get signals from previous neurons
  - Generate signals (or not) according to inputs
  - Pass signals on to next neurons
  - By layering many neurons, can create complex model



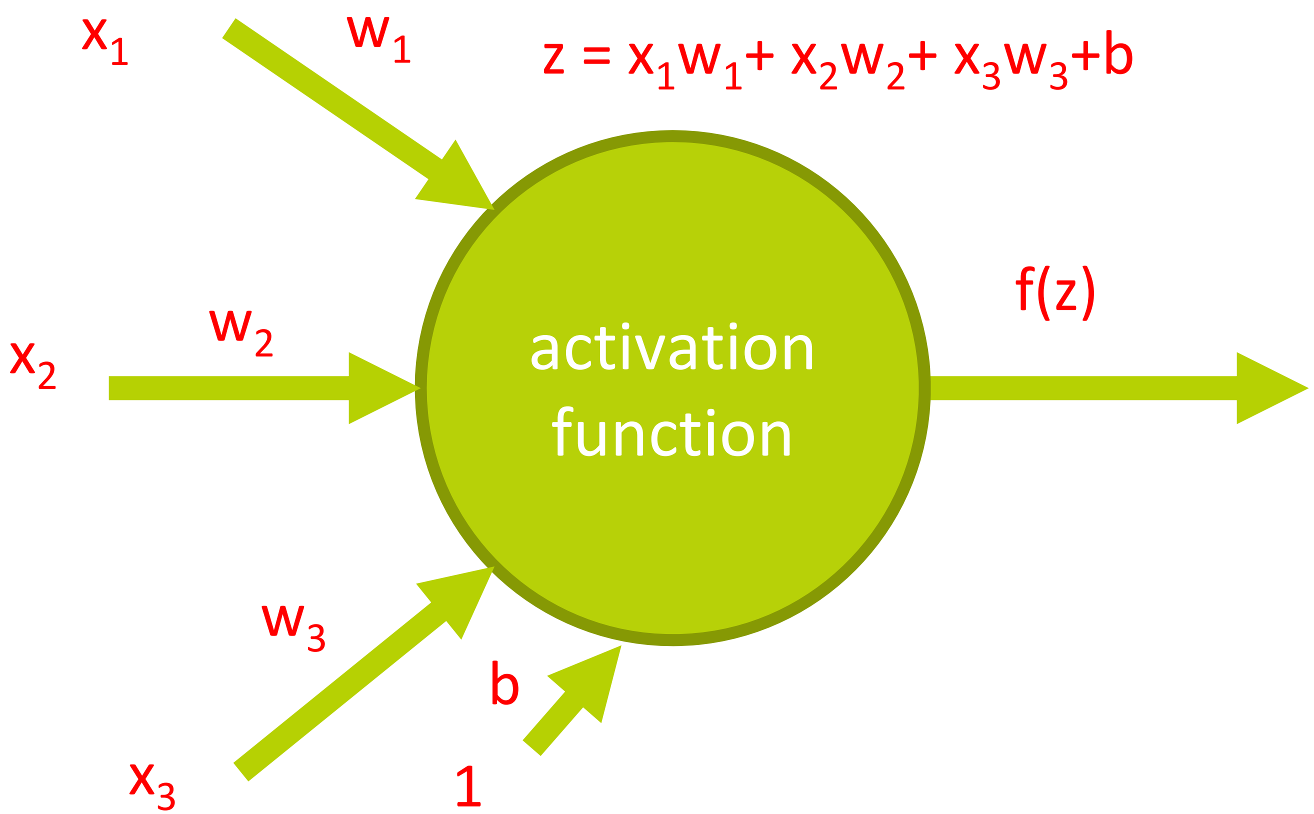
ANN

- Neural Net Structure
  - Can think of it as a complicated computation engine
  - We will ”train it” using our training data
  - Then (hopefully) it will give good answers on new data



ANN

- Basic Neuron Visualization



ANN

- Basic Neuron Visualization – In Vector Notation

z = “net input”

b = “bias term”

f = activation function

a = output to next layer

$$z = b + \sum_{i=1}^{m} x_i w_i$$

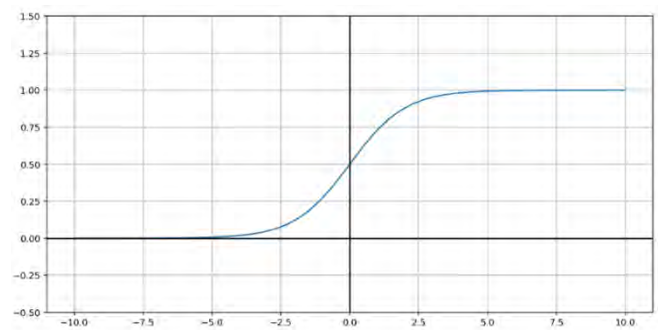
$$z = b + x^T w$$

$$a = f(z)$$

ANN

- Relation to Logistic Regression

When we choose: $f(z) = \frac{1}{1+e^{-z}}$



$$z = b + \sum_{i=1}^{m} x_i w_i = x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b$$

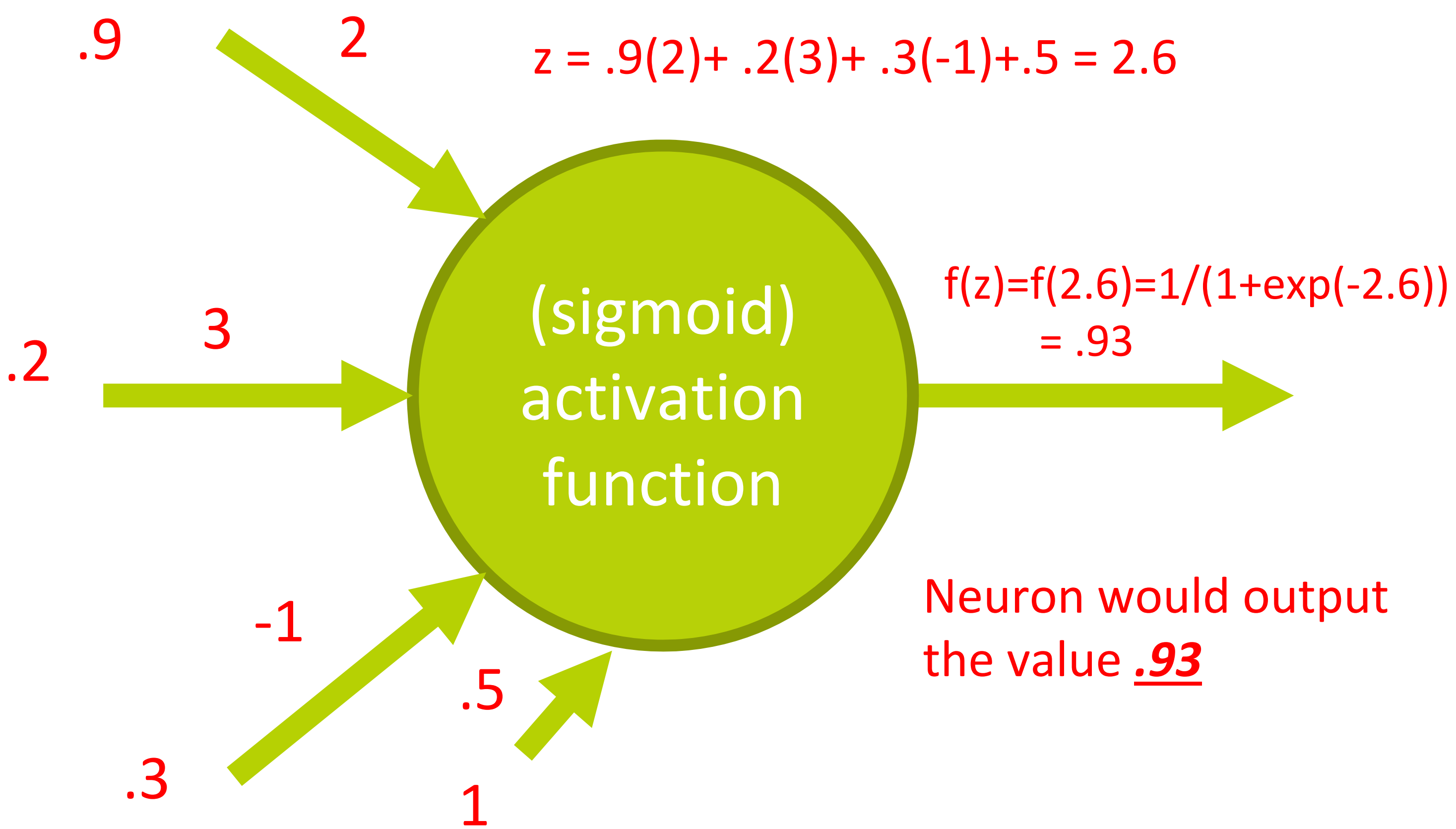
Then **a neuron is simply a ”unit” of logistic regression!**

