AWS

MACHINE LEARNING

CERTIFICATION

DOMAIN #3: MODELING

MACHINE AND DEEP LEARNING BASICS – PART #2

AWS ML CERTIFICATION EXAM DOMAINS

Domain	% of Examination
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: Machine Learning Implementation and Operations	20%
TOTAL	100%

Source: https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty_Exam%20Guide%20(1).pdf

DOMAIN #3 OVERVIEW:

SECTION #8: MACHINE AND DEEP LEARNING BASICS – PART #1

- Artificial Neural Networks Basics: Single Neuron Model
- Activation Functions
- Multi-Layer Perceptron Model
- How do Artificial Neural Networks Train?
- ANN Parameters Tuning – Learning rate and batch size
- Tensorflow playground
- Gradient Descent and Backpropagation
- Overfitting and Under fitting
- How to overcome overfitting?
- Bias Variance Trade-off
- L1 Regularization
- L2 Regularization

SECTION #9: MACHINE AND DEEP LEARNING BASICS – PART #2

- Artificial Neural Networks Architectures
- Convolutional Neural Networks
- Recurrent Neural Networks
- Vanishing Gradient Problem
- LSTM Networks
- Model Performance Assessment – Confusion Matrix
- Model Performance Assessment – Precision, recall, F1-score
- Model Performance Assessment – ROC, AUC, Heatmap, and RMSE
- K-Fold Cross validation
- Transfer Learning
- Ensemble Learning – Bagging and Boosting

VALUABLE PRIZE!

- For those of you who will successfully complete the entire Module and watch the videos till the end, they will receive a valuable prize!

10 NEW SAMPLE EXAM QUESTIONS + COMPLETE ANSWER KEY

https://pixabay.com/illustrations/trophy-win-prize-transparent-1414791/

GAME AND MINI CHALLENGES!

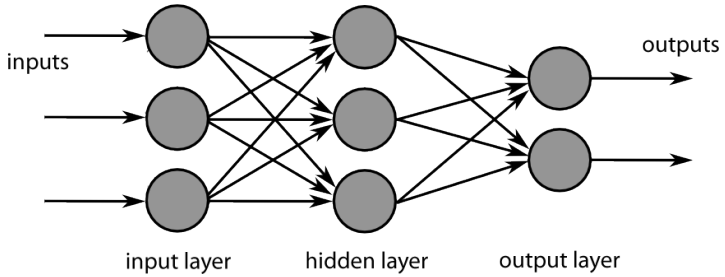
- Unfortunately, you can’t skip the videos.
- You have to collect a code throughout the lectures to unlock the exam.
- Special characters will appear at random moments throughout the video.
- You will need to collect the code and enter it to a website to access the material.
- That’s what the final code might look like!

ARTIFICIAL NEURAL NETWORKS

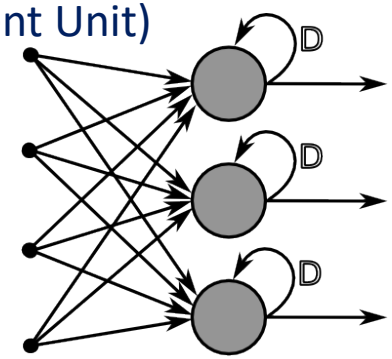
ARCHITECTURES

TYPES OF ARTIFICIAL NEURAL NETWORKS

- **Feedforward Neural Networks**
  - One to one mapping
  - Classification or regression
  - information flows from input layer directly through hidden layers then to the output layer without cycles/loops



- **Recurrent Neural Networks**
  - Considers temporal (time) information
  - Used for sequences prediction (future stock prices, translation, text.
  - Long short term memory (LSTM) and GRU (Gated Recurrent Unit)



