Deep Learning with Python

DAY 1

ANN AND CNN

Introduction of trainer

Programmes

Lecture 1 – Artificial Neural Network
Activity 1 – Classifying Greyscale Handwritten Digits
Activity 2 – Classifying Fashion

Lecture 2 – Image Convolution
Activity 3 – Classifying Fashion with CNN
Activity 4 – Classifying Pokemon with CNN

Lecture 3 – Pre-trained model and transfer learning
Activity 5 – Classifying Pokemon with pretrained VGG16
Activity 6 – Classifying Pokemon with transfer Learning using pretrained VGG16

Programmes

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
- Anaconda
- Any text editor (Pycharm)

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

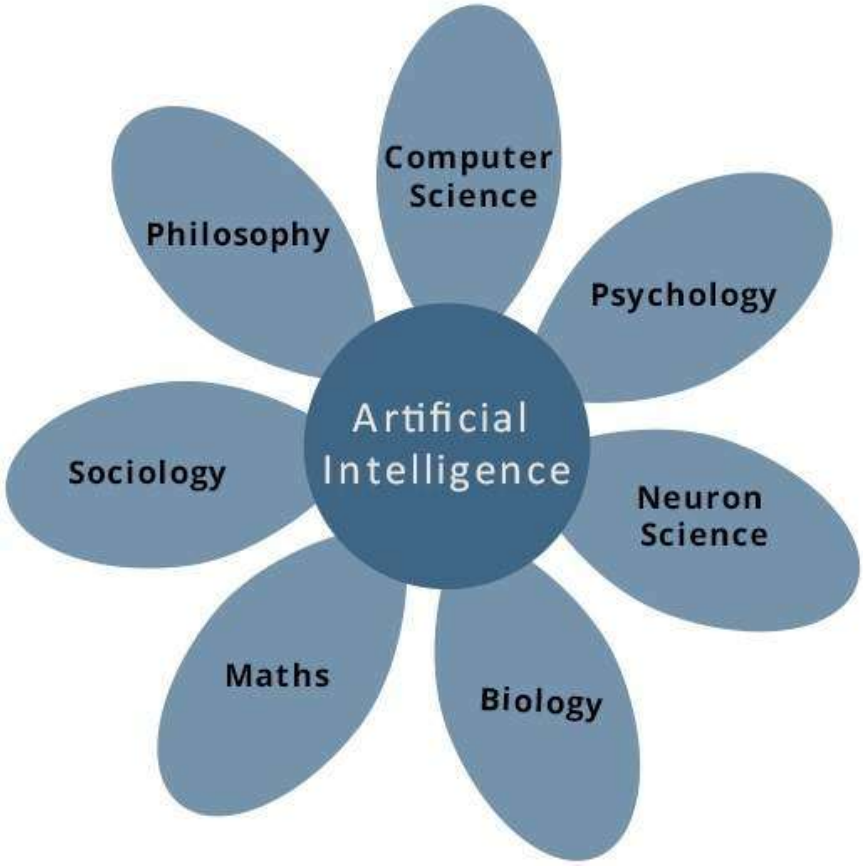
Workshop materials: http://bit.ly/3cIecYU

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

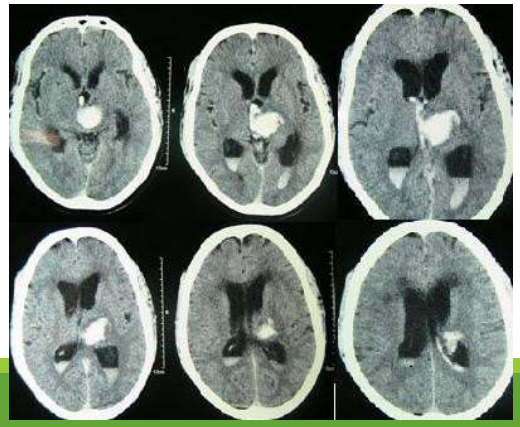
- Contribution to AI



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

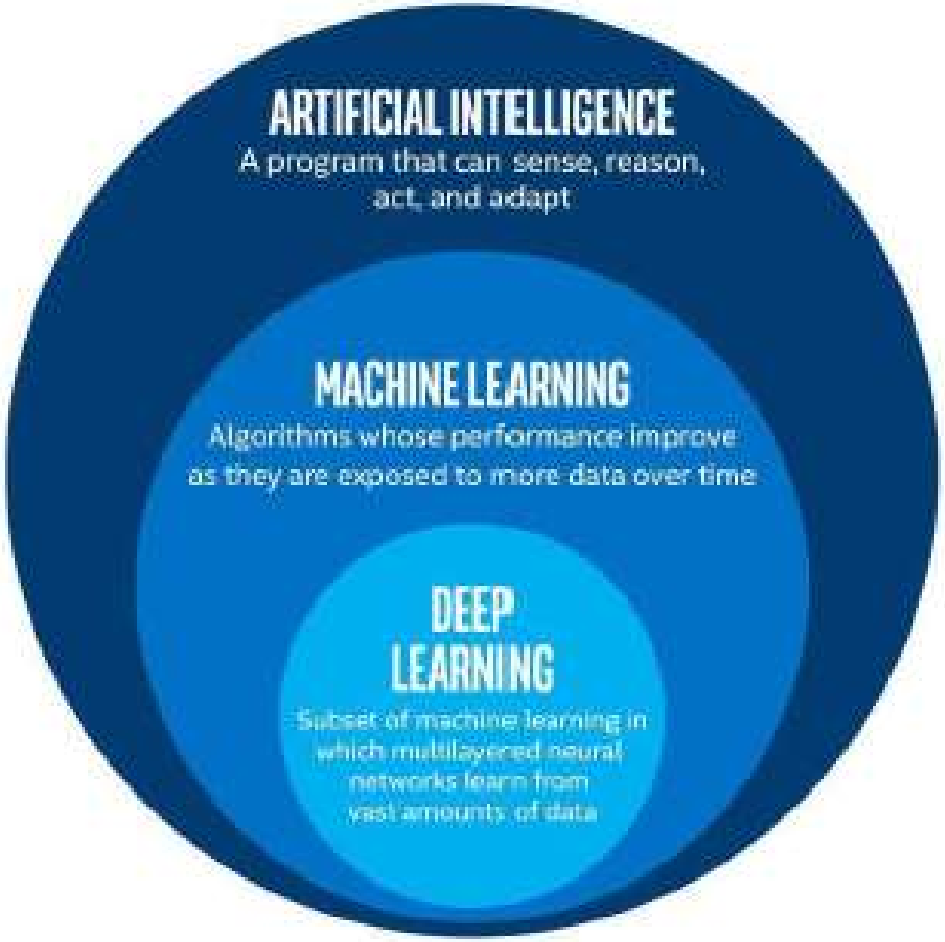
Artificial Intelligence

- 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

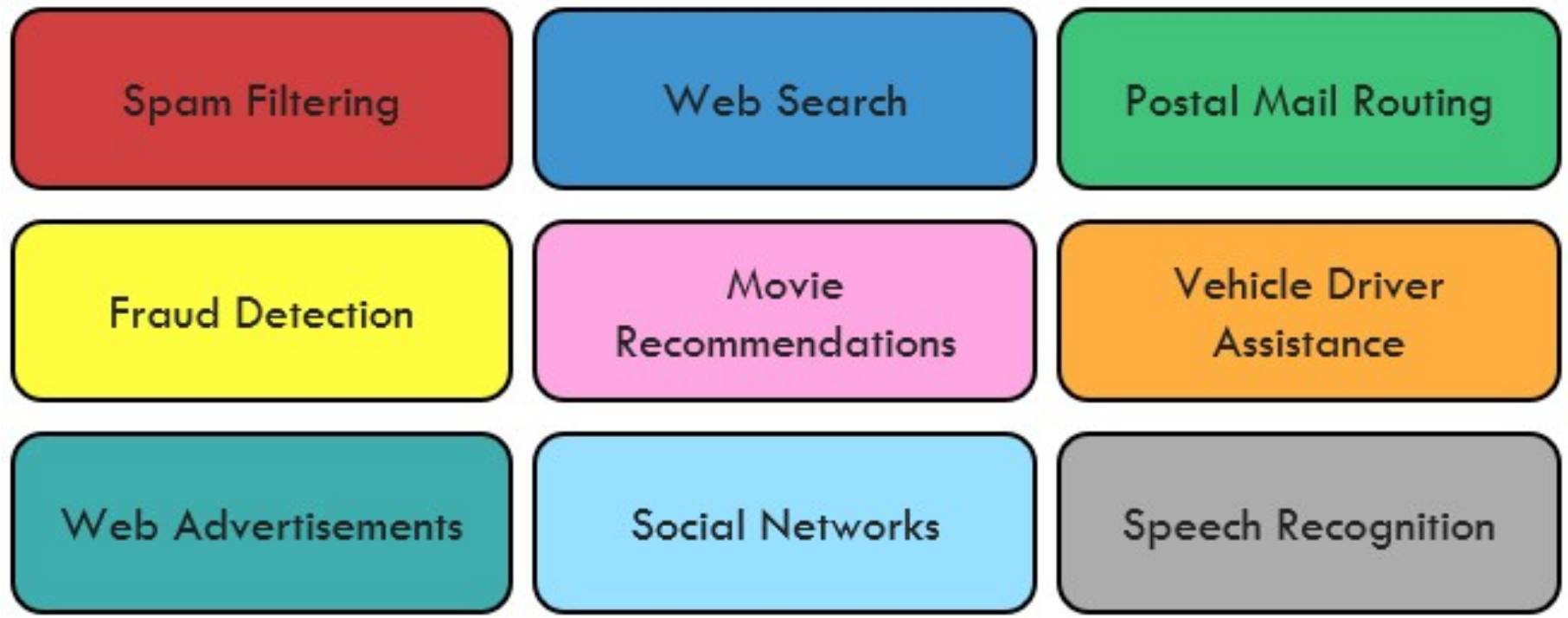
- **Two main types of learning**
  - Supervised Learning
    - Data points have known outcome
    - Goal is to make predictions - Classify and Regression
  - Unsupervised Learning
    - Data points have unknown outcome
    - Goal is to find structure within the data – Clustering
- **Other types of learning**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

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

Machine Learning

- Applications in our daily lives



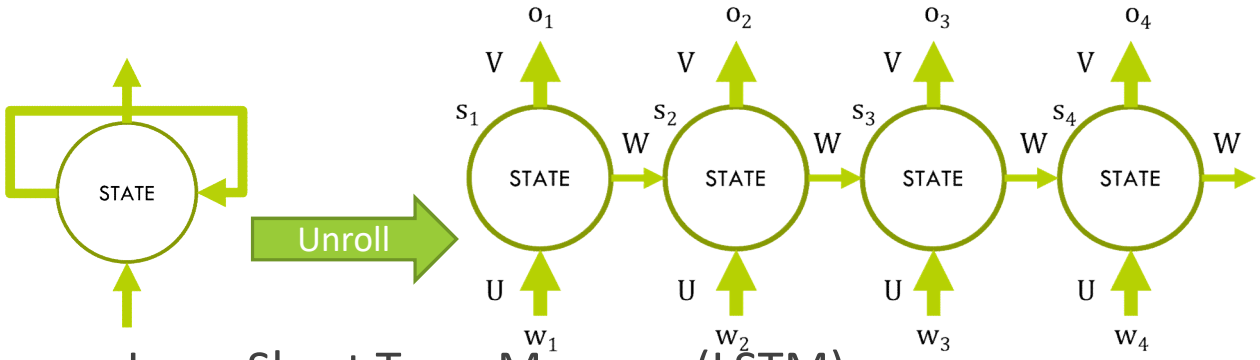
Deep Learning

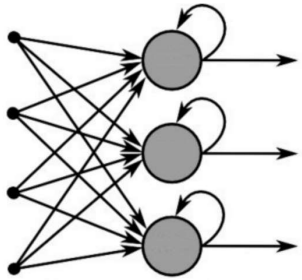
- Deep learning is a class of machine learning algorithms that:
  - use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
  - learn in supervised (e.g., classification) and/or unsupervised (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)