weights ⇔ coefficients   inputs ⇔ variables

bias term ⇔ constant term

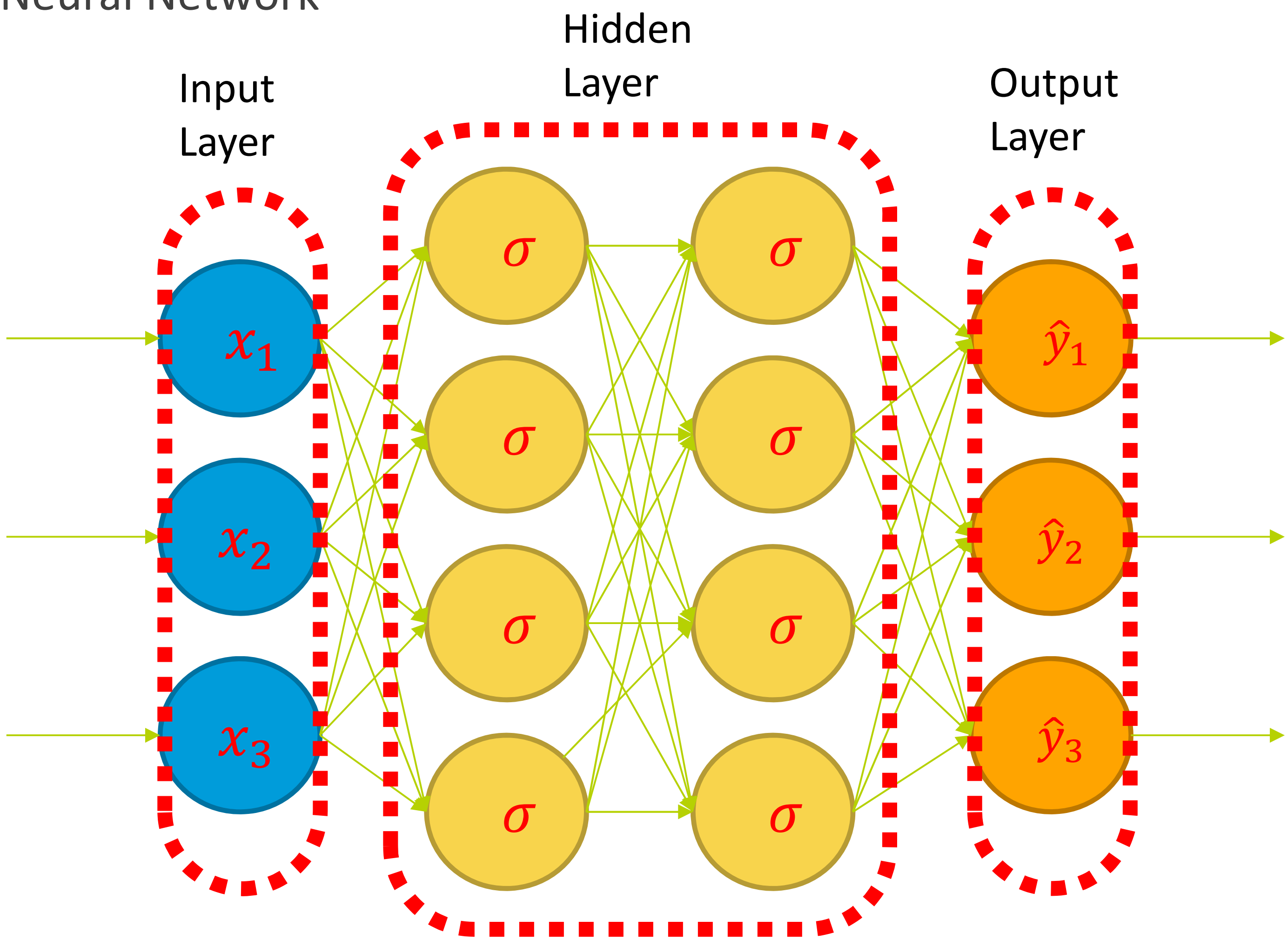
ANN

- An Example



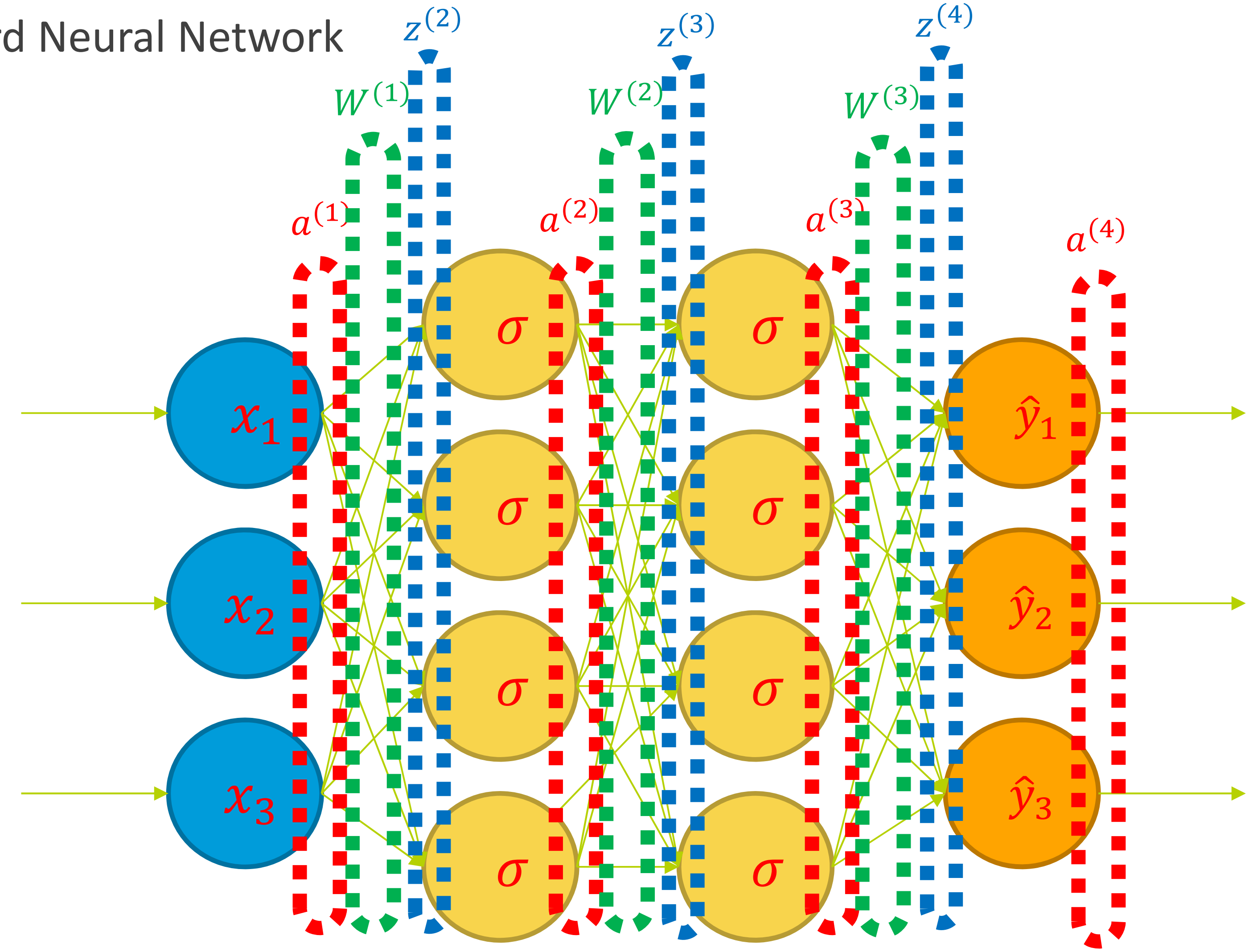
ANN

- Feedforward Neural Network
  - Layers



ANN

- Feedforward Neural Network
  - Values



ANN

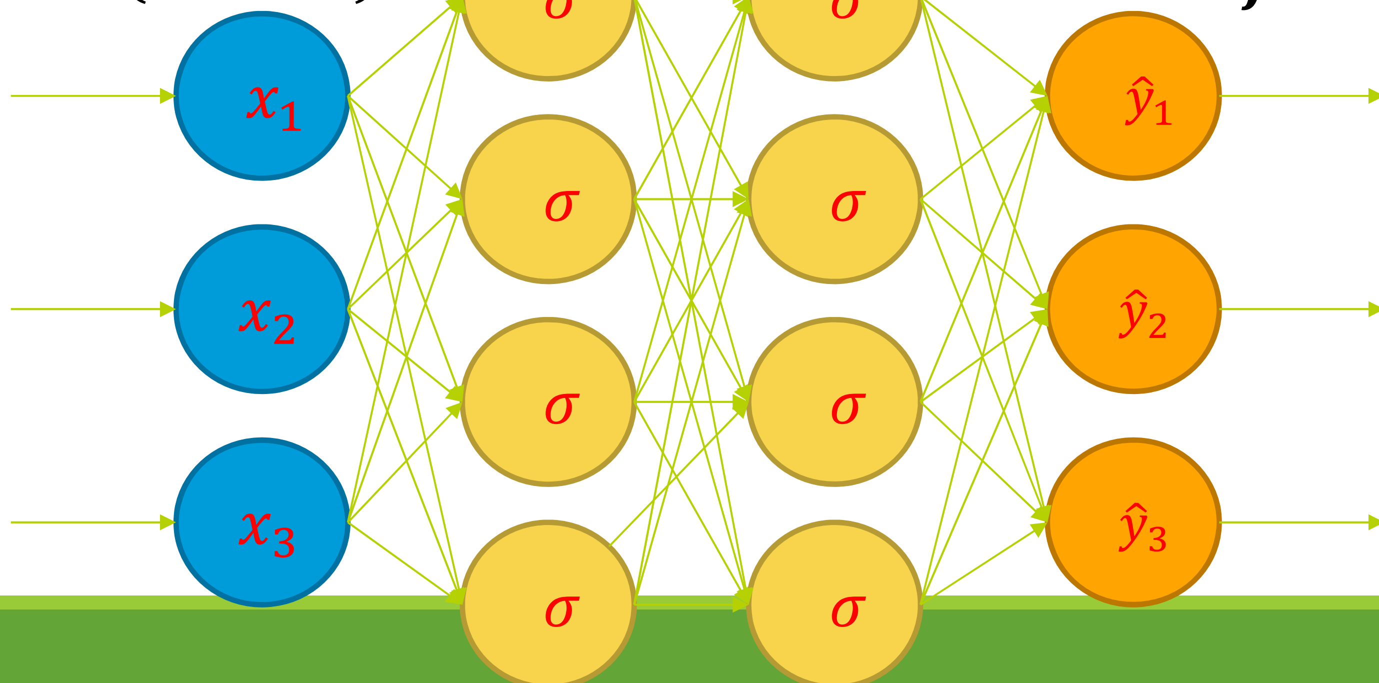
- Feedforward Neural Network
  - Calculations

$$z^{(2)} = a^{(1)}W^{(1)} \qquad z^{(3)} = a^{(2)}W^{(2)}$$

$$a^{(2)} = \sigma(z^{(2)}) \qquad a^{(3)} = \sigma(z^{(3)})$$

$$z^{(4)} = a^{(3)}W^{(3)}$$

$$(a^{(1)} = x) \qquad a^{(4)} = \boldsymbol{softmax}(z^{(4)})$$



ANN

- Training the Neural Network - Backpropagation
  - Gradient Descent Principle
  - Step 1: Make prediction
  - Step 2: Calculate Loss
  - Step 3: Calculate gradient of the loss function w.r.t. parameters
  - Step 4: Update parameters by taking a step in the opposite direction
  - Step 5: Iterate

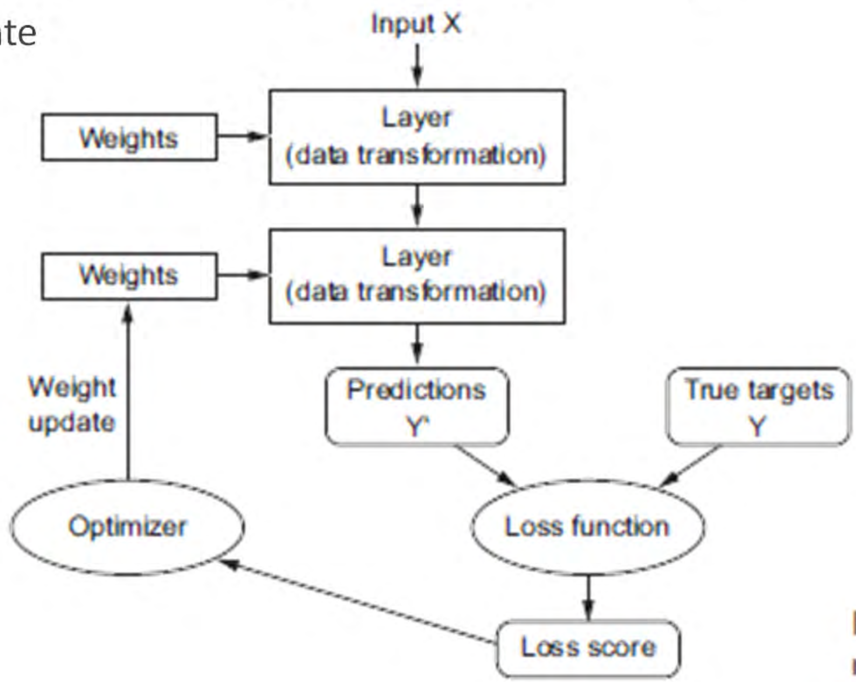
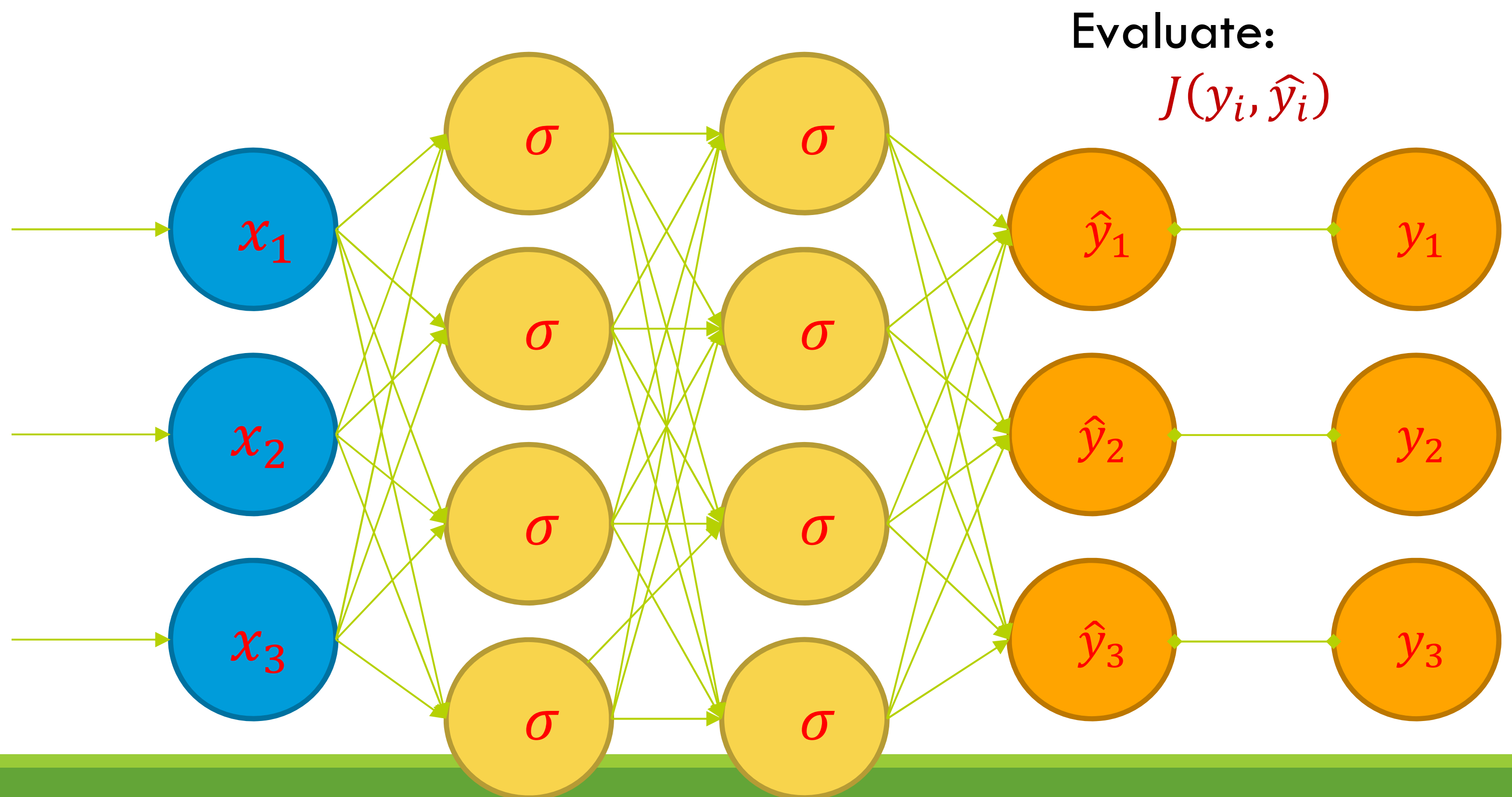


Figure 3.1 Relationship between the network, layers, loss function, and optimizer