- **Convolutional Neural Networks**
  - Image classification

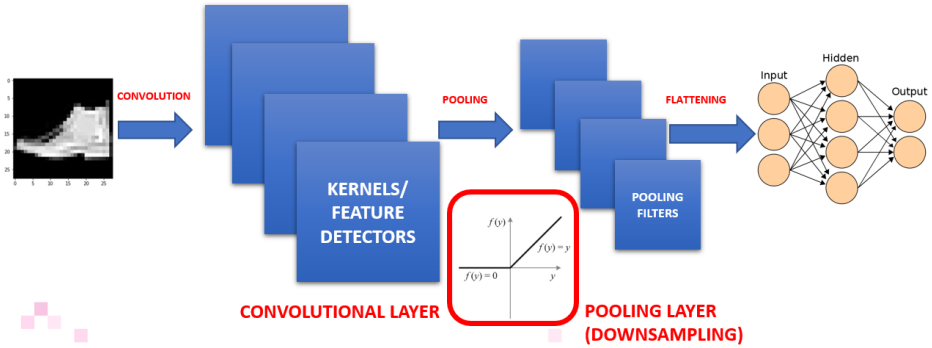


Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

DEEP LEARNING FRAMEWORKS

- **TensorFlow:**
  - Google’s end-to-end open source machine learning platform
  - https://www.tensorflow.org/

- **Keras:**
  - A high-level neural networks API that run on top of TensorFlow and Theano. Keras is super easy to use and is designed for fast implementation.
  - https://keras.io/

- **MXNet:**
  - Apache MXNet is an open-source, scalable deep learning framework to train and deploy deep neural networks.
  - https://mxnet.apache.org/

CONVOLUTIONAL NEURAL NETWORKS

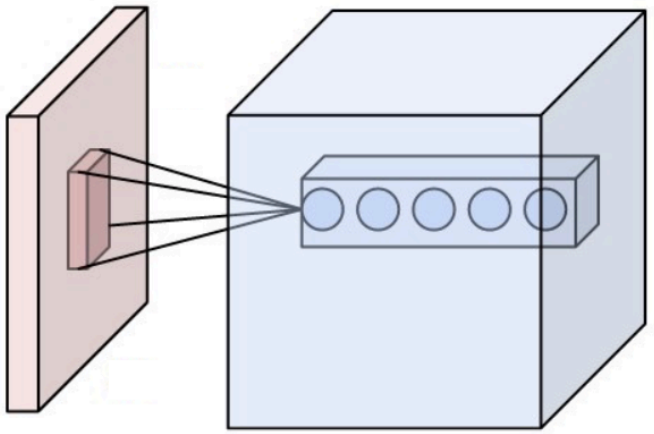
CONVOLUTIONAL NEURAL NETWORKS BASICS

- CNNs are used when data are not represented in column format.
- Here're some applications of CNNs:
  - Image classification
  - Sentiment analysis
  - Machine translation
- CNNs are "feature-location invariant" meaning that CNNs are used to detect and classify objects/features that are not in a specific location.

Photo Credit: https://www.pexels.com/photo/grey-and-white-short-fur-cat-104827/
Photo Credit: https://pixabay.com/photos/cat-kitty-pet-feline-animal-4269479/
Photo Credit: https://www.maxpixel.net/Cat-Fur-The-Cat-And-The-Dangerous-Tiger-Animal-4594189

CONVOLUTIONAL NEURAL NETWORKS BASICS

- CNNs are inspired by the visual cortex in the human brain
- CNNs work best with high-dimensional inputs such as images.
- It is generally not practical to connect all the neurons in a certain layer to all the neurons in the subsequent layer.
- Instead, each neuron is connected to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter named neuron receptive field (filter size).
- These receptive fields overlap to form a complete visual field.