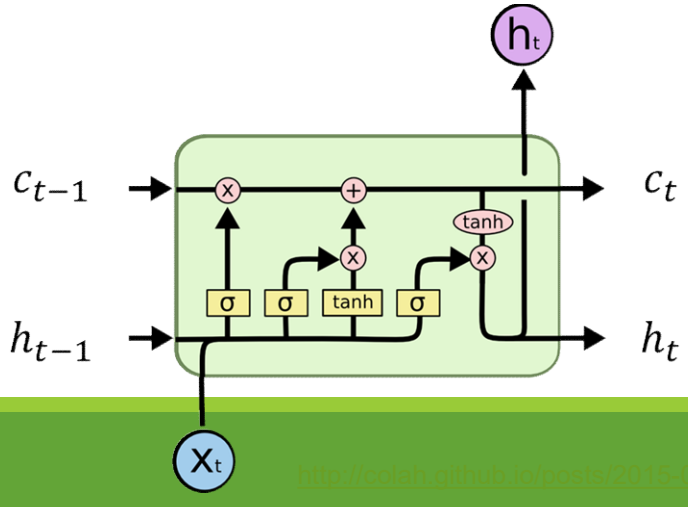




Recurrent Neural Network

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

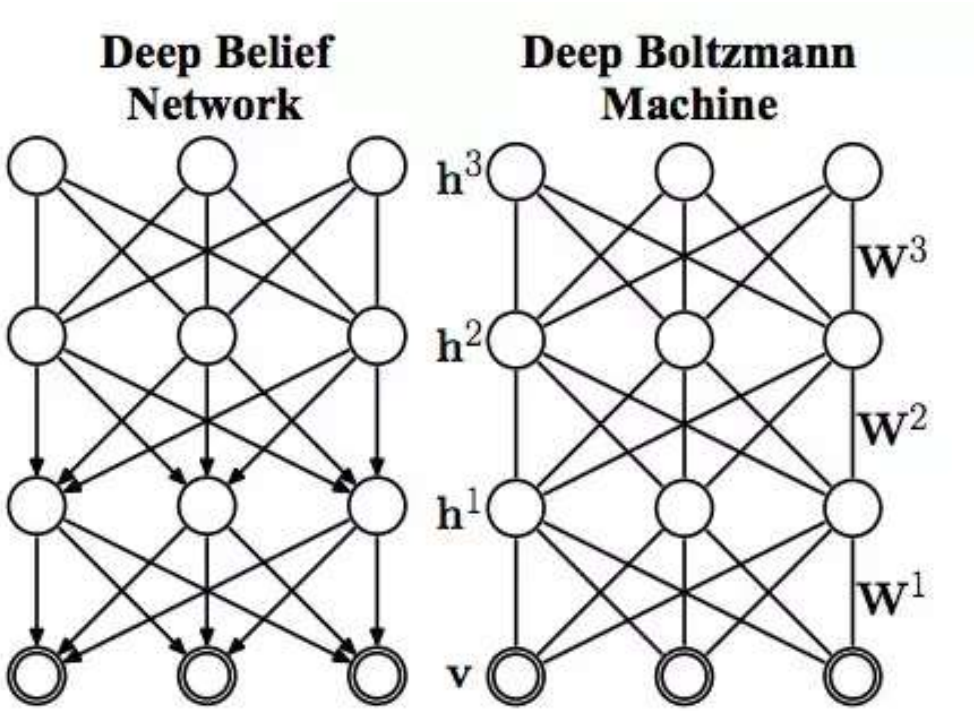
- Long-Short Term Memory (LSTM)