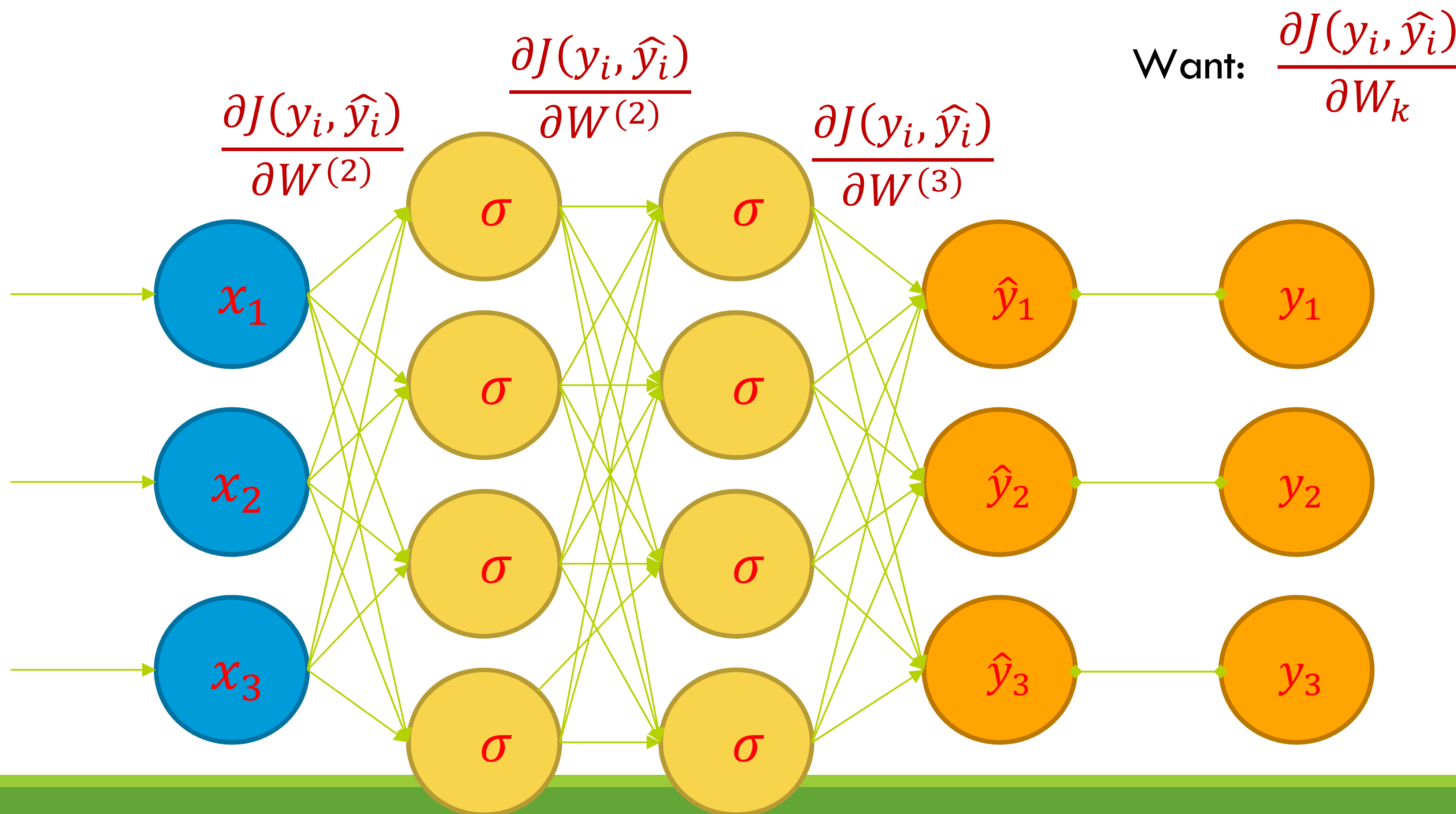
ANN

- Training the Neural Network - Backpropagation
  - Step 1: Make prediction
  - Step 2: Calculate Loss
    - Need to use an appropriate Loss function



ANN

- Training the Neural Network - Backpropagation
  - Step 3: Calculate gradient of the loss function w.r.t. parameters



ANN

- Training the Neural Network - Backpropagation
  - Step 3: Calculate gradient of the loss function w.r.t. parameters
    - Using **Chain Rule** and **Calculus**
    - This is the "**Back**" propagation
    - Though they appear complex, they are easy to compute!
    - This training approach is called "Stochastic Gradient Descent" (SGD)

$$\frac{\partial J(y,\hat{y})}{\partial W^{(3)}} = (\hat{y} - y) \cdot a^{(3)}$$

$$\frac{\partial J(y,\hat{y})}{\partial W^{(2)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot a^{(2)}$$

$$\frac{\partial J(y,\hat{y})}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

Where: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

ANN

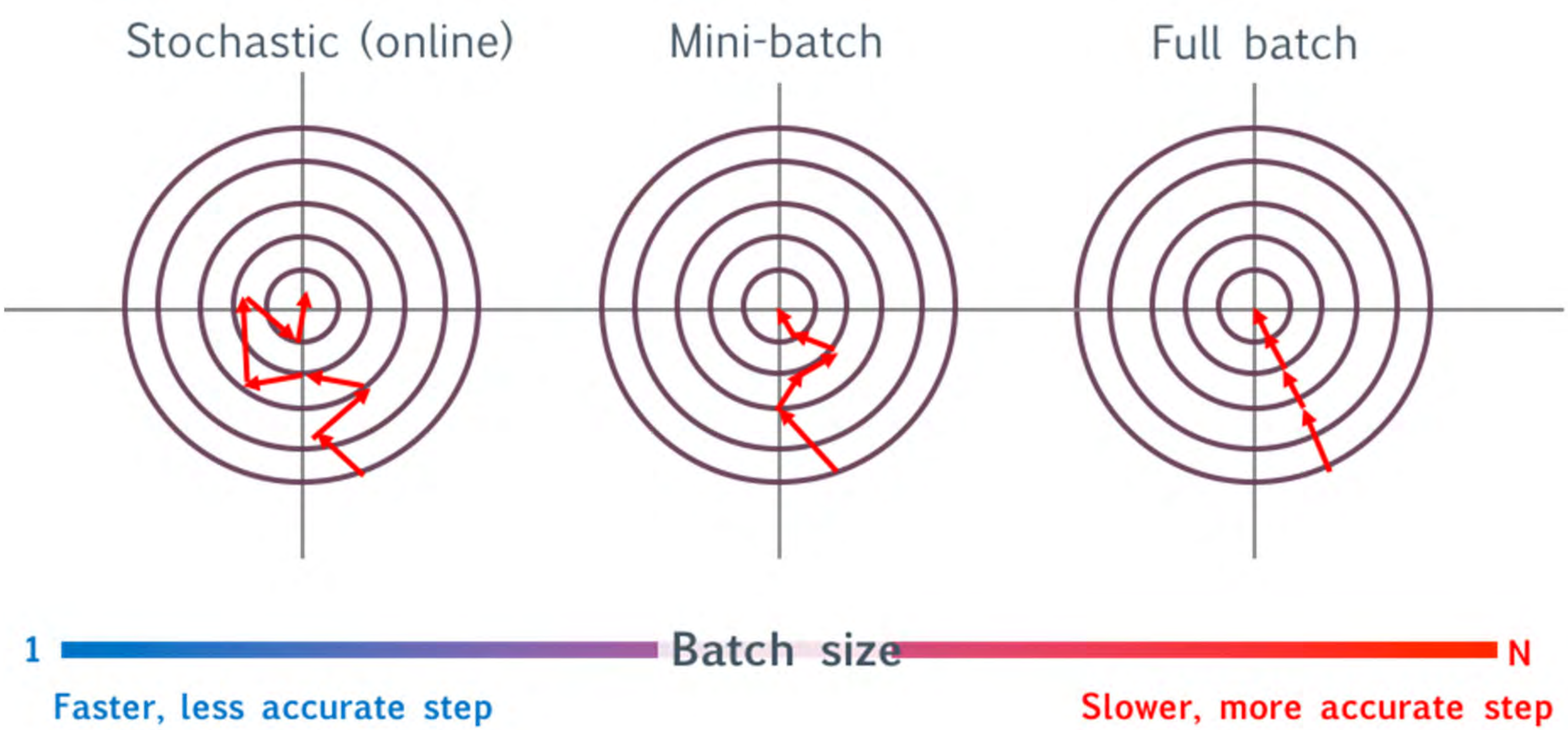
- Training the Neural Network - Backpropagation
  - Step 4: Update parameters by taking a step in the opposite direction

$$W^{(k)} = W^{(k)} - rate \times \frac{\partial J(y, \hat{y})}{\partial W^{(k)}}$$

  - Step 5: Iterate
    - Fix number of iterations
    - Until target accuracy is met

ANN

- Batch Training
  - A balance between speed and accuracy is needed

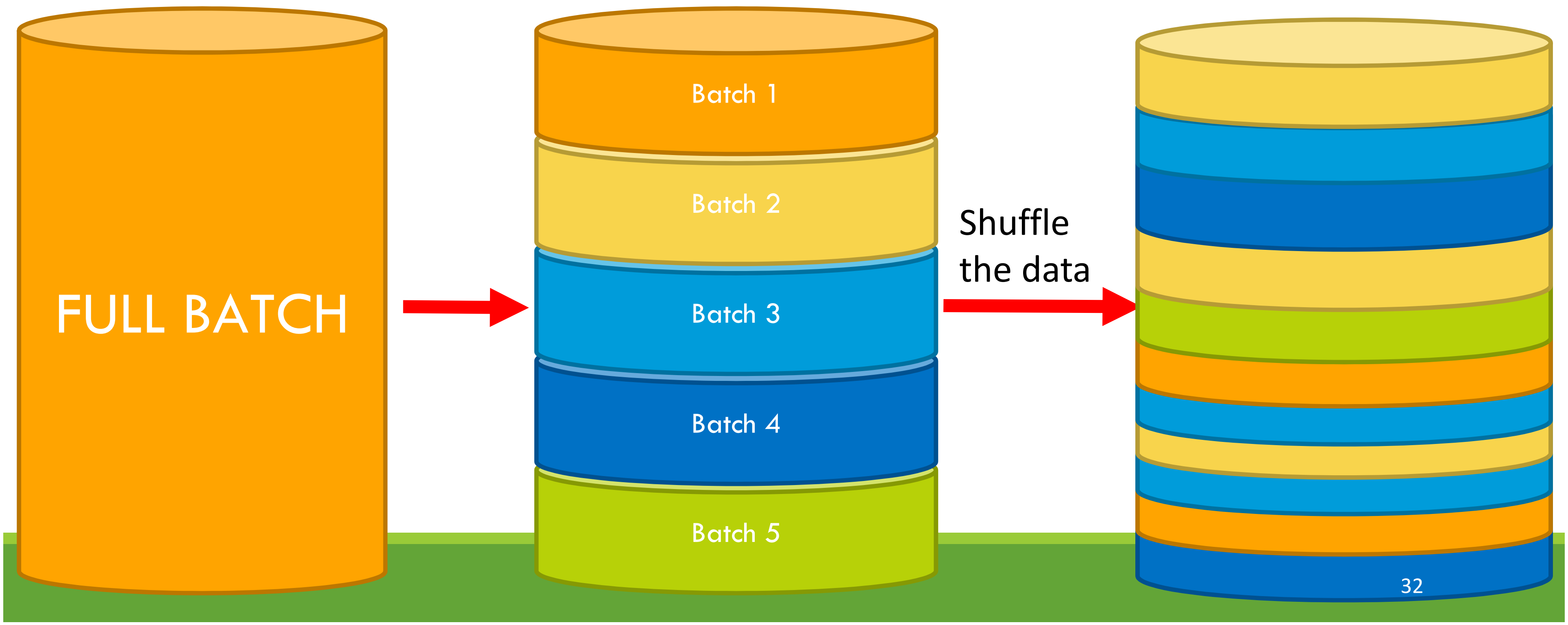


ANN

- Batch Training
  - An Epoch refers to a single pass through all of the training data.
  - In **full batch gradient descent**, there would be one step taken per epoch.
  - In **Stochastic Gradient Descent**, there would be n steps taken per epoch (n = training set size)
  - In **Minibatch** there would be (n/batch size) steps taken per epoch
  - When training, it is common to refer to the number of epochs needed for the model to be “trained”.

ANN

- Data Shuffling
  - To avoid any cyclical movement and aid convergence, it is recommended to shuffle the data after each epoch.
  - This way, the data is not seen in the same order every time, and the batches are not the exact same ones.



ANN

- Issue with ANN and Sigmoid
  - An example of gradient calculation

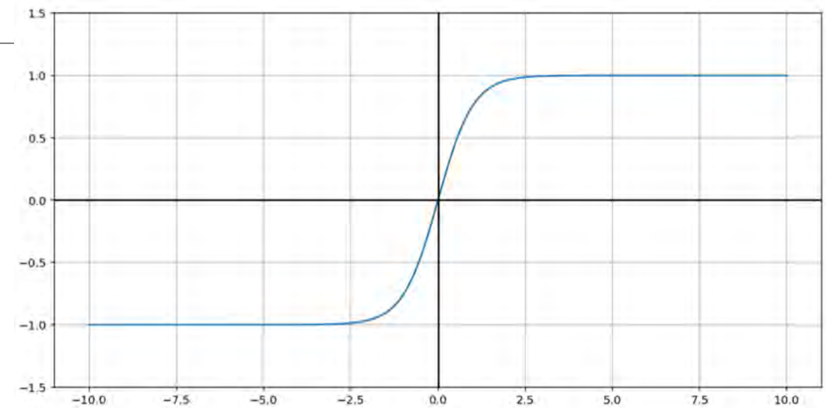
$$\frac{\partial J}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

  - It is known that $\sigma'(z) = \sigma(z)(1-\sigma(z)) \leq .25$
  - **Therefore, with many layers, multiple multiplication of $\sigma'(z)$ results in a very small value, close to 0**
  - This is known as the "**Vanishing gradient**" problem
  - **To overcome this problem, other activations functions are used.** However, this complicates the calculations.

ANN

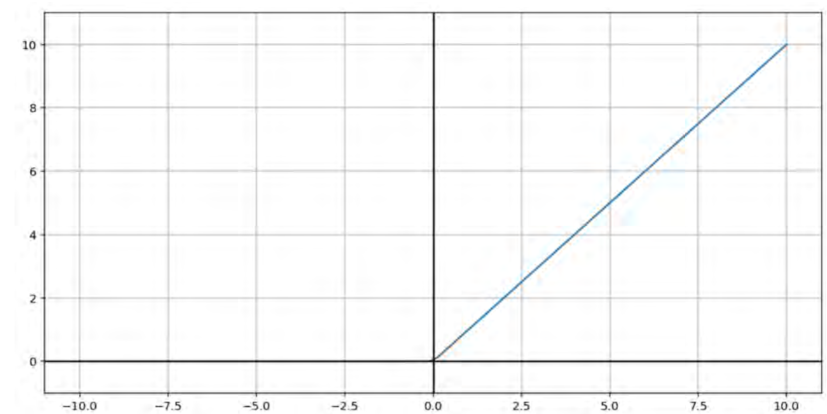
- Other Activation Functions
  - Hyperbolic tangent function

$$tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{2x} - 1}{e^{2x} + 1}$$



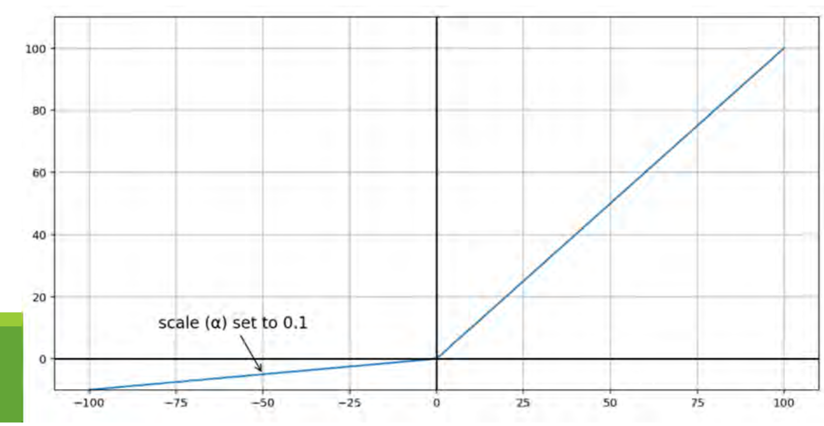
  - Rectified Linear Unit (ReLU)

$$ReLU(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$



  - "Leaky" Rectified Linear Unit (LReLU)

$$LReLU(z) = \begin{cases} \alpha z, & z < 0 \\ z, & z \geq 0 \end{cases}$$



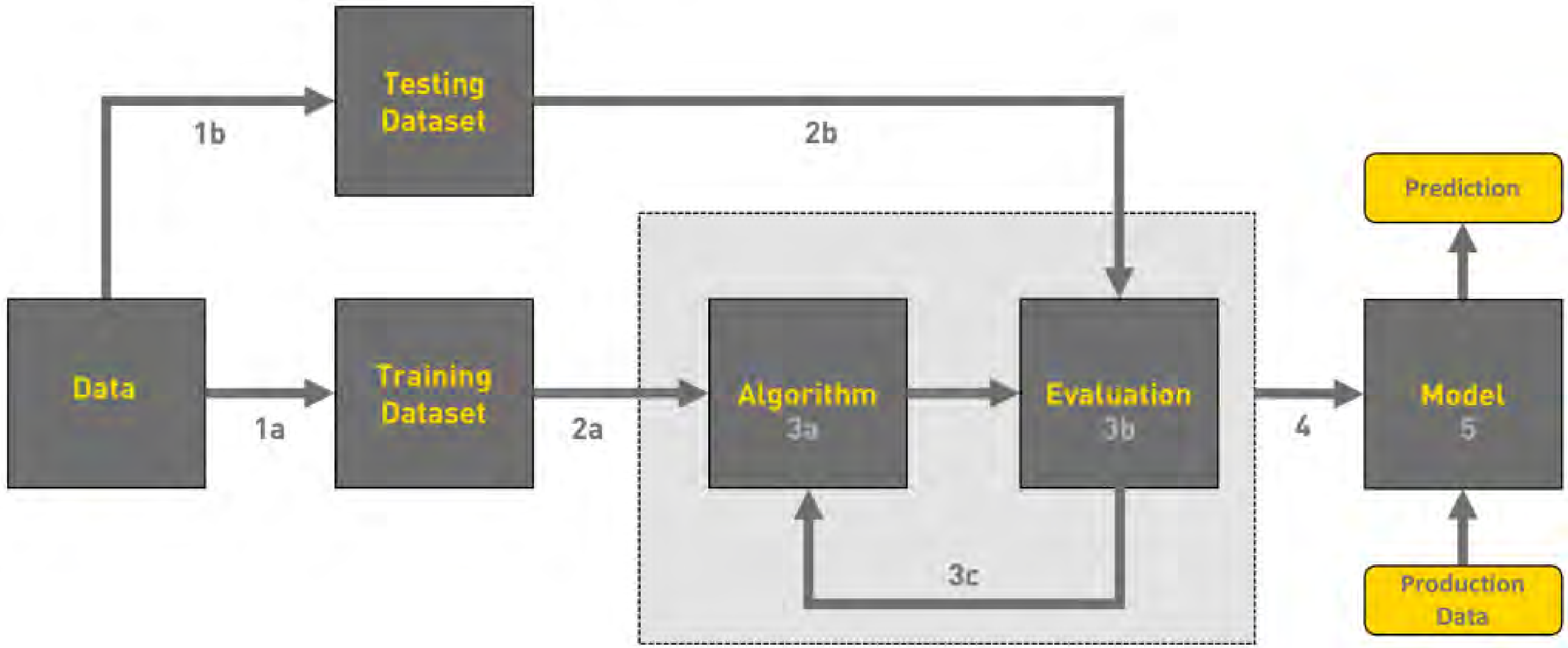
Google Colab

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

https://www.youtube.com/watch?v=inN8seMm7UI

Deep Learning workflow



https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

Evaluation metrics - accuracy

- Used when we want a model to minimize the number of errors
- Classification Accuracy

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

- Classification Error

$$Error = \frac{Incorrect\ Predictions}{Total\ Predictions}$$

Evaluation metrics - loss

- Log Loss (cross-entropy)

$$LogLoss = -((1 - y) \times \log(1 - yhat) + y \times \log(yhat)$$

- The score can be generalized to multiple classes by simply adding the terms

$$LogLoss = -\sum^{cEC} y_c \times \log(yhatc)$$

- This generalization is also known as **cross-entropy**
- summarizes the **average difference between two probability distributions**.
- **A perfect classifier has a log loss of 0.0, with worse values being positive up to infinity**

sklearn.metrics.log loss API.
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

Evaluation metrics – Regression

- Used when we want evaluate a regression model
- Mean Square Error(MSE)/Root Mean Square Error(RMSE)
  - a relative measure of how well the model fits dependent variables, Mean Square Error is an absolute measure of the goodness for the fit.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- Mean Absolute Error(MAE)
  - MAE is taking the sum of absolute value of error.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

A smaller value is better

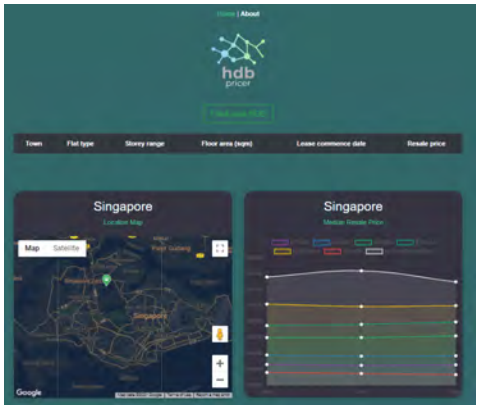
Confusion matrix

- a table to **describe the performance of a classification model** on a set of test data for which the true values are known.
- relatively simple to understand, but the related terminology can be confusing.

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

Activity 1 – Predicting HDB Resale price

Tryout the app at:
http://www.hdbpricer.com/



The Resale Flat Prices dataset can be downloaded at https://data.gov.sg/dataset/resale-flat-prices. It contains the following data, month of the transaction, the town, flat type, block, street name, story range, floor area, flat model, lease commence date, remaining lease and Resale price.

What we want to do is to train a neural network model that is able to predict the resale price of a particular HDB flat, given some information like town, lease commence date and floor area for example.

month	town	flat_type	block	street_name	storey_range	floor_area_sqm	flat_model	lease_commence_date	remaining_lease	resale_price
2017-01	ANG MO KIO	2 ROOM	406	ANG MO KIO AVE 10	10 TO 12	44.0	Improved	1979	61 years 04 months	232000.0
2017-01	ANG MO KIO	3 ROOM	108	ANG MO KIO AVE 4	01 TO 03	67.0	New Generation	1978	60 years 07 months	250000.0
2017-01	ANG MO KIO	3 ROOM	602	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	262000.0
2017-01	ANG MO KIO	3 ROOM	465	ANG MO KIO AVE 10	04 TO 06	68.0	New Generation	1980	62 years 01 month	265000.0
2017-01	ANG MO KIO	3 ROOM	601	ANG MO KIO AVE 5	01 TO 03	67.0	New Generation	1980	62 years 05 months	265000.0

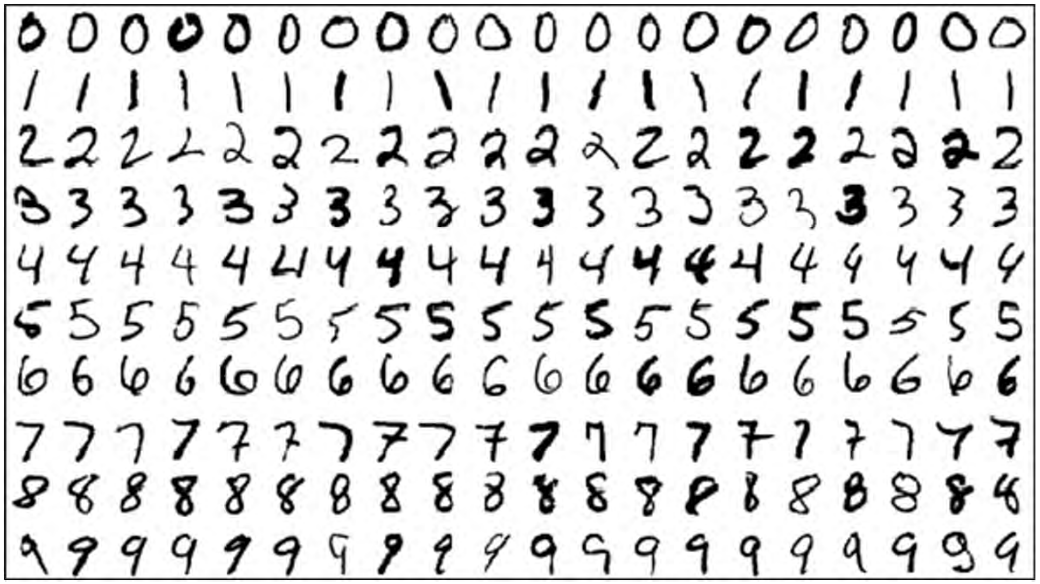
Watch and listen to the instructor's demonstration

15 mins

Activity 2 - Classifying Grayscale Handwritten Digits

MNIST : dataset of 70,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

Colab.research.google.com



Exercises:

- Add a hidden fully connected layer with 128 neurons and relu activation function.
- Compare the performance

Step 1:
Watch and listen to the instructor’s demonstration

15 mins

Step 2:
Work through the activities

Individual Activity

20 mins

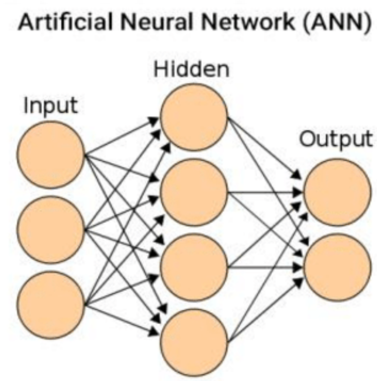
Activity 3 - Classifying Fashion

An MNIST-like dataset of 70,000 28x28 labeled fashion images

Labels

Each training and test example is assigned to one of the following labels:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot



Exercises:
- Add another hidden fully connected layer after the 1st hidden layer.
- Compare the performance

Step 1:
Watch and listen to the instructor's demonstration

10 mins

Step 2:
Work through the activities

Individual Activity

20 mins

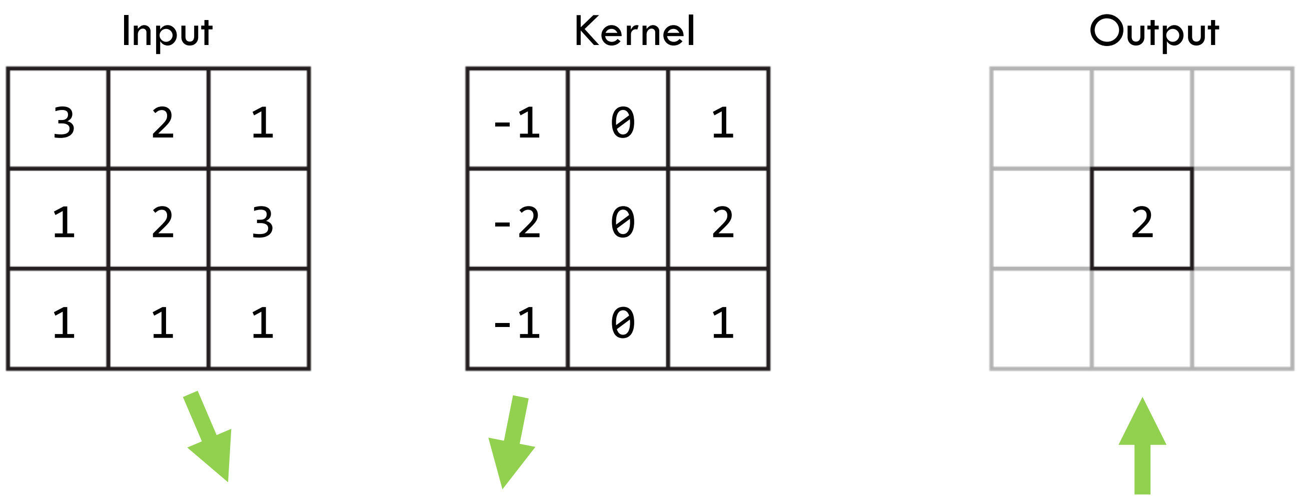
Convolutional Neural Network

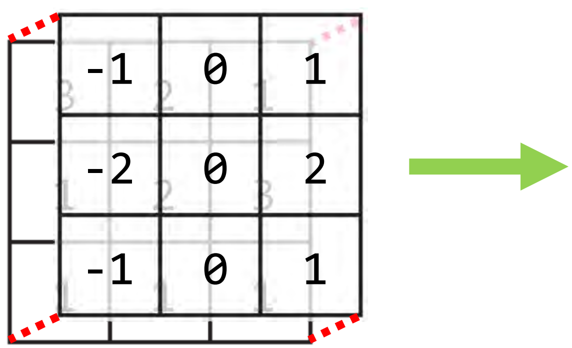
Image Convolution

- Basic concept of Convolution – Kernels
  - A *kernel* is a grid of weights “overlaid” on image, centred on one pixel
  - Each weight multiplied with pixel underneath it
  - Output over the centred pixel is $\sum_{p=1}^{P} W_p \cdot pixel_p$
  - Used for traditional image processing techniques:
    - Blur
    - Sharpen
    - Edge detection
    - Emboss