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

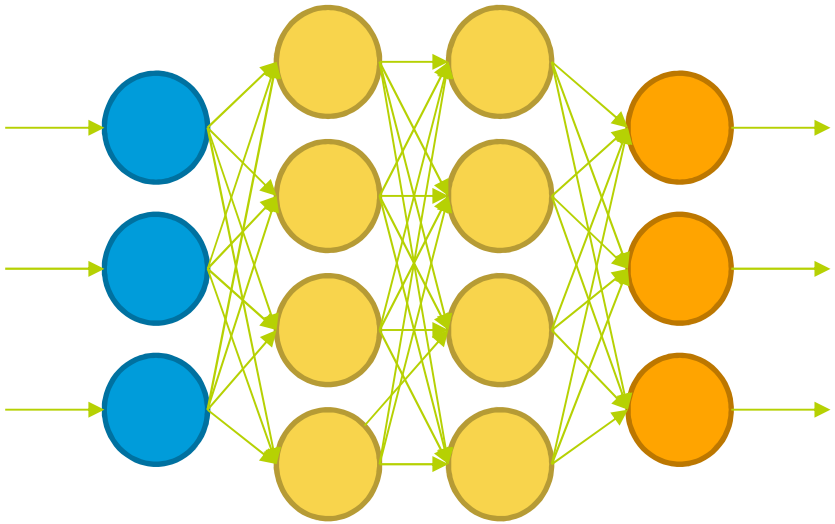


https://www.youtube.com/watch?v=MnGXXDjGNd0

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

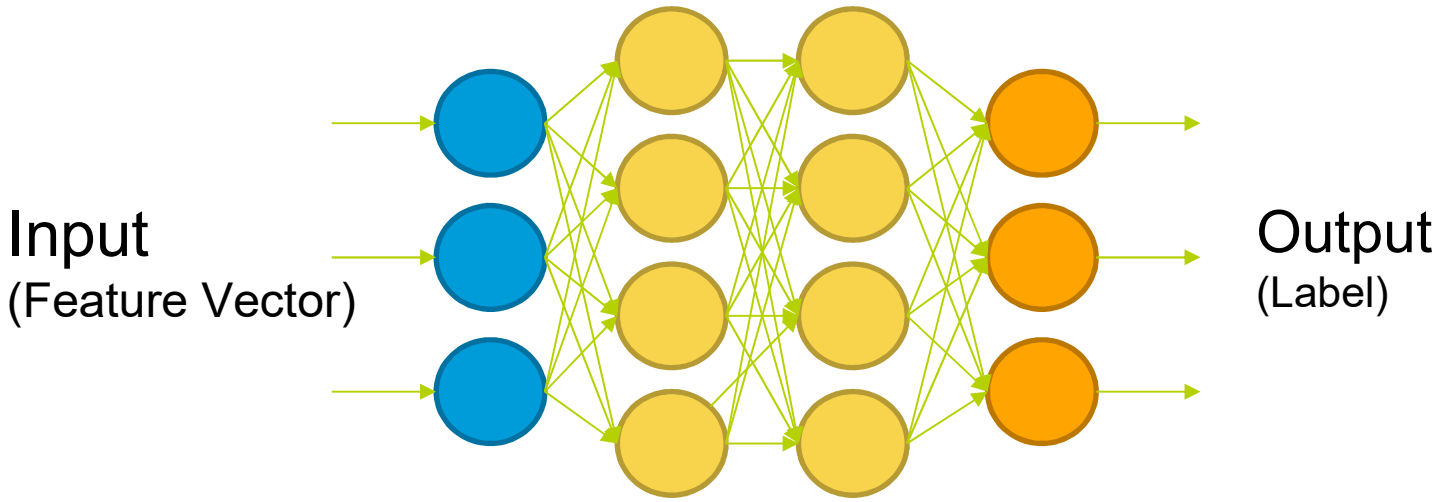
ANN

- 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



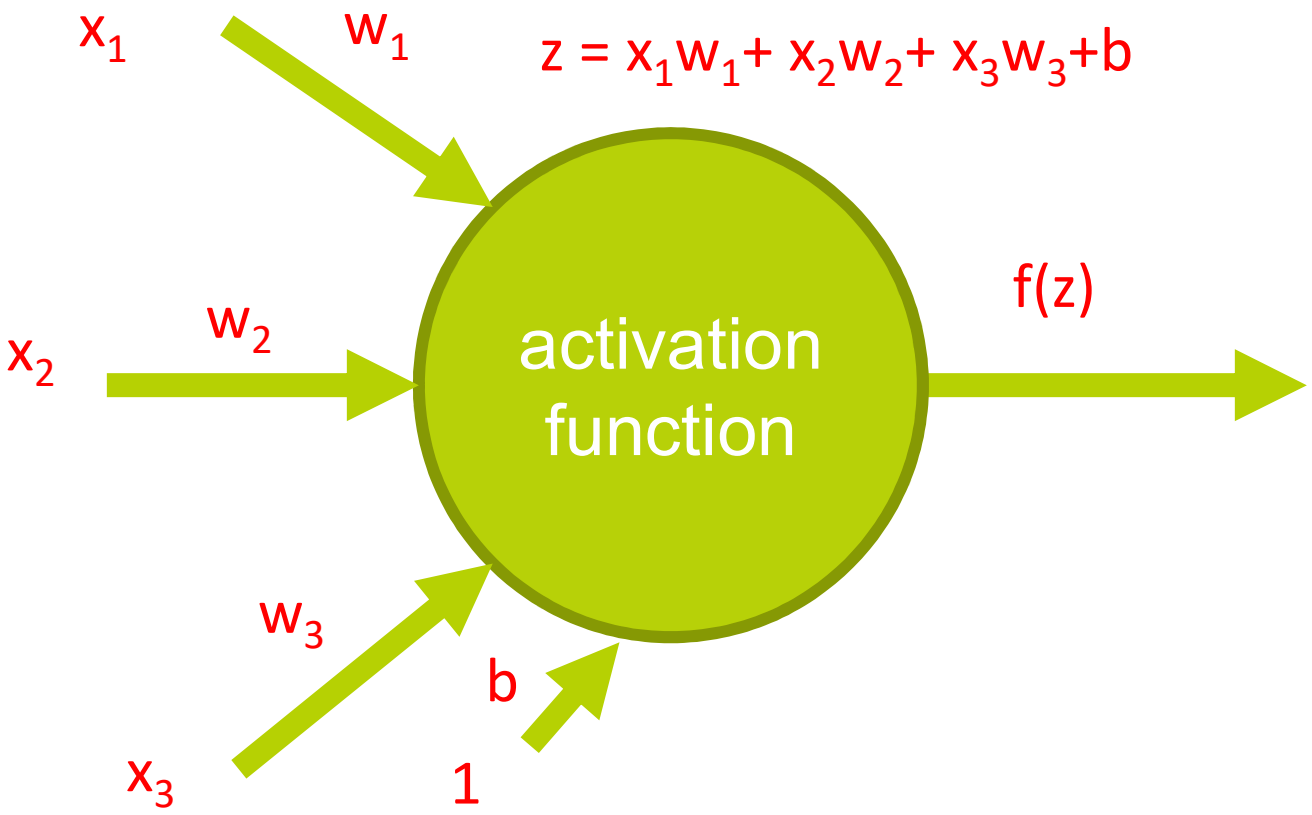
ANN

- 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

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

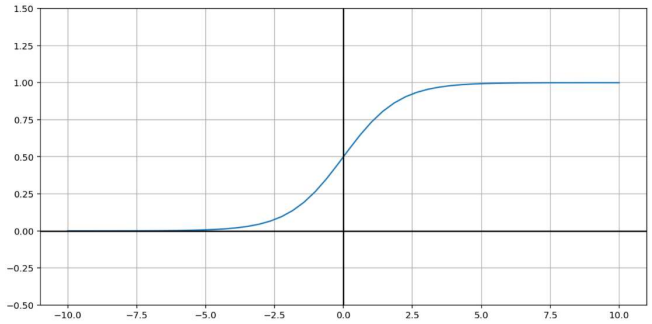
$$z = b + x^T w$$

$$a = f(z)$$

ANN

- Relation to Logistic Regression

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



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

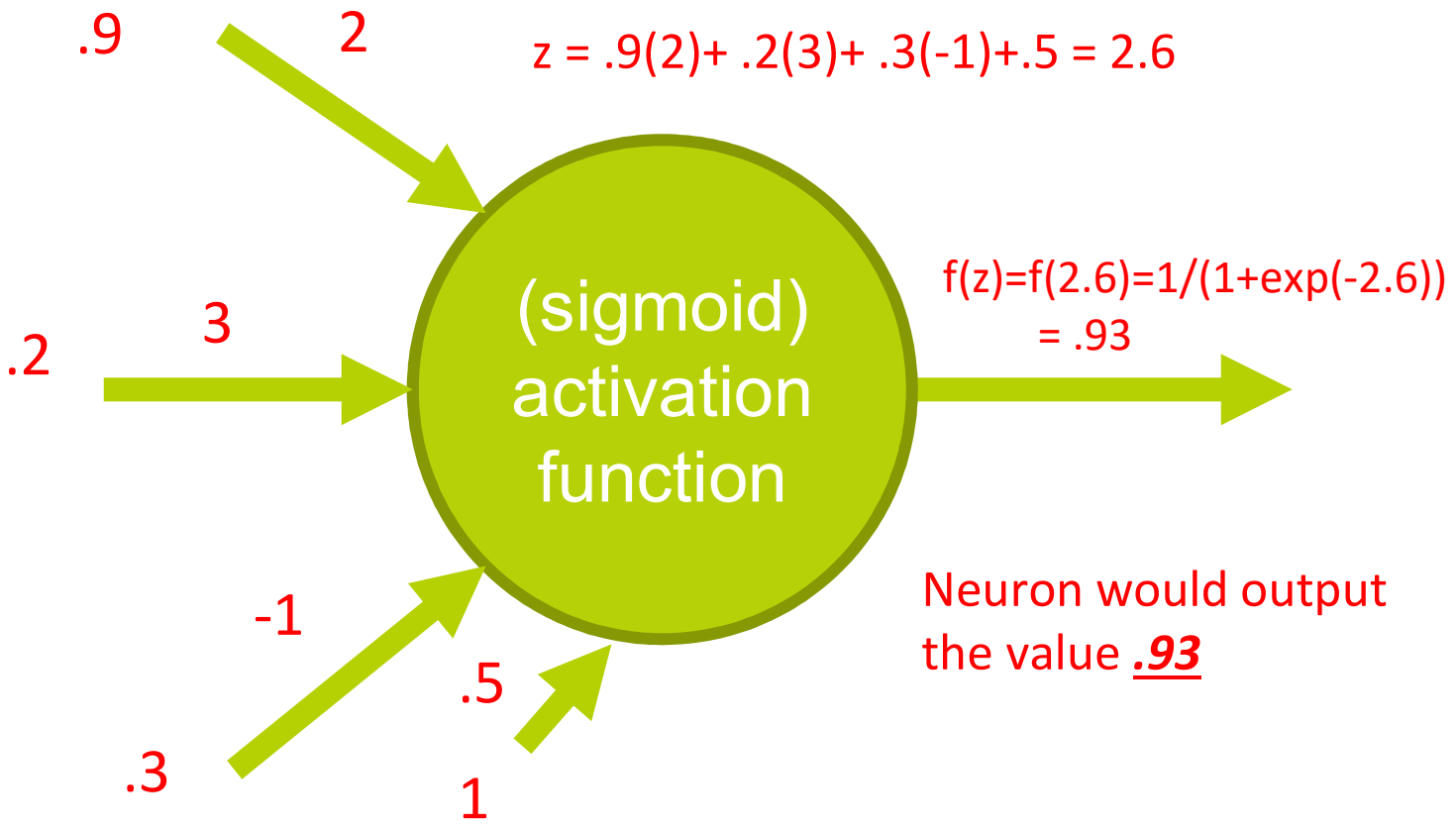
Then a neuron is simply a ”unit” of logistic regression!