Image Convolution

- Basic concept of Convolution – **Kernels**
  - 3x3 Example





$$= (3 \cdot -1) + (2 \cdot 0) + (1 \cdot 1)$$
$$+(1 \cdot -2) + (2 \cdot 0) + (3 \cdot 2)$$
$$+(1 \cdot -1) + (1 \cdot 0) + (1 \cdot 1)$$

$$= -3 + 1 - 2 + 6 - 1 + 1 = 2$$

Image Convolution

- Basic concept of Convolution – **Kernels**
  - "Local feature detectors"

Vertical Line Detector

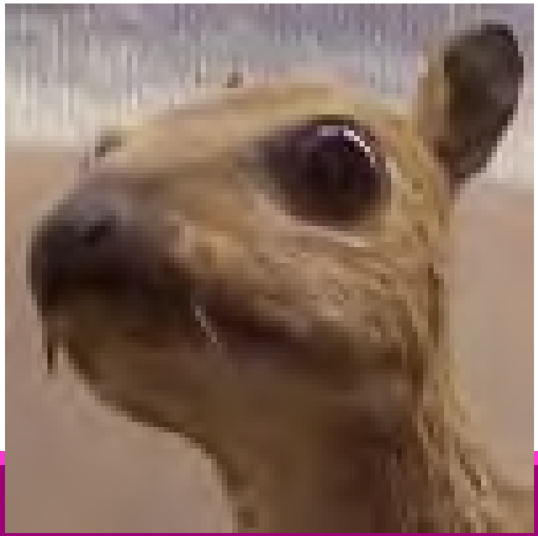
-1	1	-1
-1	1	-1
-1	1	-1

Horizontal Line Detector

-1	-1	-1
1	1	1
-1	-1	-1

Corner Detector

-1	-1	-1
-1	1	1
-1	1	1



*

Edge Detector

-1	-1	-1
-1	8	-1
-1	-1	-1

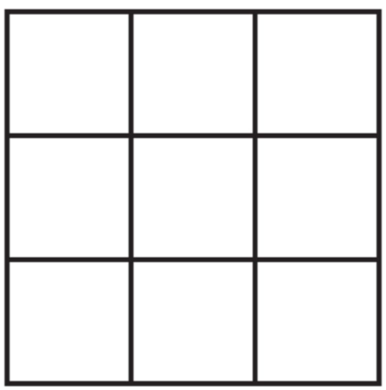
=

https://en.wikipedia.org/wiki/Kernel_(image_processing)

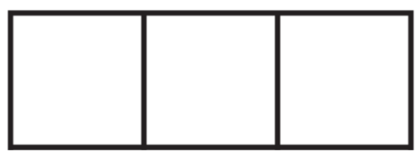
Image Convolution

- Basic concept of Convolution – Settings
  - ***Grid Size*** (Height and Width):
    - The number of pixels a kernel “sees” at once
    - Typically use odd numbers so that there is a “center” pixel
    - Kernel does not need to be square

Height: 3, Width: 3



Height: 1, Width: 3



Height: 3, Width: 1

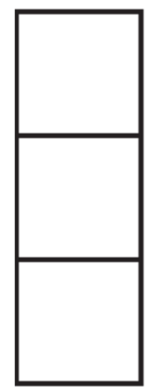


Image Convolution

- Basic concept of Convolution – Settings
  - ***Padding***
    - Using Kernels directly, there will be an "edge effect"
    - Pixels near the edge will not be used as "center pixels" since there are not enough surrounding pixels
    - Padding adds extra pixels around the frame
    - So every pixel of the original image will be a center pixel as the kernel moves across the image
    - Added pixels are typically of value zero (zero-padding)

Image Convolution

- Basic concept of Convolution – Settings
  - *Padding*

Without Padding

1	2	0	3	1
1	0	0	2	2
2	1	2	1	1
0	0	1	0	0
1	2	1	1	1

input

-1	1	2
1	1	0
-1	-2	0

kernel

-2		

output

Image Convolution

- Basic concept of Convolution – Settings
  - *Padding*

With Padding

0	0	0	0	0	0	0
0	1	2	0	3	1	0
0	1	0	0	2	2	0
0	2	1	2	1	1	0
0	0	0	1	0	0	0
0	1	2	1	1	1	0
0	0	0	0	0	0	0

input

-1	1	2
1	1	0
-1	-2	0

kernel

-1				

output

Image Convolution

- Basic concept of Convolution – Settings
  - ***Stride***
    - The ”step size” as the kernel moves across the image
    - Can be different for vertical and horizontal steps (but usually is the same value)
    - When stride is greater than 1, it scales down the output dimension

Image Convolution

- Basic concept of Convolution – Settings
  - ***Stride***

Stride 2 Example – No Padding

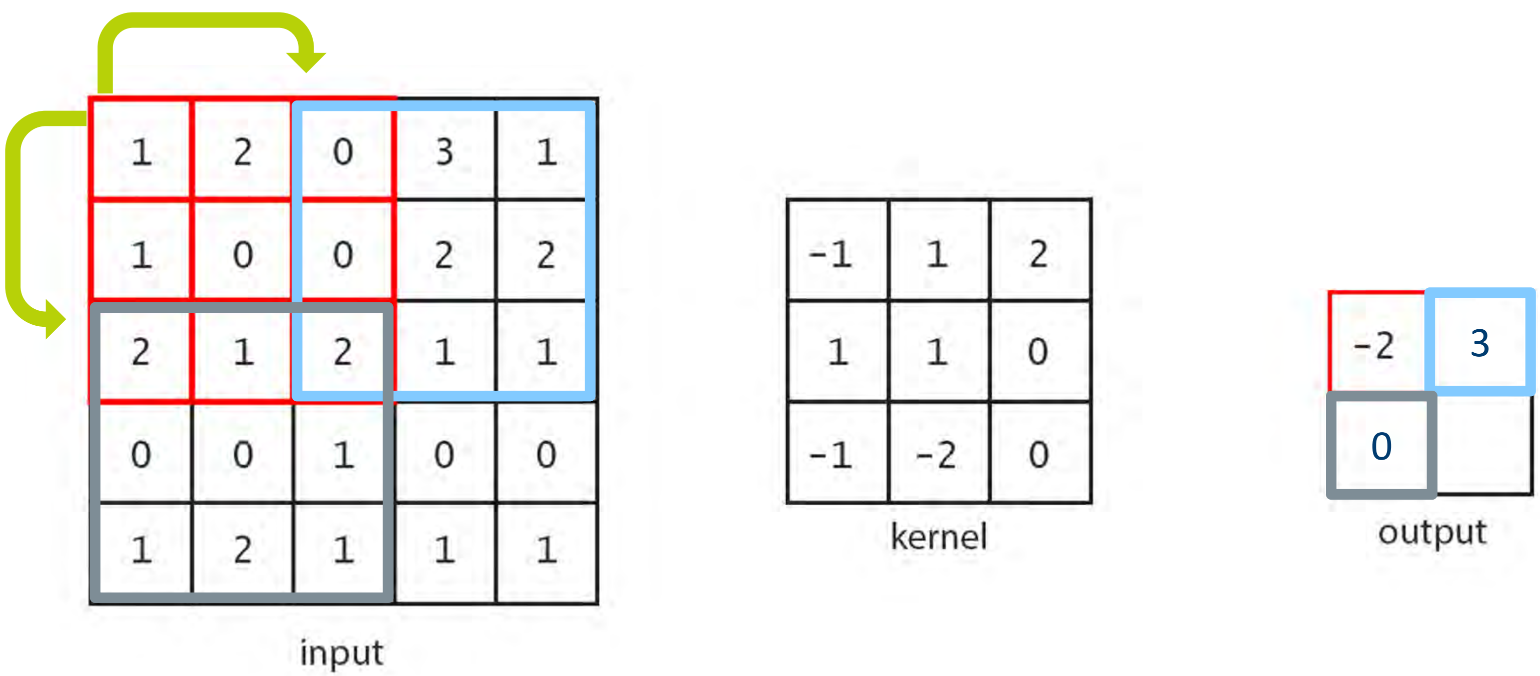


Image Convolution

- Basic concept of Convolution – Settings
  - ***Output***
    - Same calculation for height

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1$$

$where,$

W – width of image

K – kernel size

P – padding size

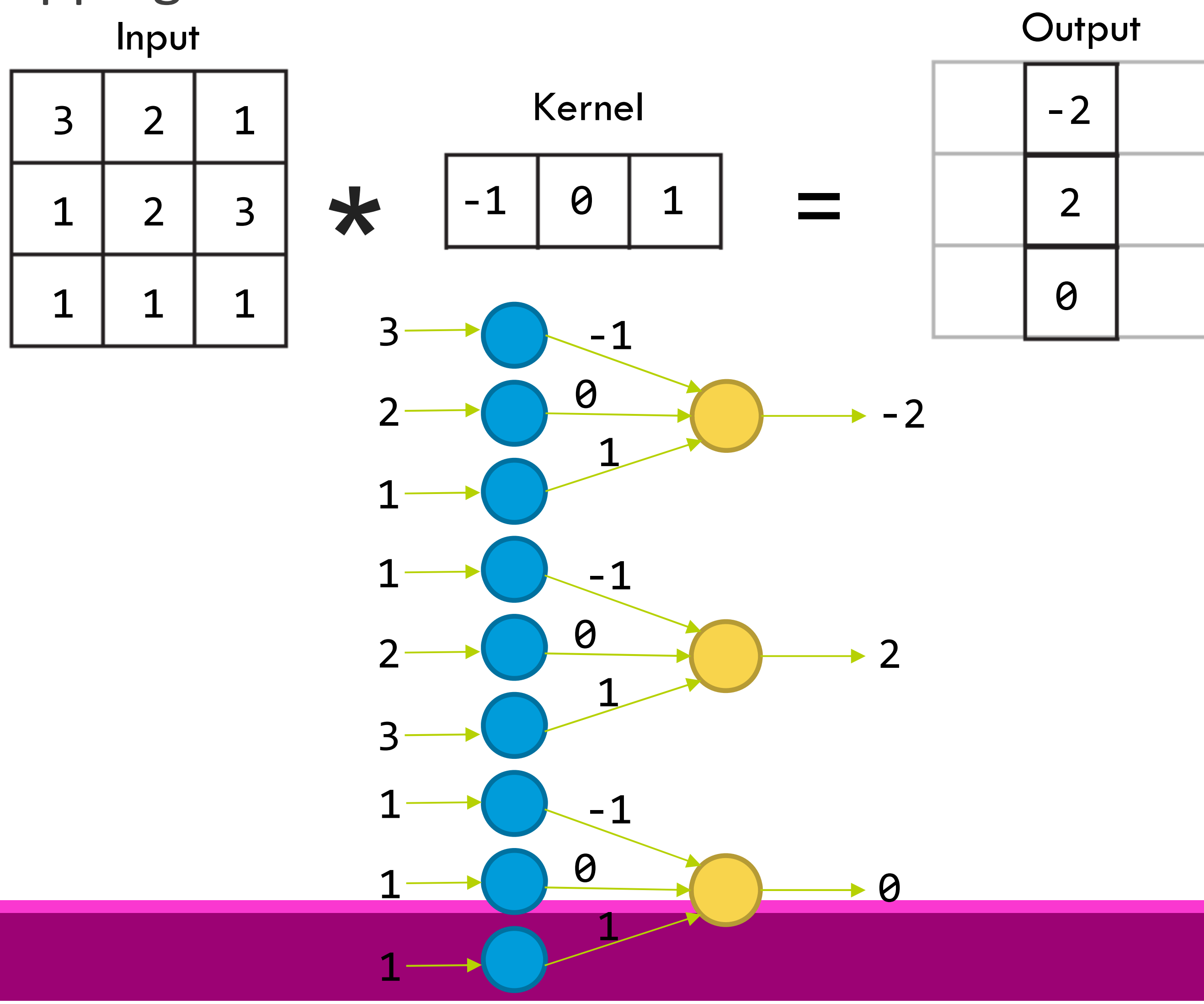
S – stride size

Image Convolution

- Basic concept of Convolution – Settings
  - ***Depth***
    - In images, we often have multiple numbers associated with each pixel location.
    - These numbers are referred to as "channels"
      - RGB image – 3 channels
      - CMYK – 4 channels
    - The number of channels is referred to as the "depth"
    - So the kernel itself will have a "depth" the same size as the number of input channels
    - The output from the layer will also have a depth

Convolution Layer

- Mapping Convolution to ANN



Convolution Layer

- Basic idea of Convolution Layer
  - The network is not fully connected
  - Same set of kernels (weights) across entire image
    - Reduces number of parameters and variance
  - There will be several ‘kernels’ detecting different aspect of the image
  - The output of the different kernels are merged or combined together
  - The network will learn a set of kernels that are best to analyze the images
- Convolution is able to analyze a subset of the data and produce a value
  - Useful if there are relationships between each node of the input layer. E.g Images.

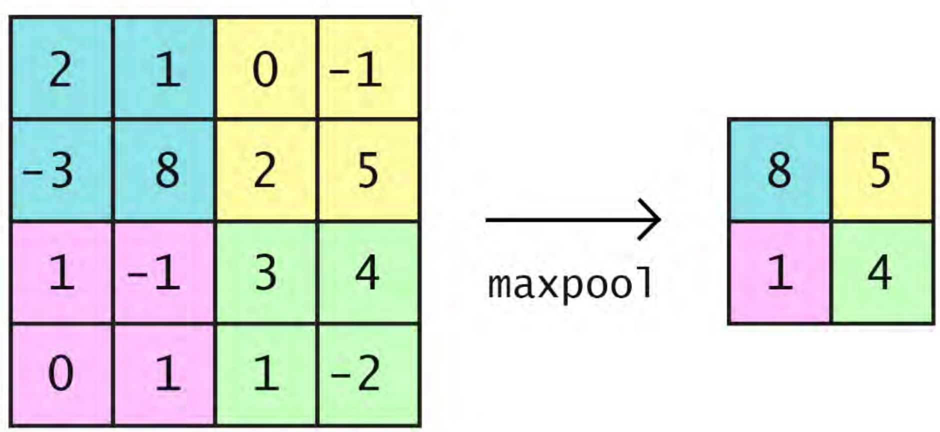
Other important layer: Pooling Layer

- **Basic concept of Pooling**
  - Reduce the image size by mapping a patch of pixels to a single value.
  - Shrinks the dimensions of the image.
  - Does not have parameters, though there are different types of pooling operations.

```
network.add(layers.MaxPooling2D(pool_size=(2, 2)))
```

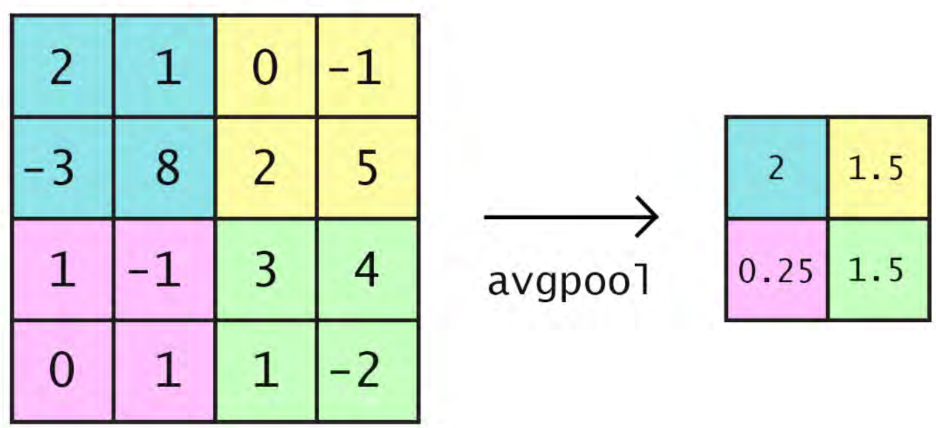
Pooling Layer

- **Max-pool**
  - For each distinct patch, represent it by the maximum
  - 2x2 maxpool shown below



Pooling Layer

- **Average-pool**
  - For each distinct patch, represent it by the average
  - 2x2 avgpool shown below.



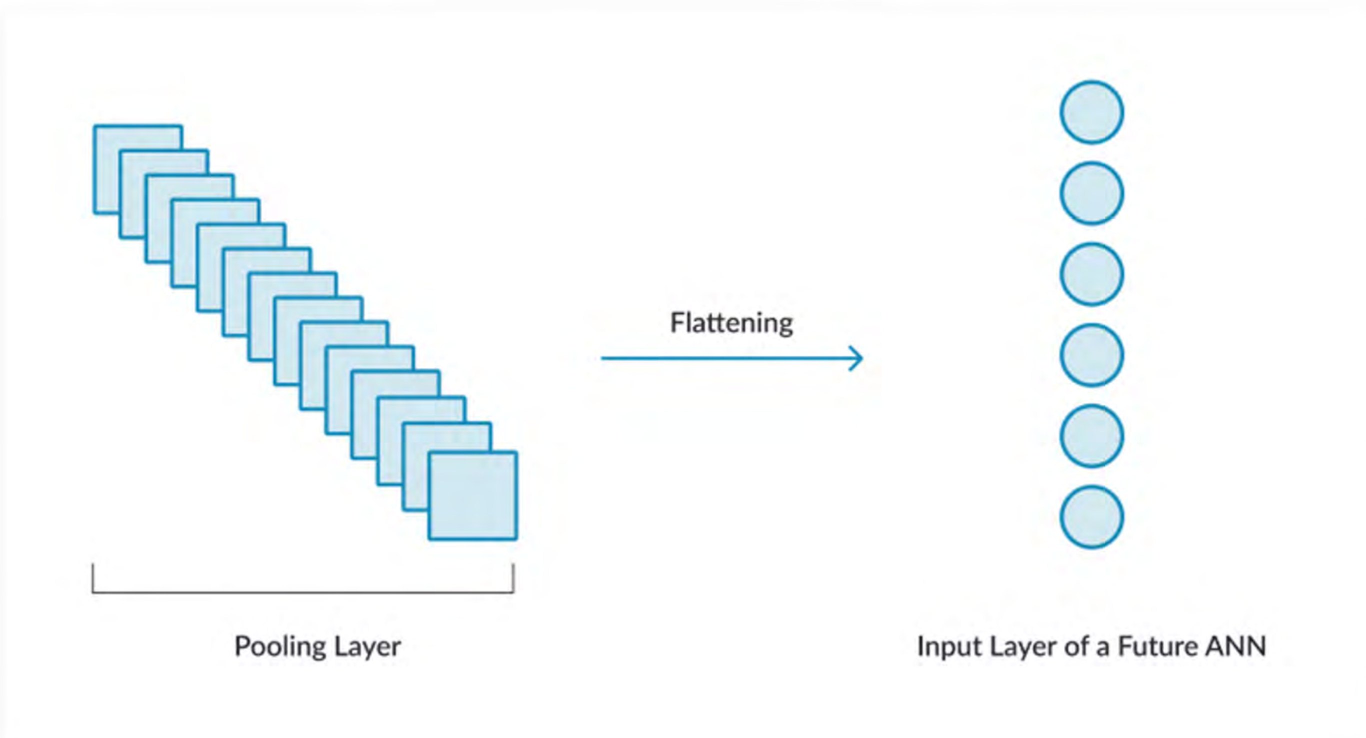
Dropout Layer

- **randomly removes one or more nodes from a network**
- **For each node removed, dropout removes the node's incoming and outgoing connections and their weights**
- **To prevent overfitting of the model**
- **Dropout is ONLY performed during training**

```
network.add(layers.Dropout(0.25))
```

Flatten layer

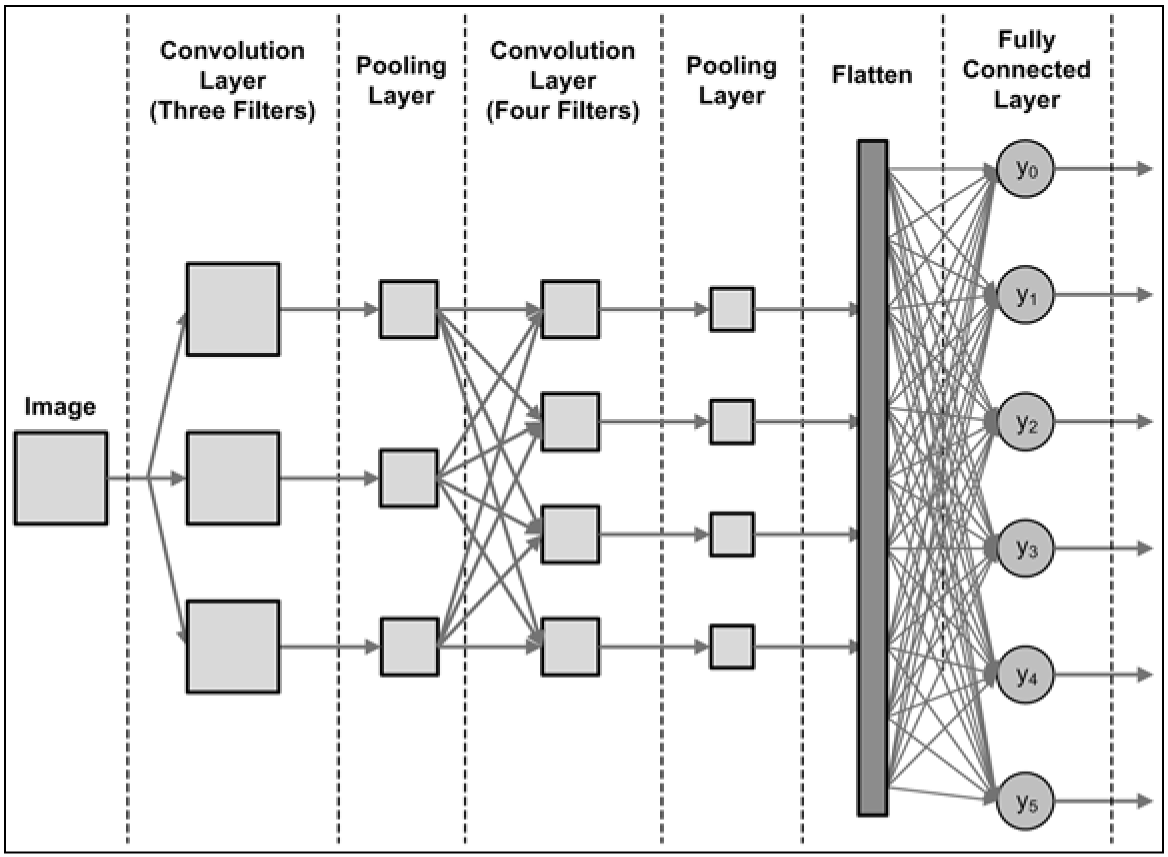
- transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.



https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/

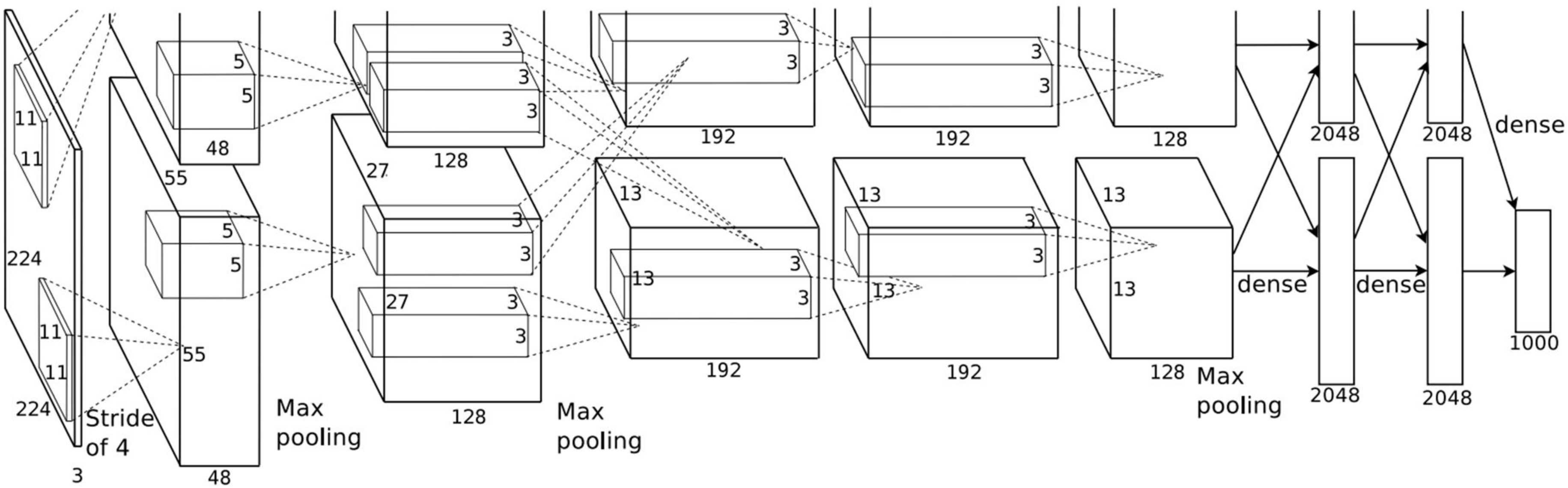
Convolution Neural Network

- By combining the convolution, pooling and dropout layers, they formed a Convolution Neural Network.



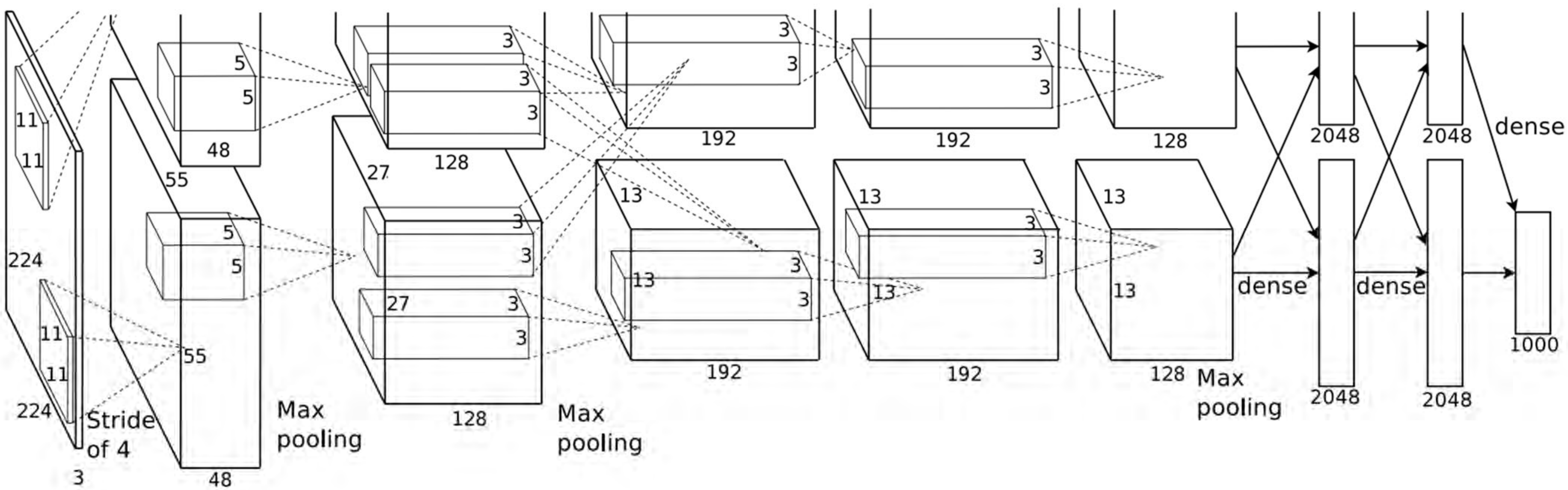
Convolutional Neural Network

- Examples of CNN – AlexNet
  - Created in 2012 for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
    - Task: predict the correct label from among 1000 classes
    - Dataset: around 1.2 million images
    - Top 5 error rate of 15.4%
    - Next best: 26.2%