weights ⇔ coefficients inputs ⇔ variables

bias term ⇔ constant term

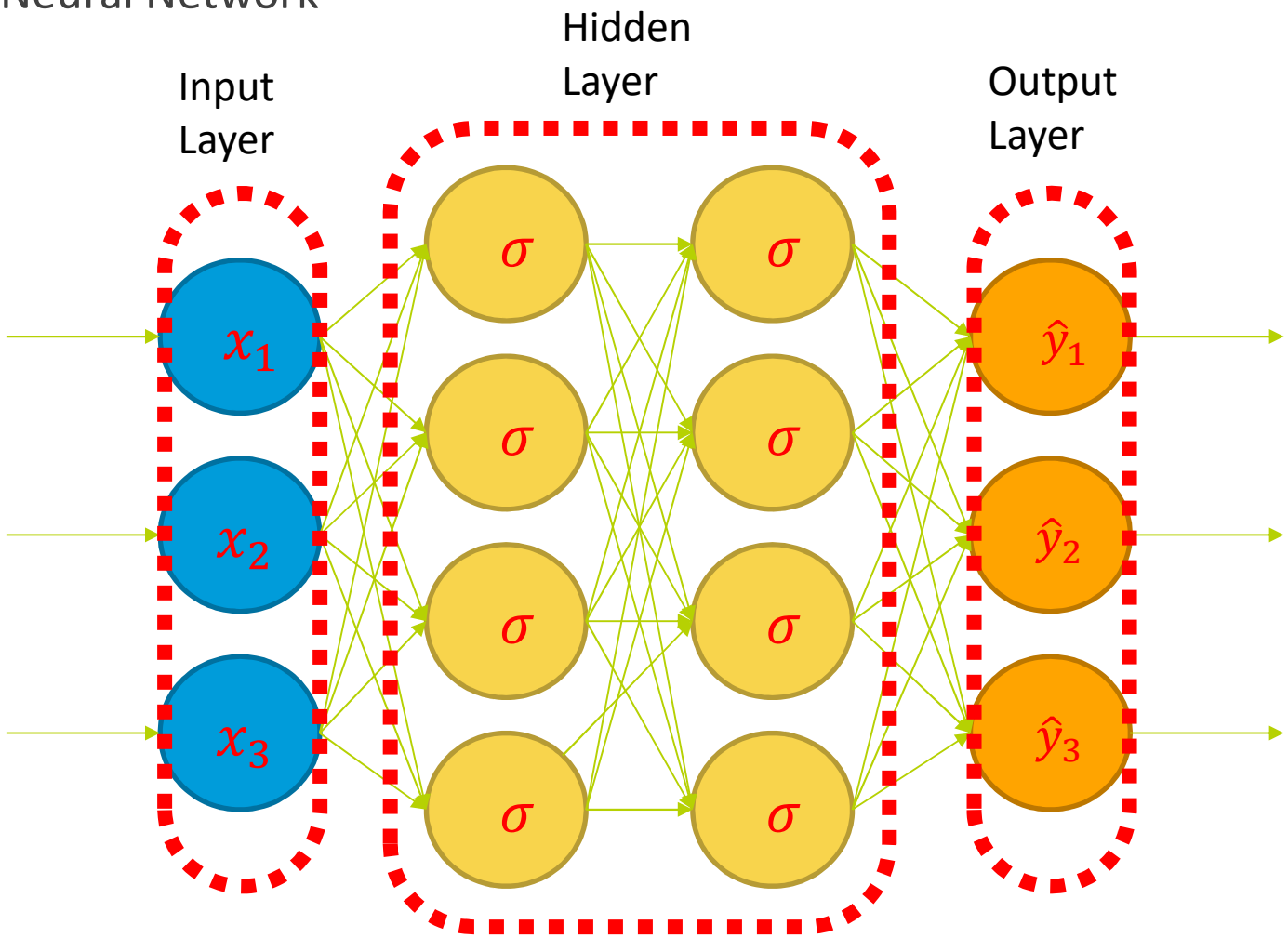
ANN

- An Example



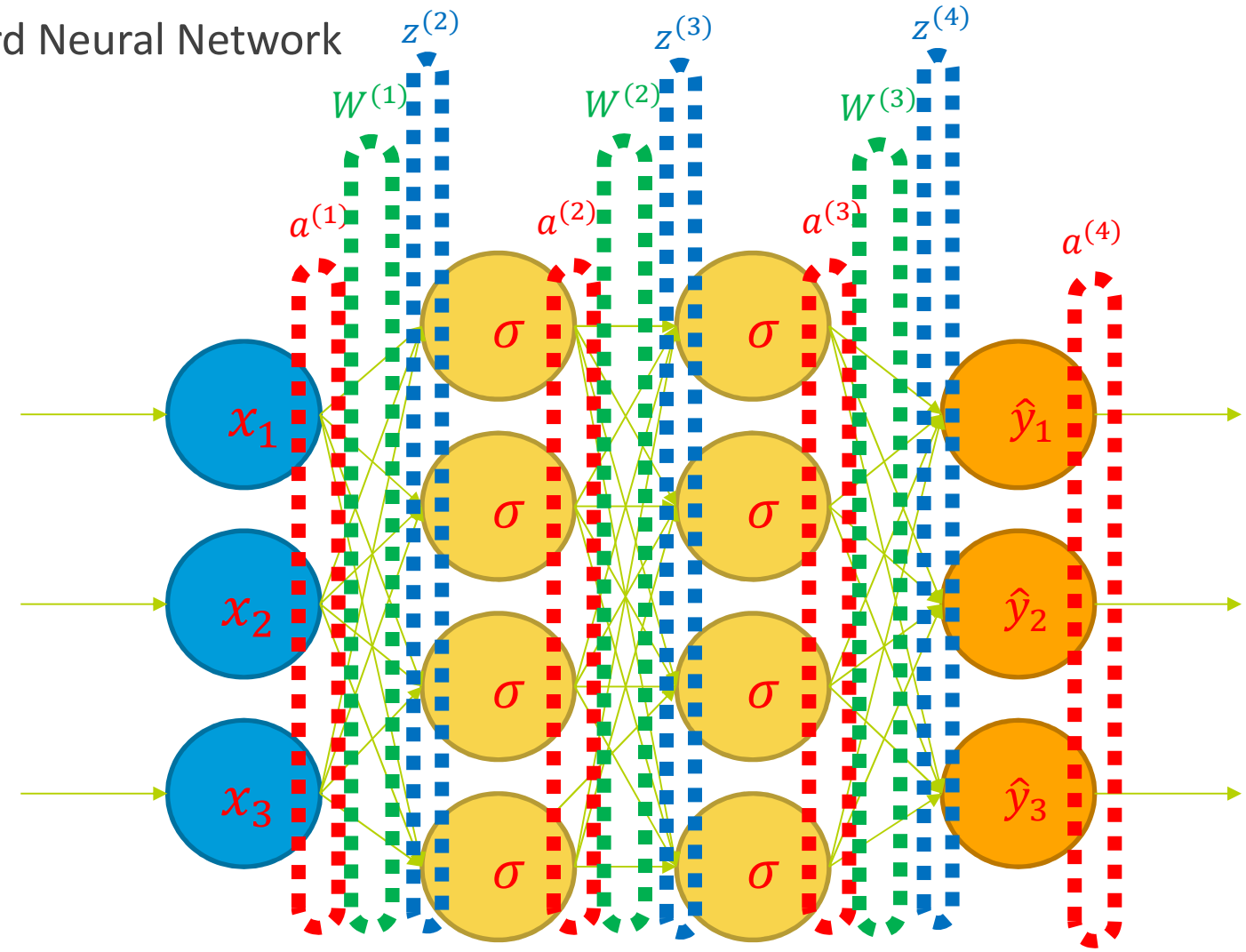
ANN

- Feedforward Neural Network
  - Layers



ANN

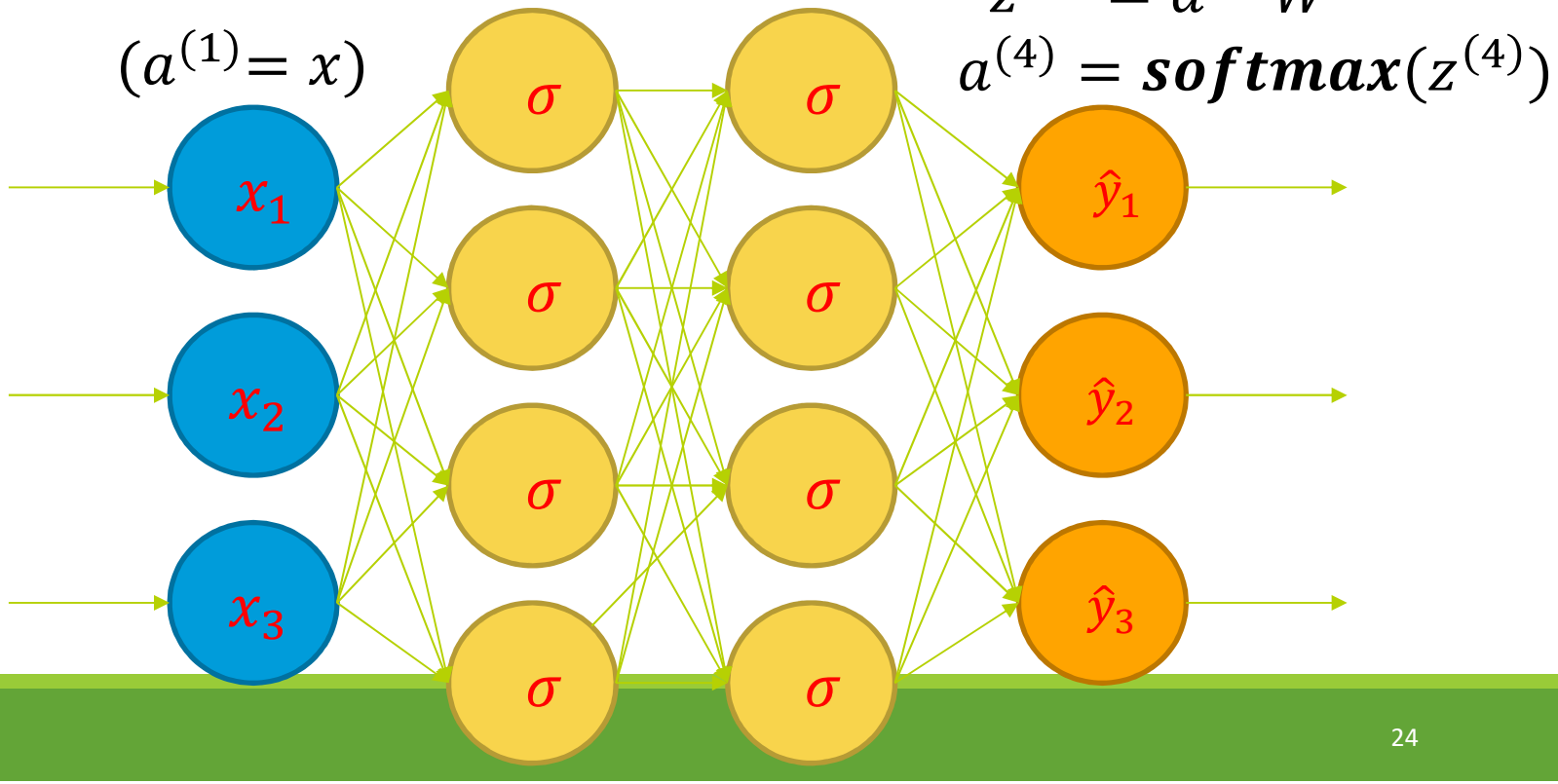
- Feedforward Neural Network
  - Values



ANN

- Feedforward Neural Network
  - Calculations

$$z^{(2)} = a^{(1)}W^{(1)} \quad z^{(3)} = a^{(2)}W^{(2)}$$
$$a^{(2)} = \sigma(z^{(2)}) \quad a^{(3)} = \sigma(z^{(3)})$$
$$z^{(4)} = a^{(3)}W^{(3)}$$
$$a^{(4)} = \boldsymbol{softmax}(z^{(4)})$$



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

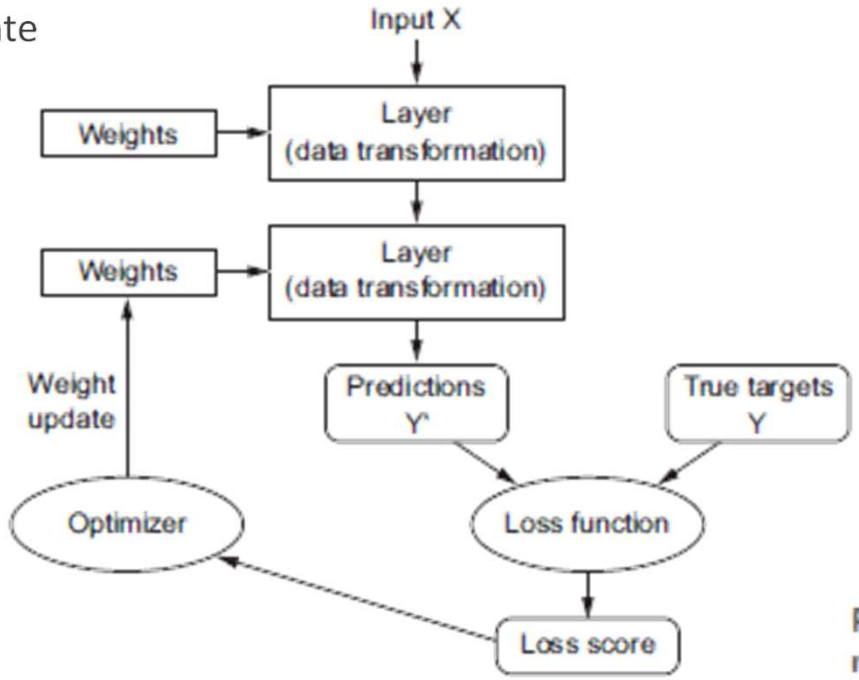
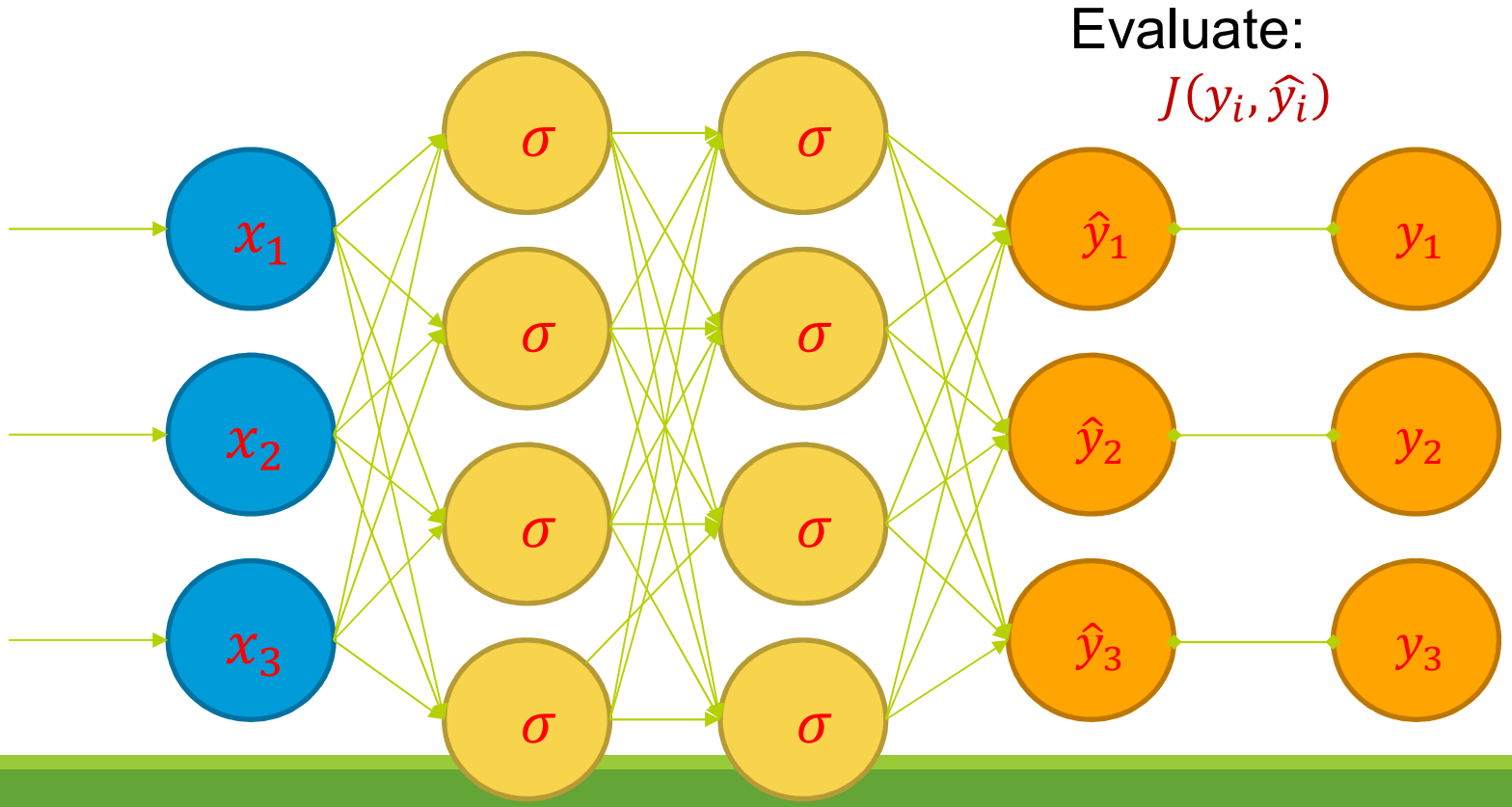


Figure 3.1 Relationship between the network, layers, loss function, and optimizer

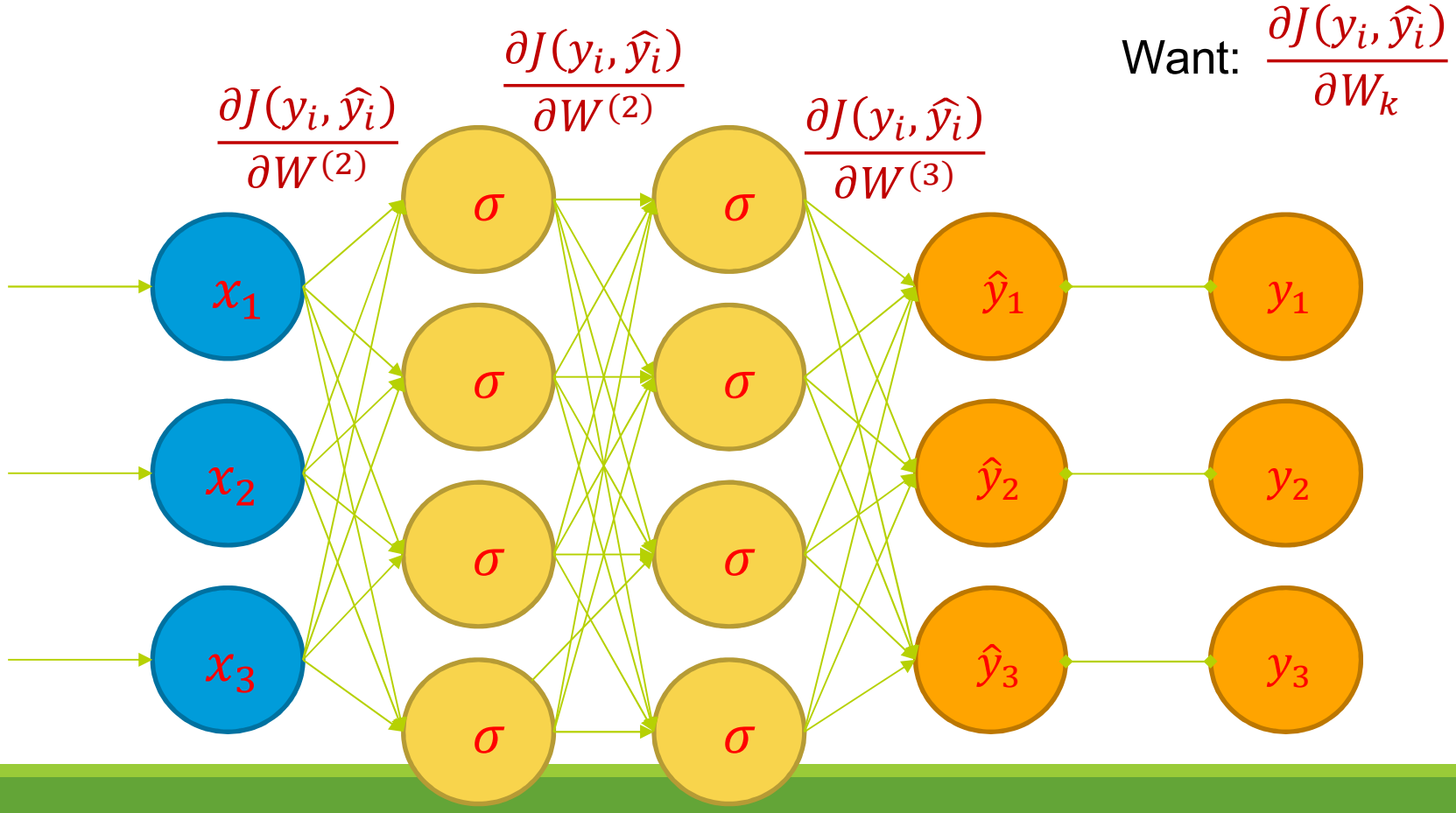
ANN

- 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

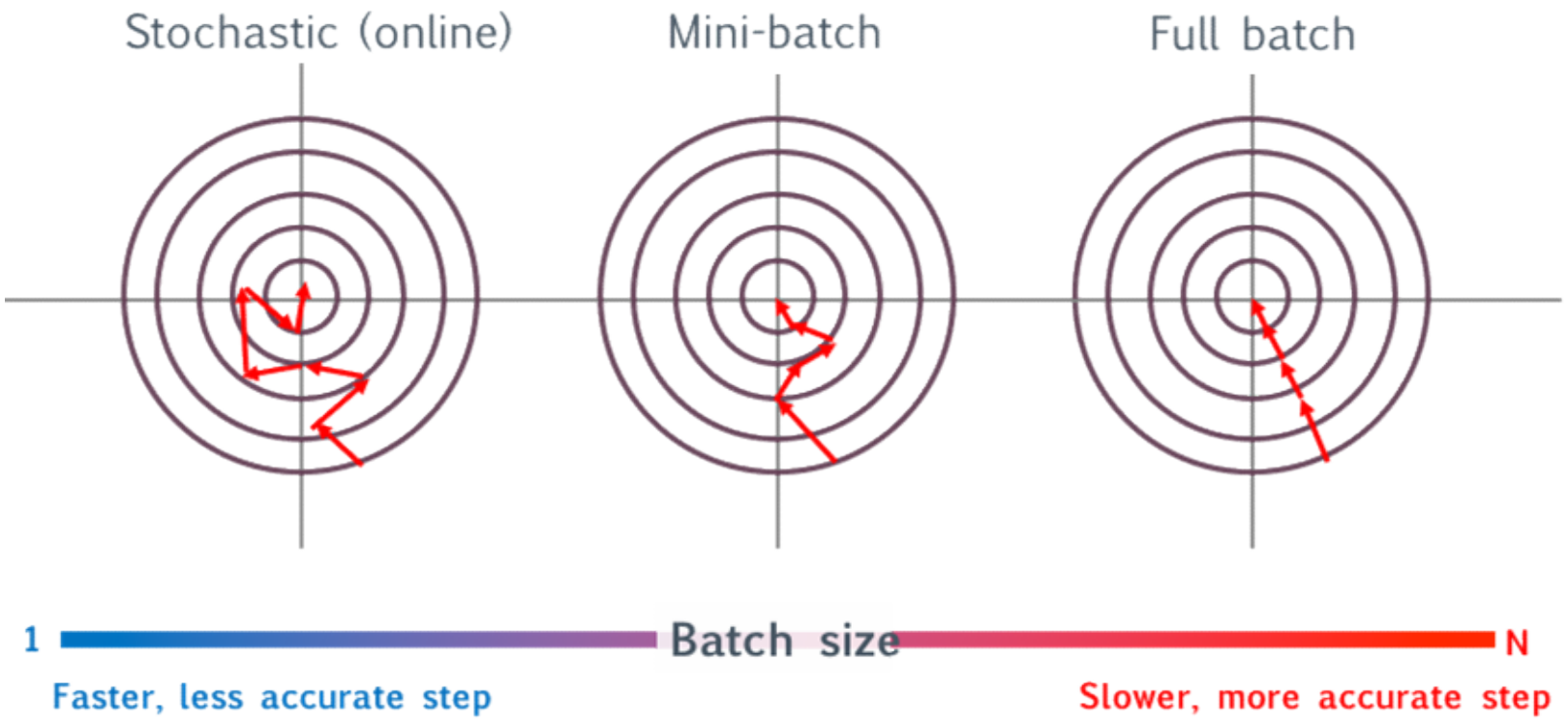
- Training the Neural Network - Backpropagation
  - Step 4: Update parameters by taking a step in the opposite direction

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

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

ANN

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

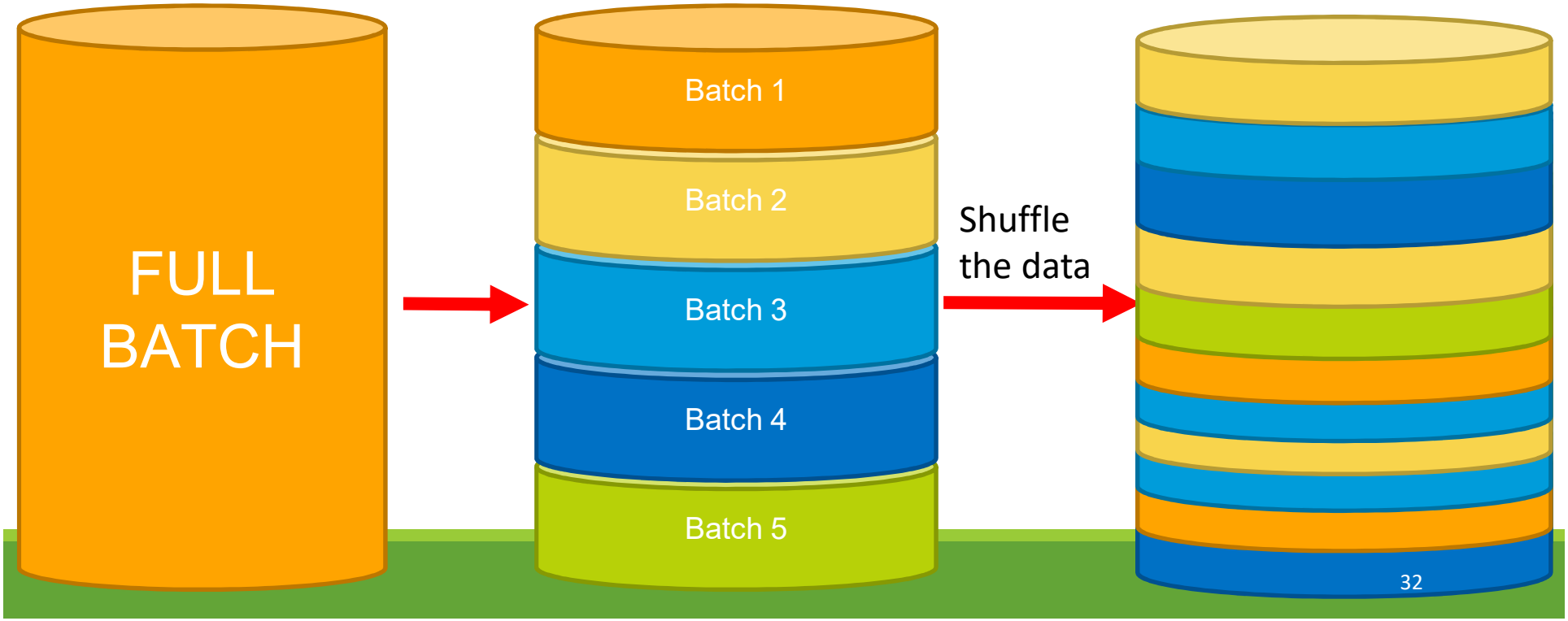


ANN

- 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

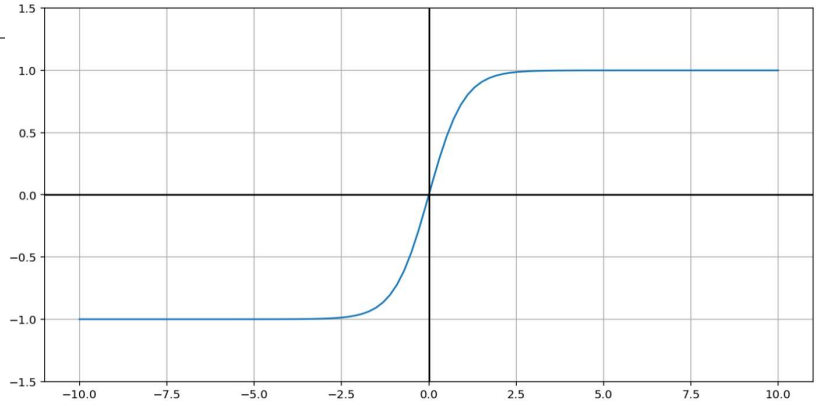
$$\frac{\partial J}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

  - 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

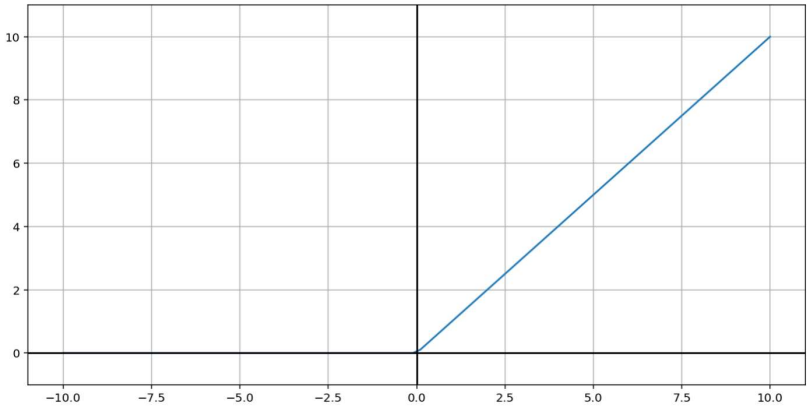
- Other Activation Functions
  - Hyperbolic tangent function

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



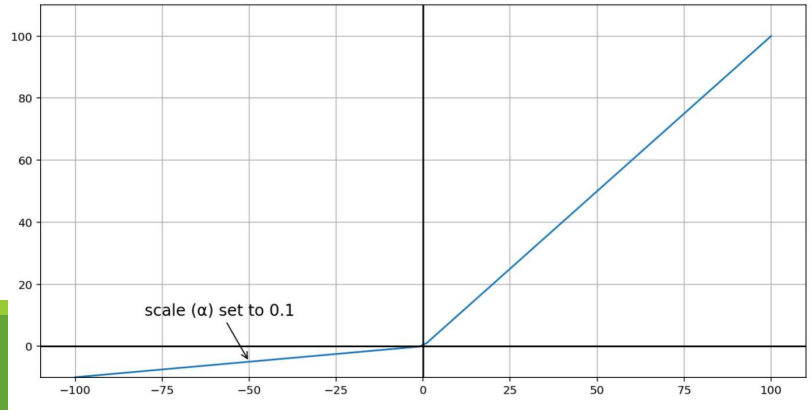
  - Rectified Linear Unit (ReLU)

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



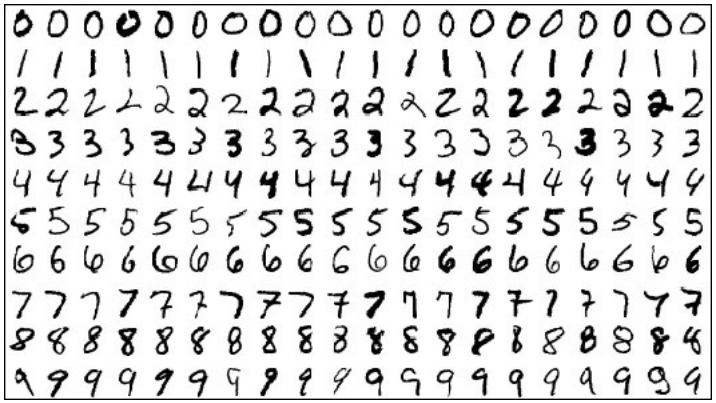
  - "Leaky" Rectified Linear Unit (LReLU)

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



Activities

- Activity 1 - **Classifying Grayscale Handwritten Digits**
-



- Activity 2 - **Classifying Fashion**

Quiz

Image Convolution

- Basic concept of Convolution – Kernels
  - A *kernel* is a grid of weights "overlaid" on image, centered on one pixel
  - Each weight multiplied with pixel underneath it
  - Output over the centered 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