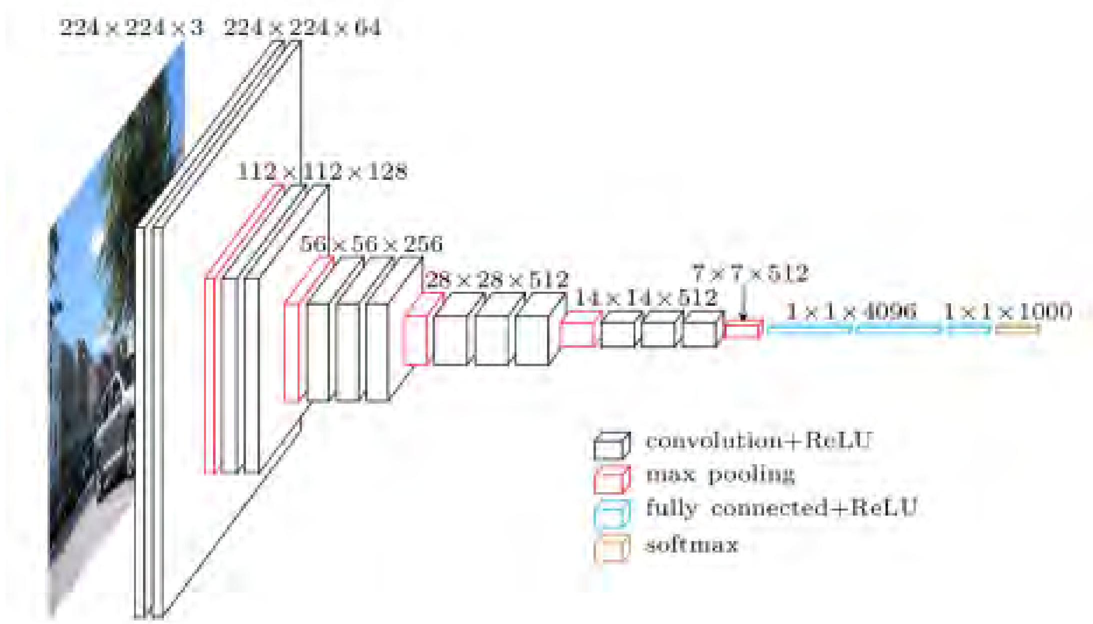
Convolutional Neural Network

- Examples of CNN – AlexNet
  - Basic Template:
    - Convolutions with ReLUs
    - Sometimes add maxpool after convolutional layer
    - Fully connected layers at the end before a softmax classifier



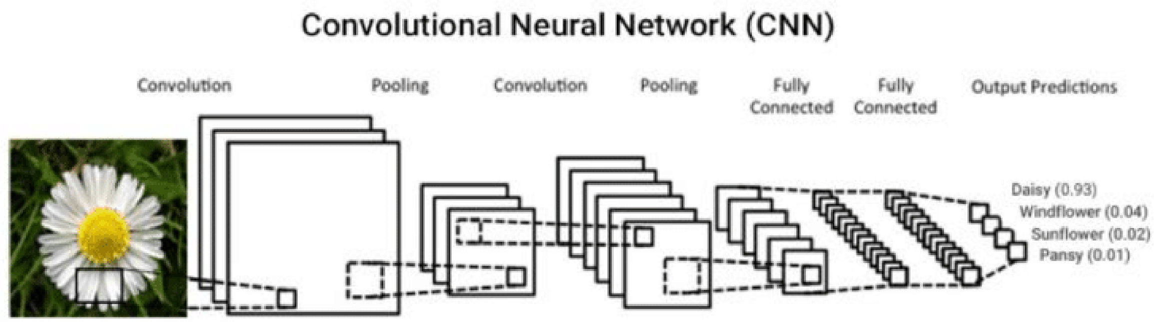
Convolutional Neural Network

- Examples of CNN – VGG16
  - Simplify Network Structure
  - Avoid Manual Choices of Convolution Size
  - Very Deep Network with 3x3 Convolutions
  - These “effectively” give rise to larger convolutions



Activity 4 - Classifying Fashion with CNN

An MNIST-like dataset of 70,000 28x28 labeled fashion images



Labels

Each training and test example is assigned to one of the following labels:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

Exercises:
- Add more convolution layers to improve the accuracy?
- How about adding more filters, changing kernel size, epochs, and other setting?

Step 1:
Watch and listen to the instructor's demonstration

10 mins

Step 2:
Work through the activities

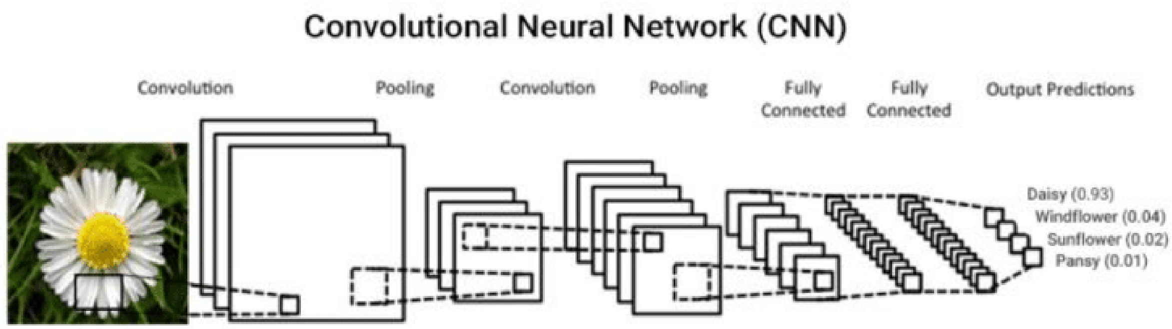
Individual Activity

30 mins

67

Activity 5 - Classifying Flowers with CNN

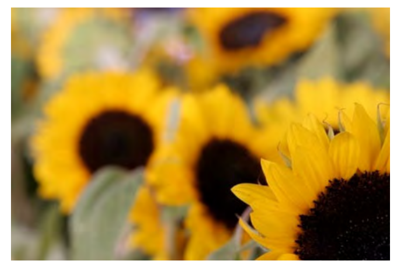
5 Flowers species were scrapped from the internet. 100 images for each species.




Daisy


Dandelion


Rose


Sunflower


Tulip

Step 1:
Watch and listen to the instructor's demonstration


10 mins

Step 2:
Work through the activities

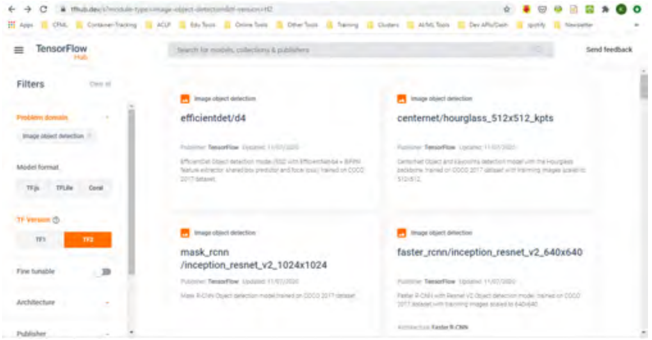
Individual Activity


30 mins

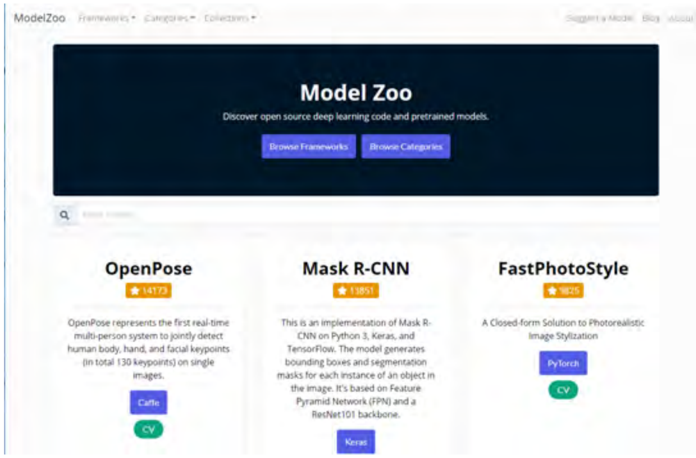
68

Pre-trained CNN models

- Models for image classification with weights trained on ImageNet:
  - Xception
  - VGG16
  - VGG19
  - ResNet, ResNetV2
  - InceptionV3
  - InceptionResNetV2
  - MobileNet
  - MobileNetV2
  - DenseNet
  - NASNet
  - EfficentNet



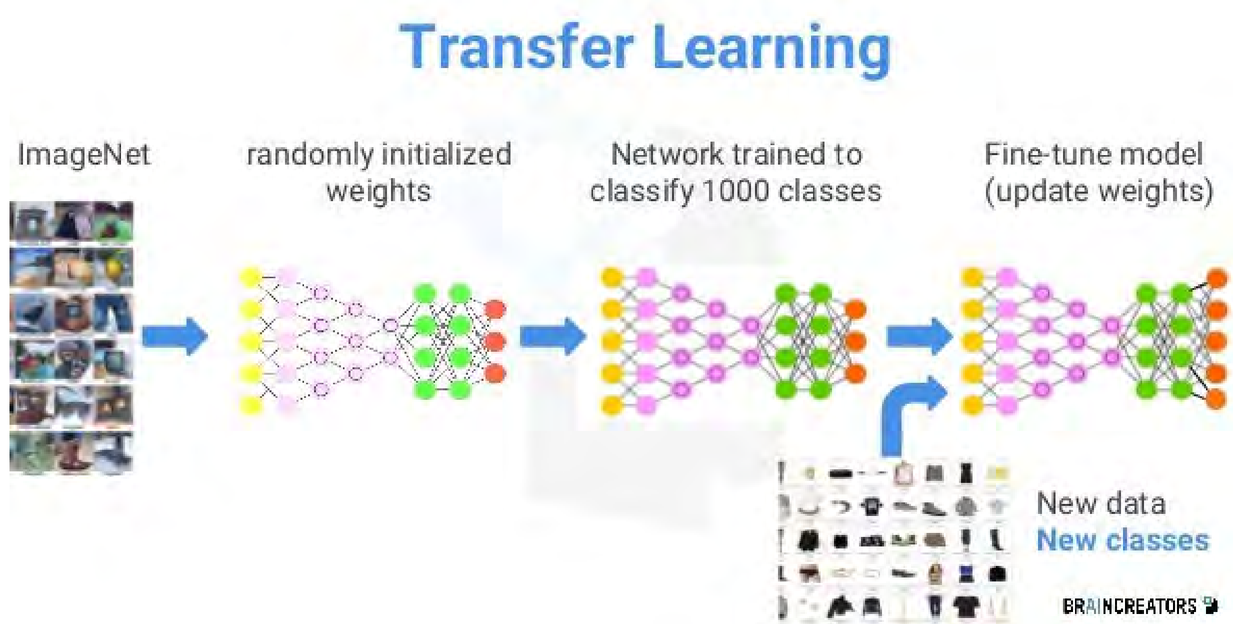
TensorFlow Hub (https://tfhub.dev)



Model Zoo (https://modelzoo.co/)

Transfer Learning

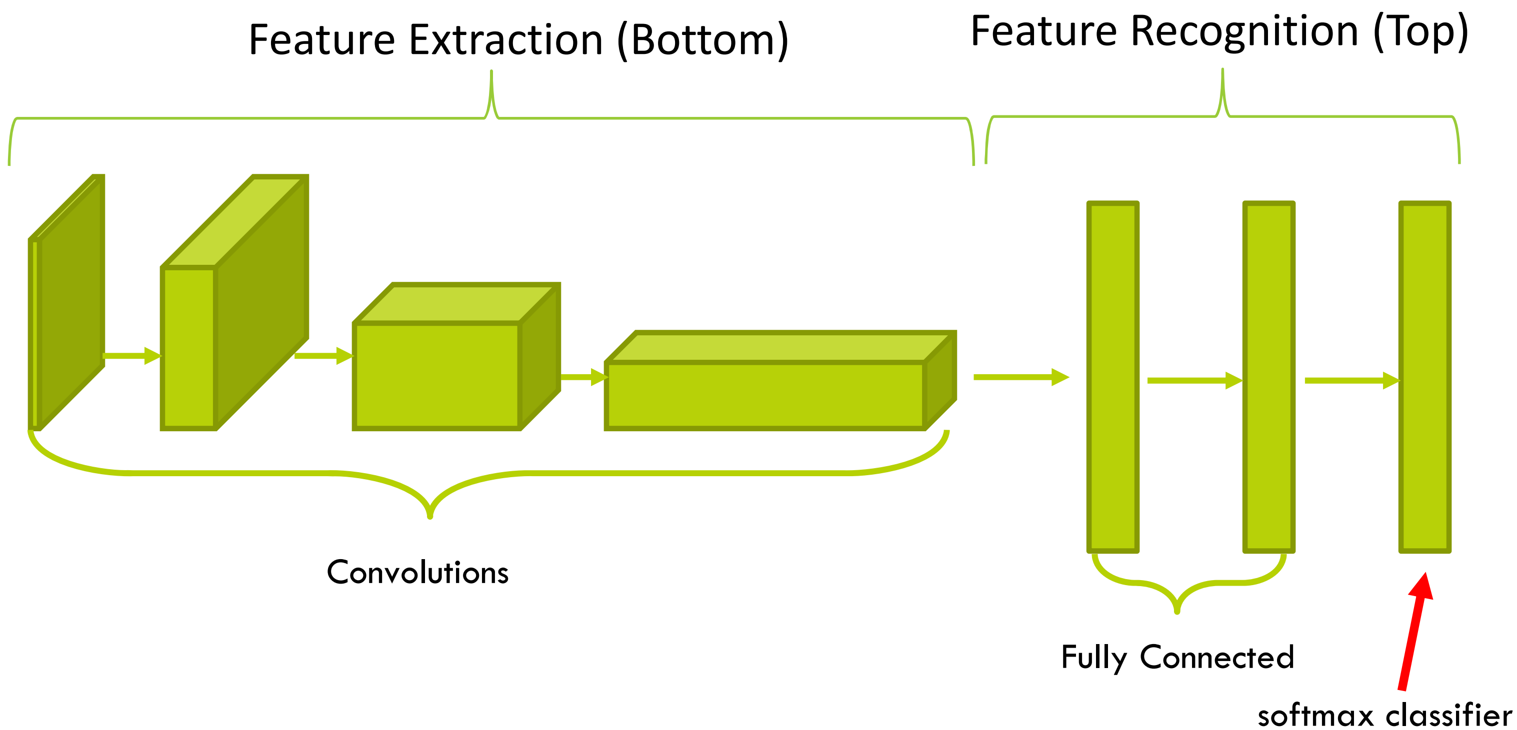
- Models are difficult to train from scratch
  - Huge datasets (like ImageNet)
  - Long number of training iterations
  - Very heavy computing machinery
  - Time experimenting to get hyper-parameters just right