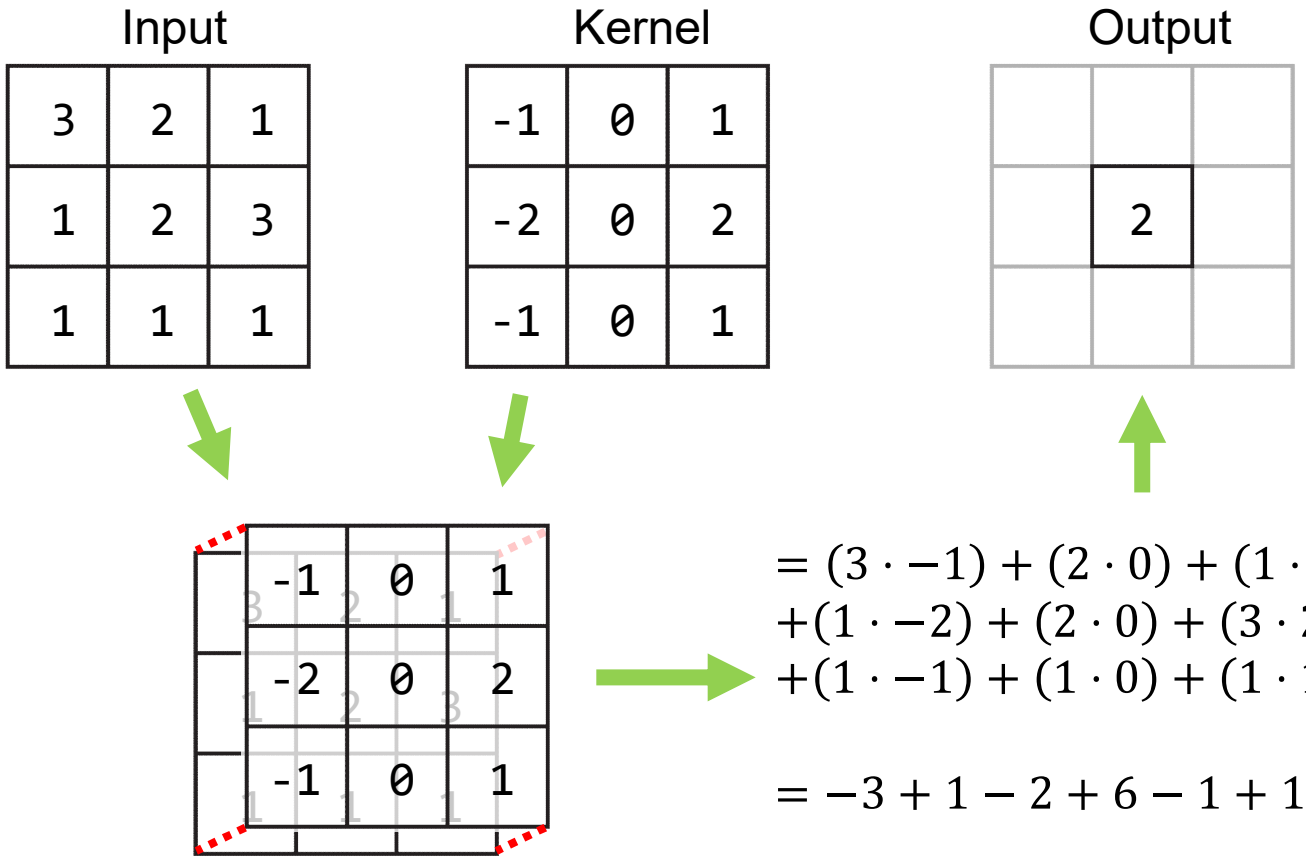


Image Convolution

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

Vertical Line Detector

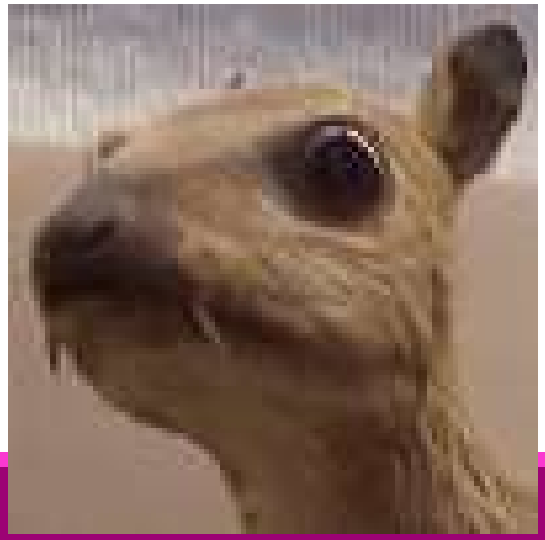
-1	1	-1
-1	1	-1
-1	1	-1

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

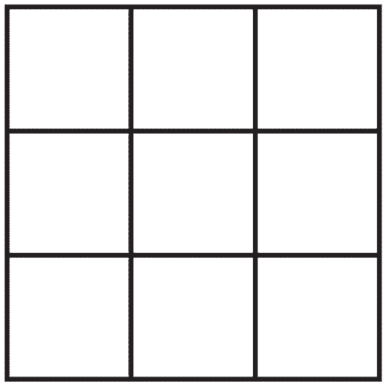
=

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

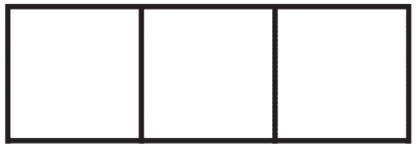
Image Convolution

- 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

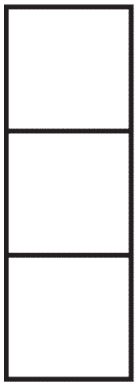


Image Convolution

- 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

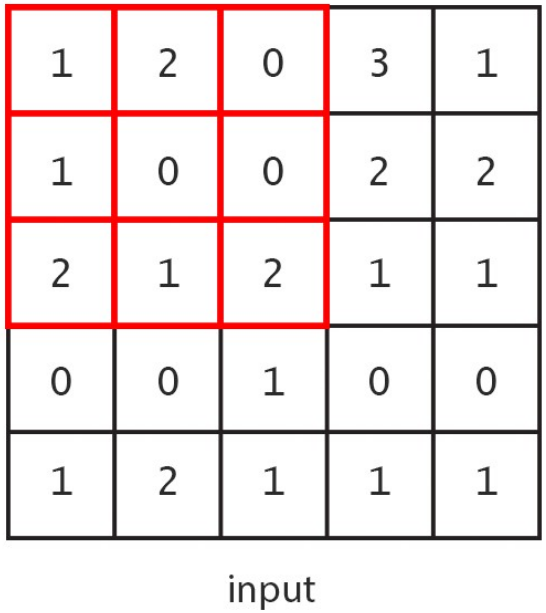
Image Convolution

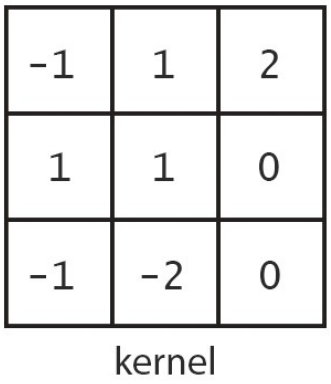
- Basic concept of Convolution – Settings
  - *Padding*
    - Using Kernels directly, there will be an “edge effect”
    - Pixels near the edge will not be used as “center pixels” since there are not enough surrounding pixels
    - Padding adds extra pixels around the frame
    - So every pixel of the 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





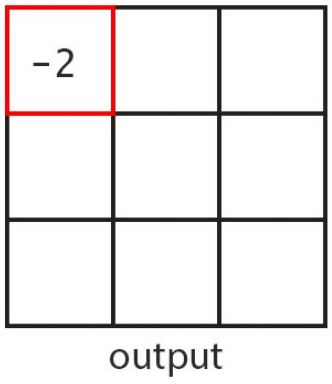


Image Convolution

- 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

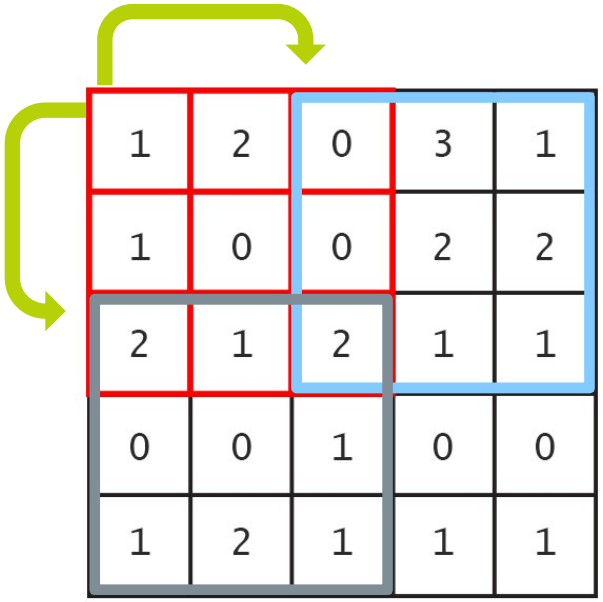
Image Convolution

- 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

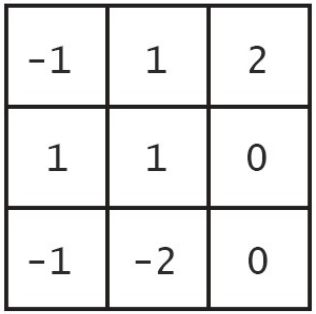
Image Convolution

- Basic concept of Convolution – Settings
  - *Stride*

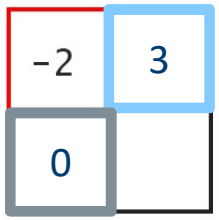
Stride 2 Example – No Padding



input



kernel



output

Image Convolution

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

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

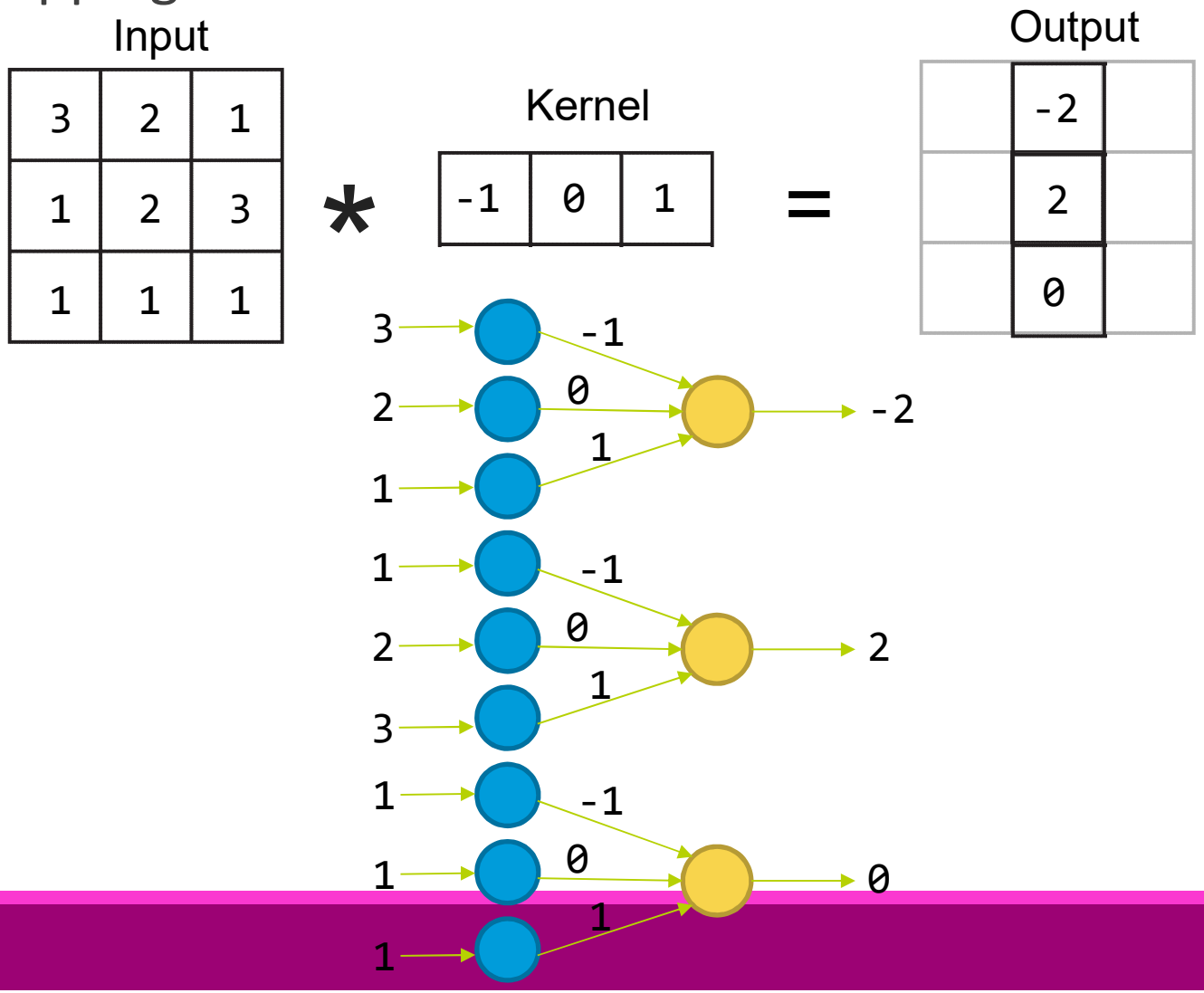
$where,$
W – width of image
K – kernel size
P – padding size
S – stride size

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.

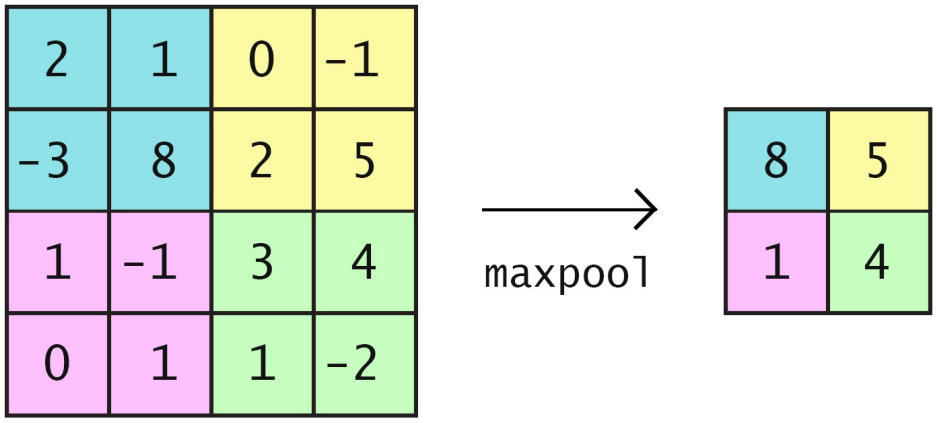
Other important layer: Pooling Layer

- **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)))
```

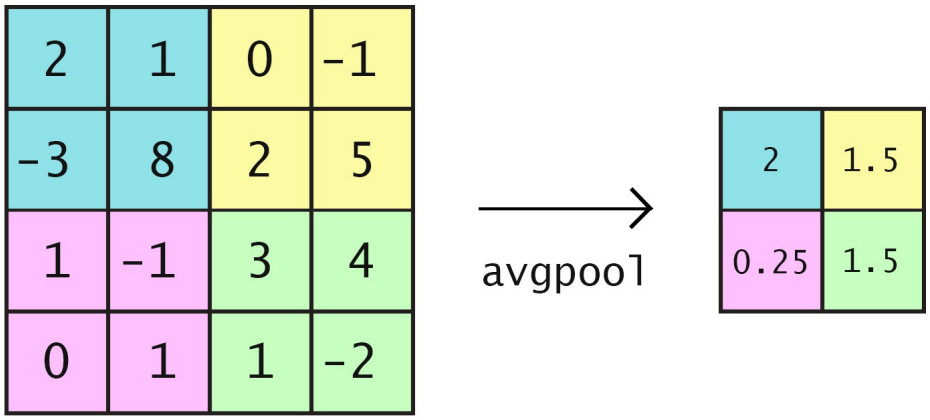
Pooling Layer

- **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**

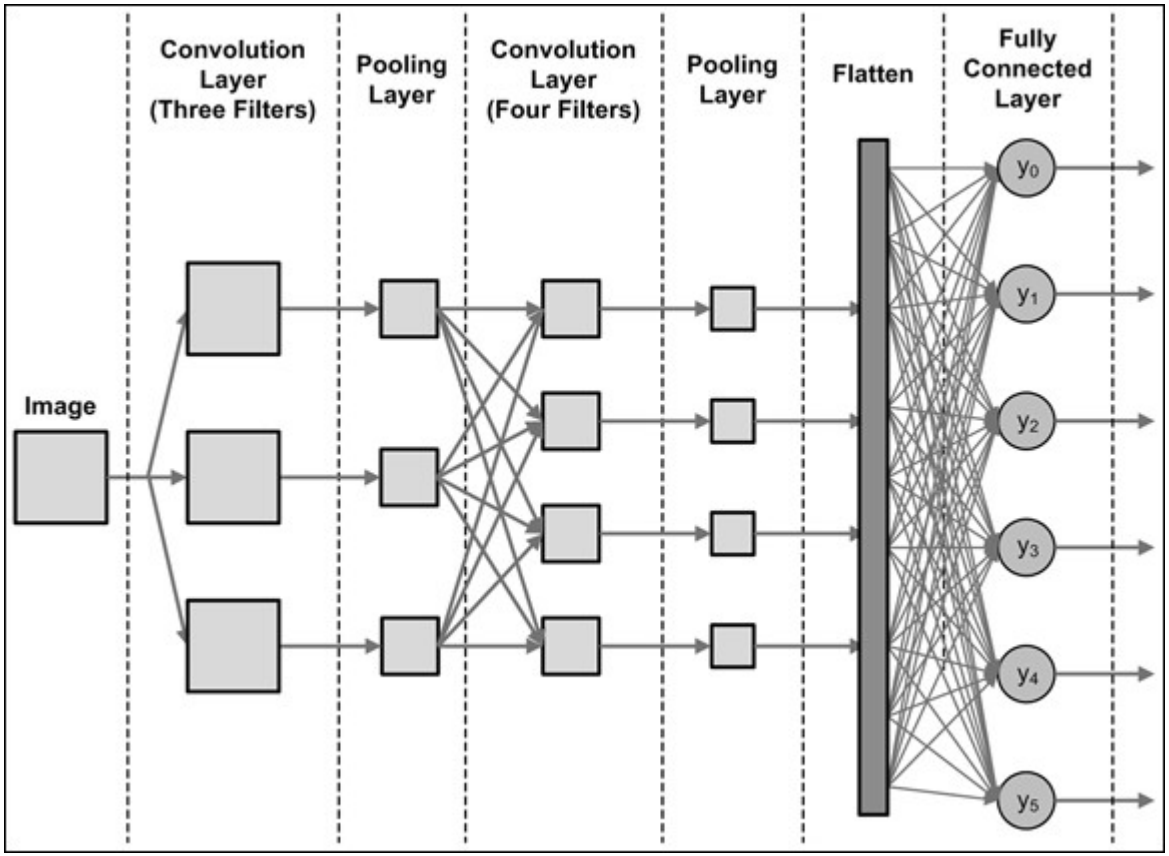
```
network.add(layers.Dropout(0.25))
```

Flatten layer

- A special layer that ‘flattens’ the inputs into a 1D array or list.

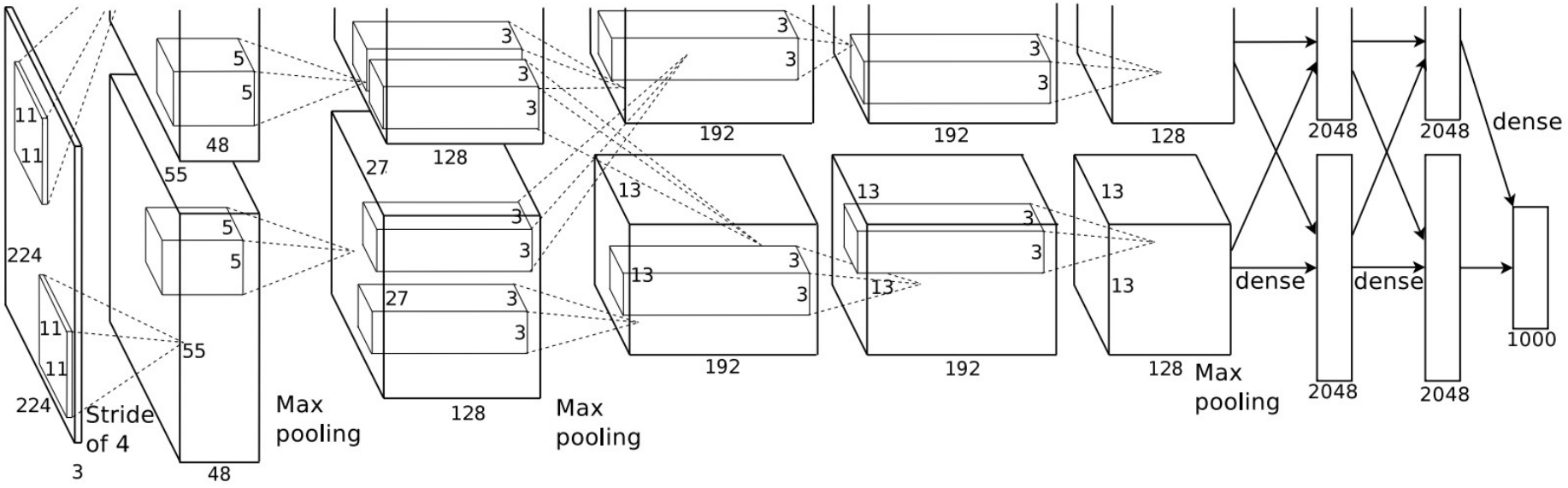
Convolution Neural Network

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



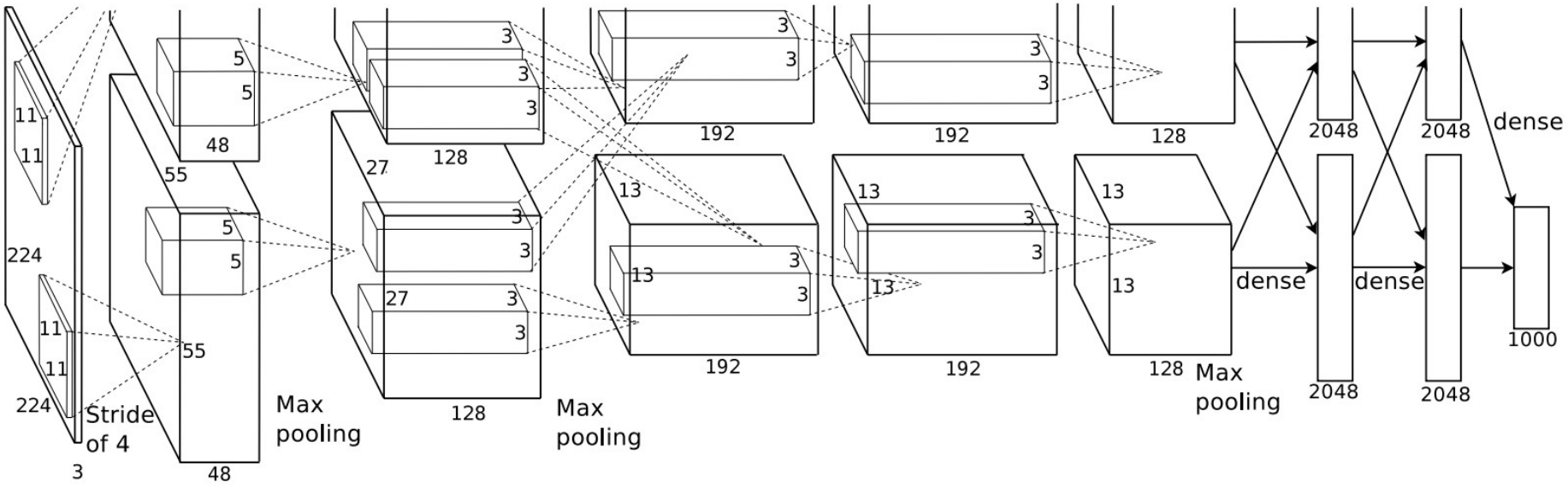
Convolutional Neural Network

- 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%



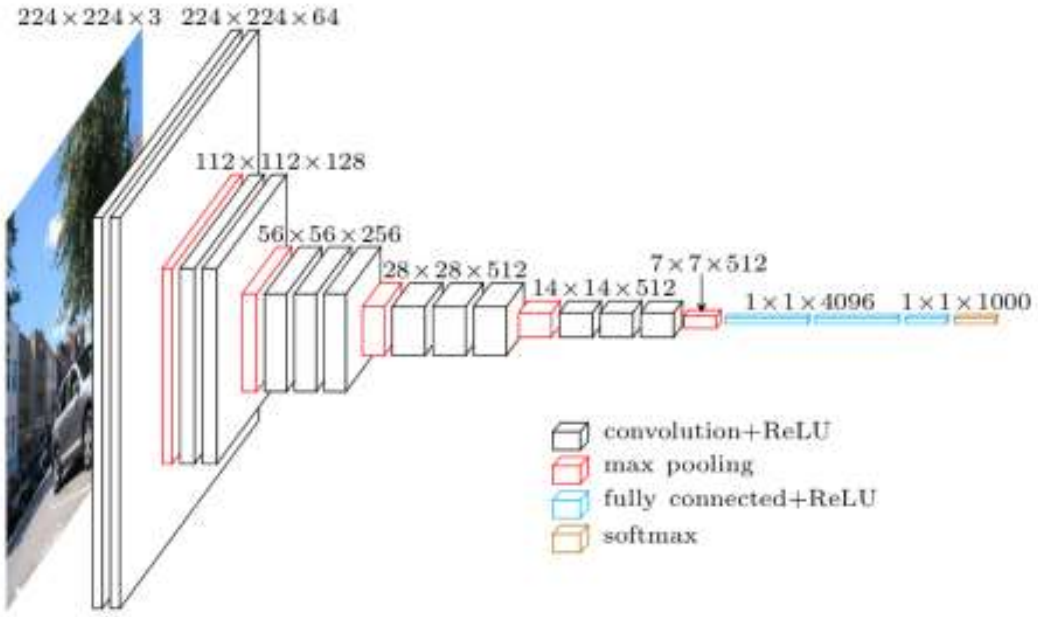
Convolutional Neural Network

- 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



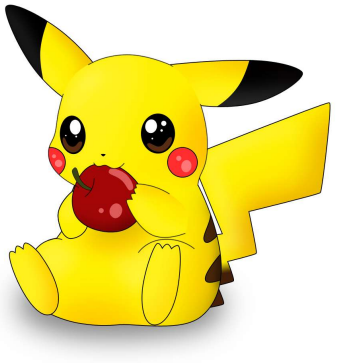
Convolutional Neural Network

- 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



Acitivities

- Activity 3 - **Classifying Fashion with CNN**