Transfer Learning

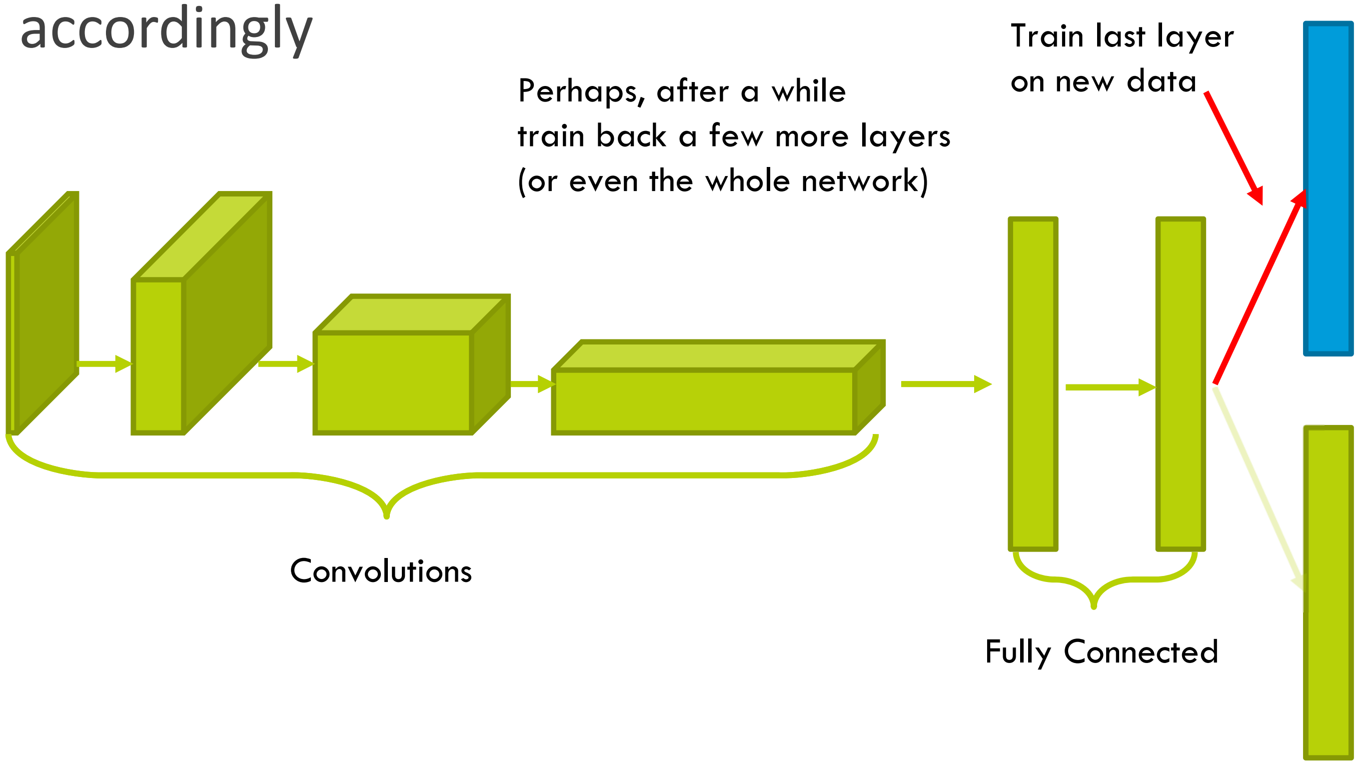
- Basic idea
  - Early layers in a Neural Network are the hardest (i.e. slowest) to train
  - Due to vanishing gradient property
  - But these ”primitive” features should be general across many image classification tasks
  - **Later layers in the network are capturing features** that are more particular to the specific image classification problem.
  - **Later layers are easier (quicker) to train** since adjusting their weights has a more immediate impact on the final result.
  - **Idea**: keep the early layers of a pre-trained network, and re-train the later layers for a specific application

Transfer Learning



Transfer Learning

- Reconstruct Top layer
  - Change the output layer to suit your needs
    - E.g different number of categories, detect different features
  - Adjust the number of layers and nodes of the hidden layers accordingly



Transfer Learning

- Fine Tuning
  - The additional training of a pre-trained network on a specific new dataset is referred to as "***Fine-Tuning***"
  - There are different options on "how much" and "how far back" to fine-tune.
    - Should I train just the very last layer?
    - Go back a few layers?
    - Re-train the entire network (from the starting point of the existing network)?
  - While there are no "hard and fast" rules, there are some guiding principles to keep in mind.

Transfer Learning

- Principle 1:

The more similar your data and problem are to the source data of the pre-trained network, the less fine-tuning is necessary.

- E.g. Using a network trained on ImageNet to distinguish "dogs" from "cats" should need relatively little fine-tuning. It already distinguished different breeds of dogs and cats, so likely has all the features you will need.

Transfer Learning

- Principle 2:

The more data you have about your specific problem, the more the network will benefit from longer and deeper fine-tuning.

- E.g. If you have only 100 dogs and 100 cats in your training data, you probably want to do very little fine-tuning. If you have 10,000 dogs and 10,000 cats you may get more value from longer and deeper fine-tuning.

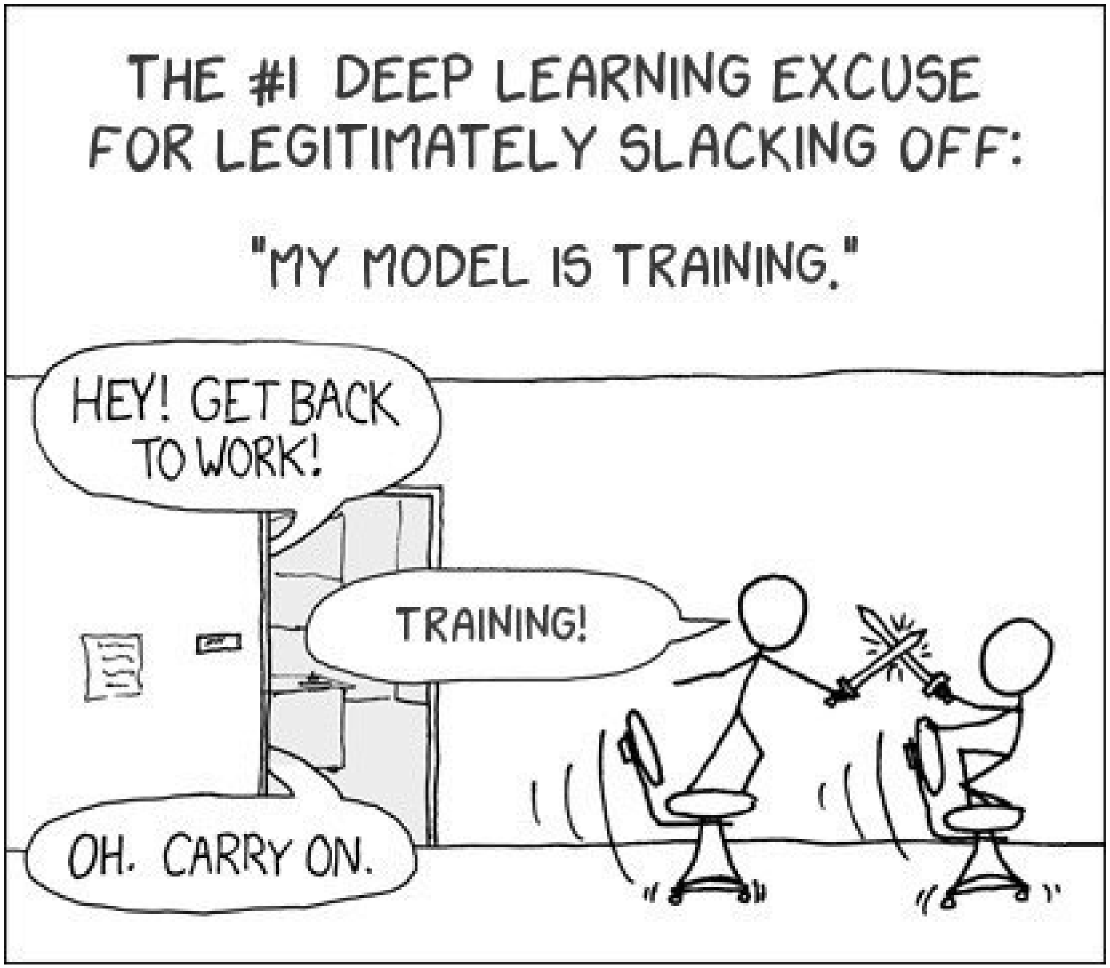
Transfer Learning

- Principle 3:

If your data is substantially different in nature than the data the source model was trained on, Transfer Learning may be of little value.

- E.g. A network that was trained on recognizing typed Latin alphabet characters would not be useful in distinguishing cats from dogs. But it likely would be useful as a starting point for recognizing Cyrillic Alphabet characters.

Training!


THE #1 DEEP LEARNING EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY MODEL IS TRAINING."
HEY! GET BACK TO WORK!
TRAINING!
OH. CARRY ON.

Activity 6 –
Classifying Flowers with pre-trained VGG16

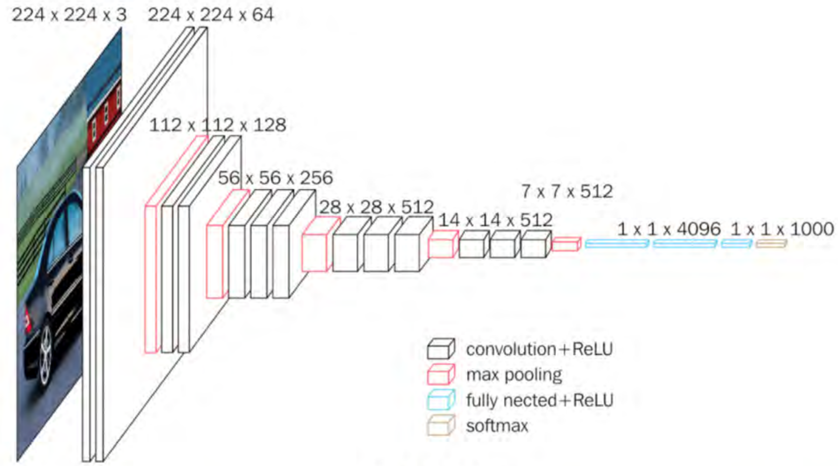

Daisy

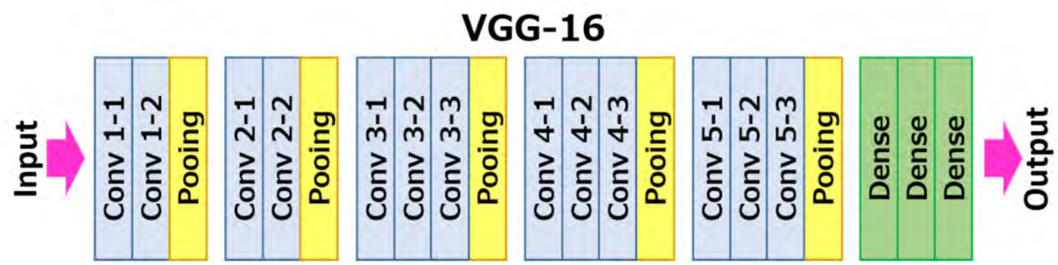

Rose


Dandelion


Sunflower


Tulip





Step 1:
Watch and listen to the instructor's demonstration


5 mins

Step 2:
Work through the activities

Individual Activity


15 mins

79

Activity 7 –
Classifying Flowers with transfer learning using pretrained VGG16

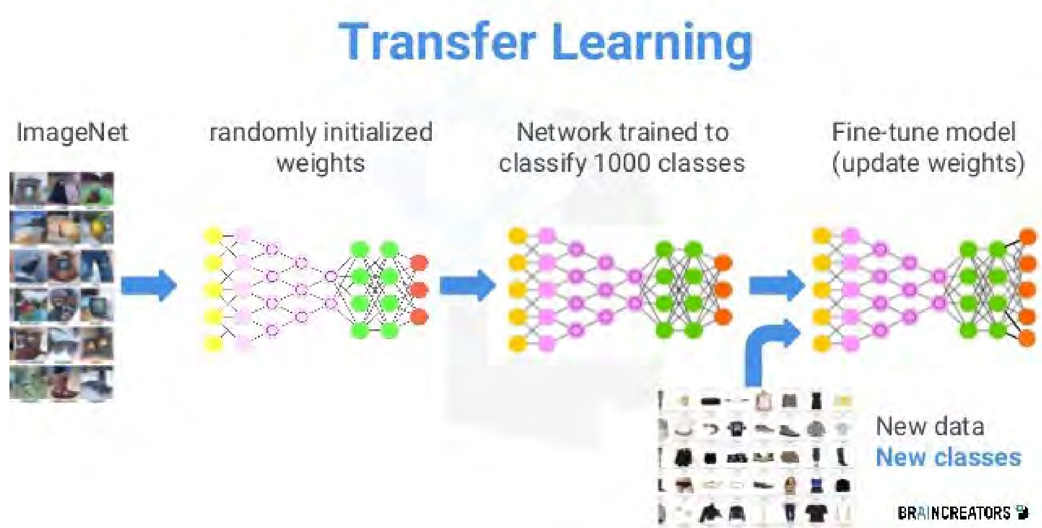

Daisy


Rose


Sunflower


Tulip


Dandelion



Step 1:
Watch and listen to the instructor's demonstration


10 mins

Step 2:
Work through the activities

Individual Activity


30 mins

80

Quiz

https://bit.ly/kw_poll

SCAN ME

Thank you