- Activity 4 - **Classifying Pokemon with CNN**



Pre-trained CNN models

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

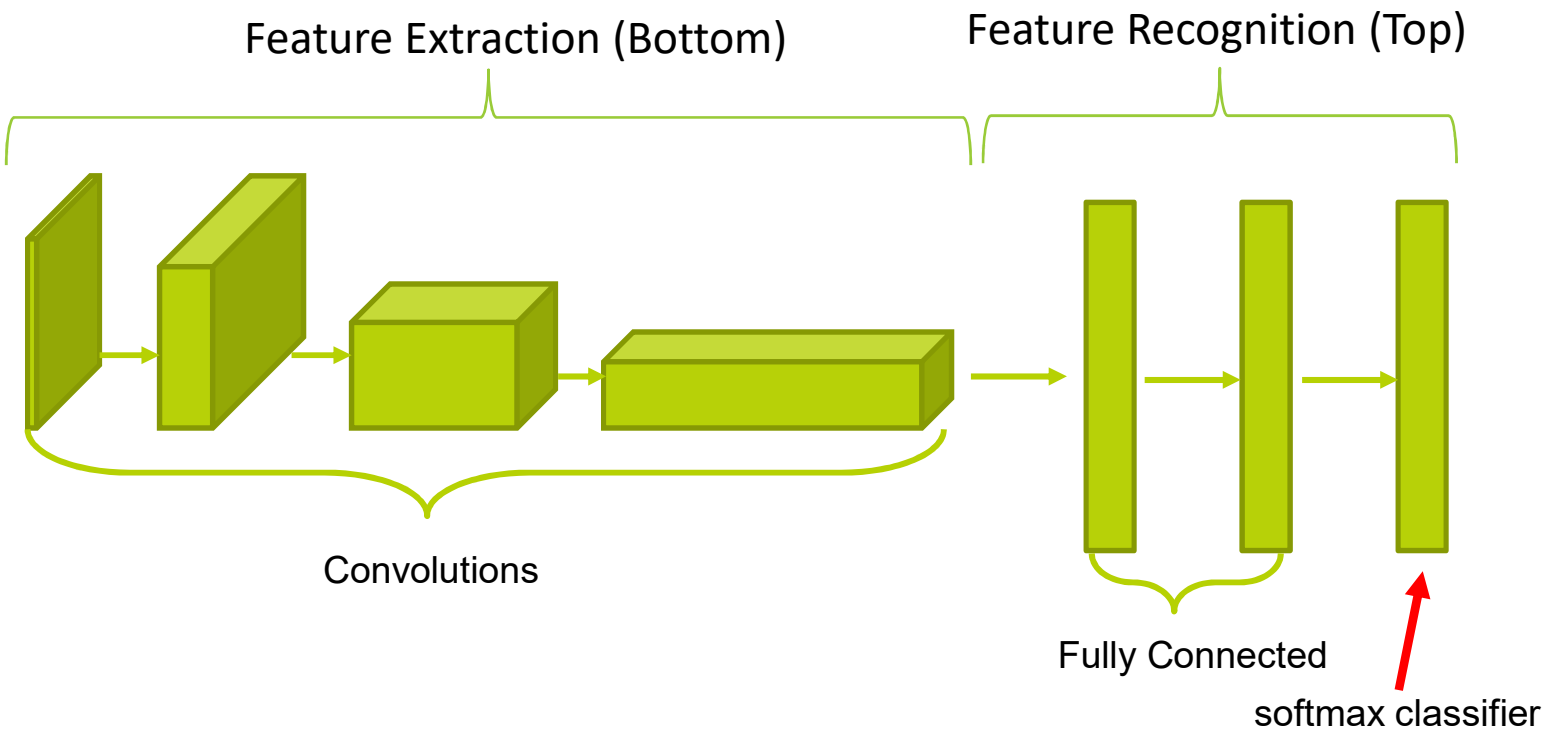
Transfer Learning

- 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

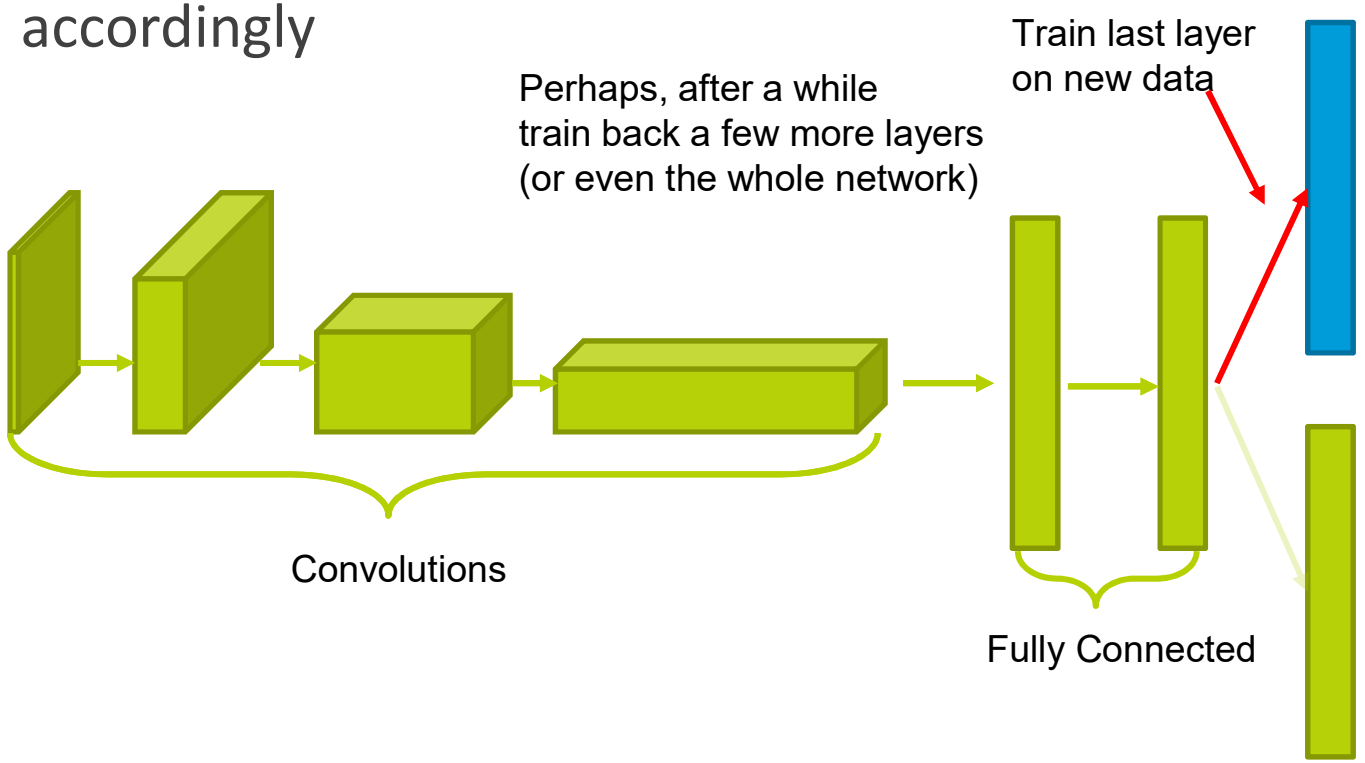
- Basic idea
  - Early layers in a Neural Network are the hardest (i.e. slowest) to train
  - Due to vanishing gradient property
  - But these ”primitive” features should be general across many image classification tasks
  - Later layers in the network are capturing features that are more particular 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.

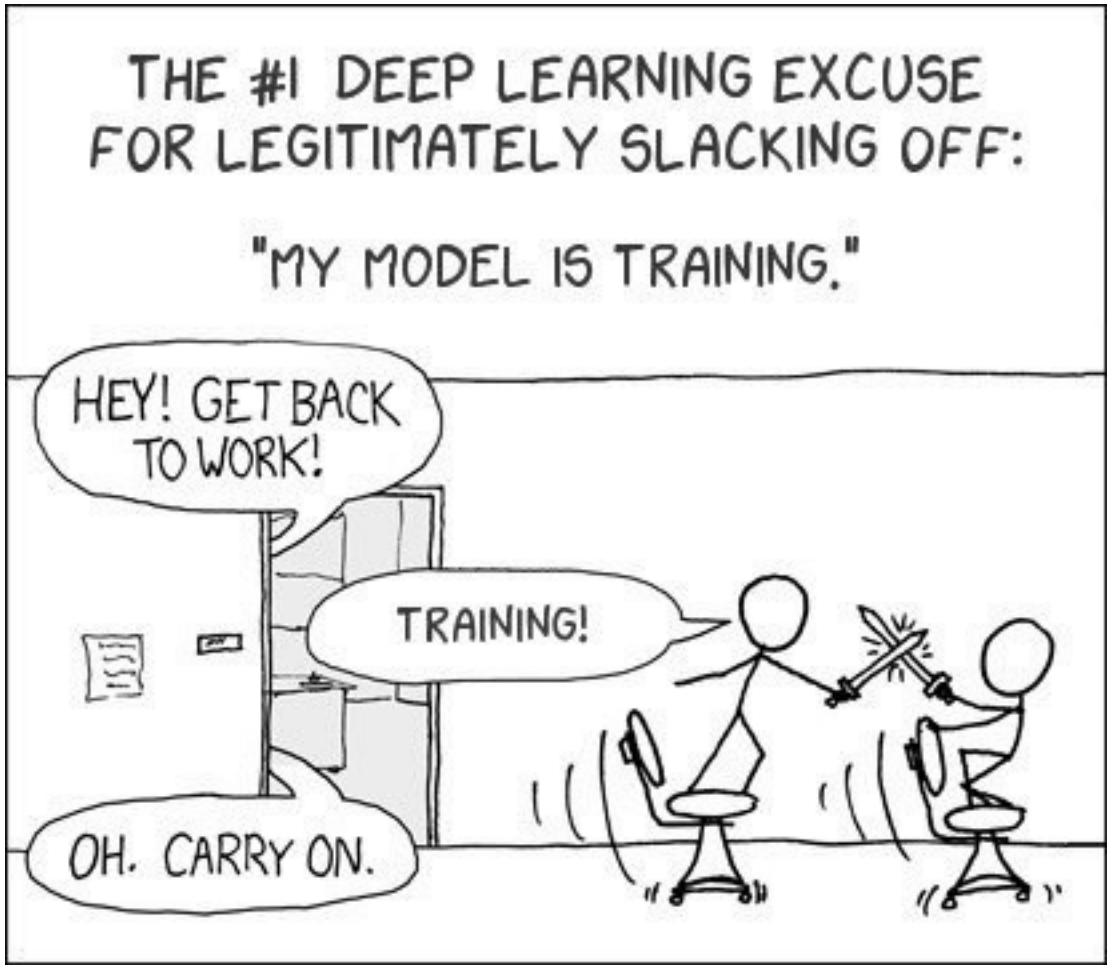
Transfer Learning

- 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.

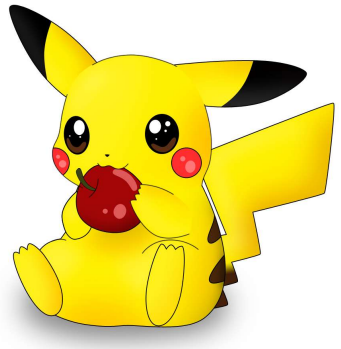
Training!


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

Acitivities

- Activity 5 - **Classifying Pokemon with pre-trained VGG16**

- Activity 6 - **Classifying Pokemon with transfer learning using pretrained VGG16**



Quiz

Thank you