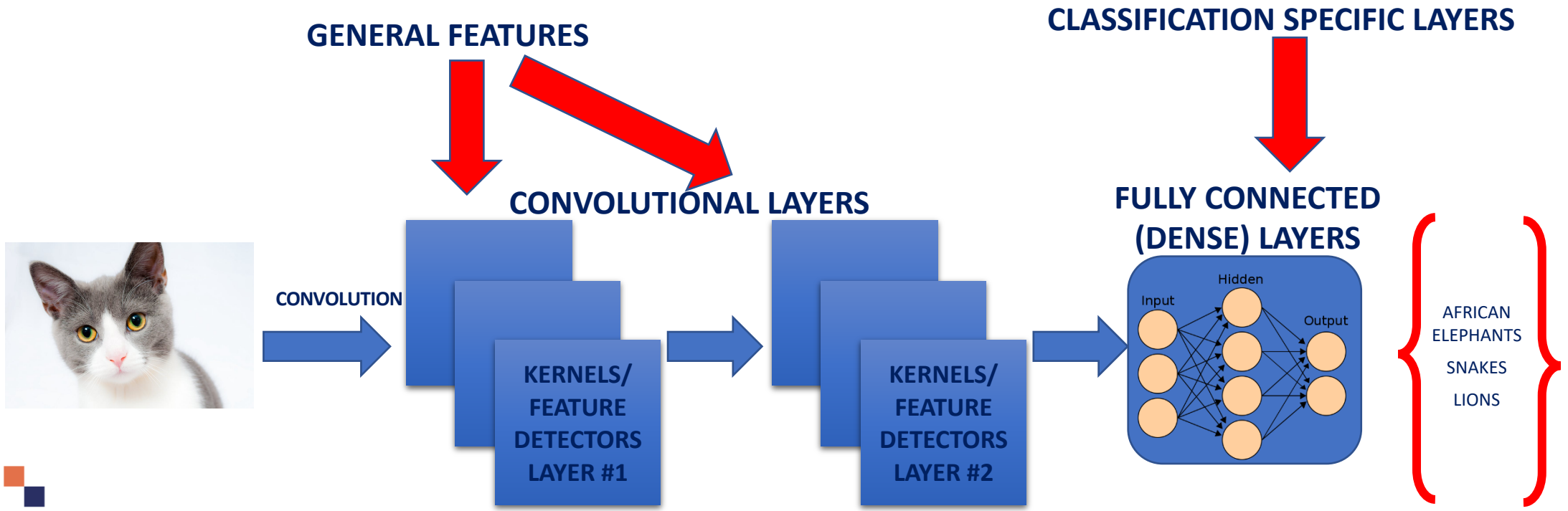


INPUT IMAGE

Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Conv_layer.png

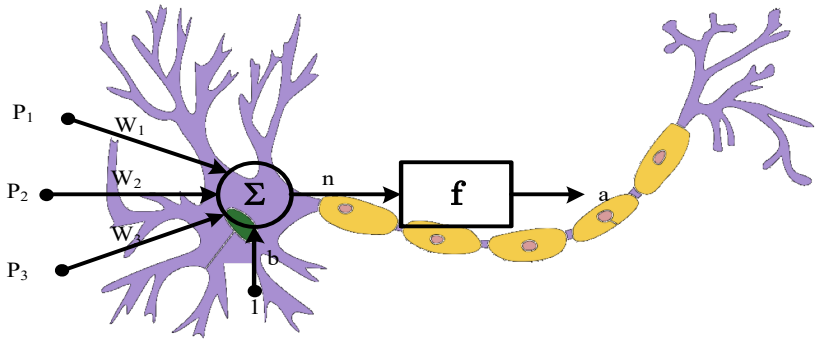
CNN LAYERS

- The first CNN layers are used to **extract high level general features.**
- The last couple of layers are used to perform classification (on a specific task).
- Local respective fields scan the image first searching for simple shapes such as edges/lines
- These edges are then picked up by the subsequent layer to form more complex features of the cat



CONVOLUTIONAL NEURAL NETWORKS BASICS

- The neuron collects signals from input channels named dendrites, processes information in its nucleus, and then generates an output in a long thin branch called the axon.
- Human learning occurs adaptively by varying the bond strength between these neurons.



$$n = P_1W_1 + P_2W_2 + P_3W_3 + b$$
$$a = f(n)$$

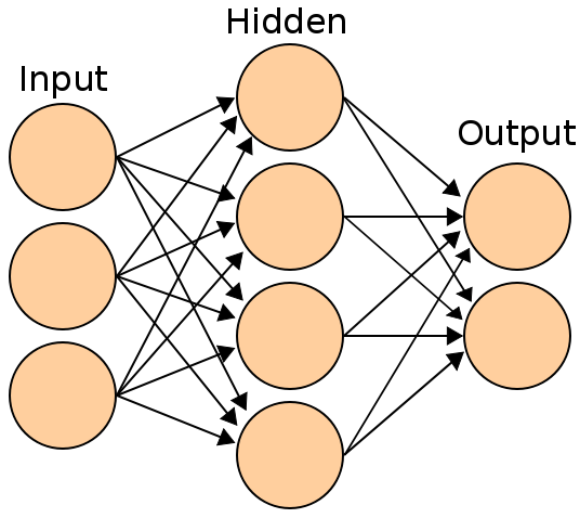


Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg
Photo Credit: https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg

CONVOLUTIONAL NEURAL NETWORKS: ENTIRE NETWORK OVERVIEW

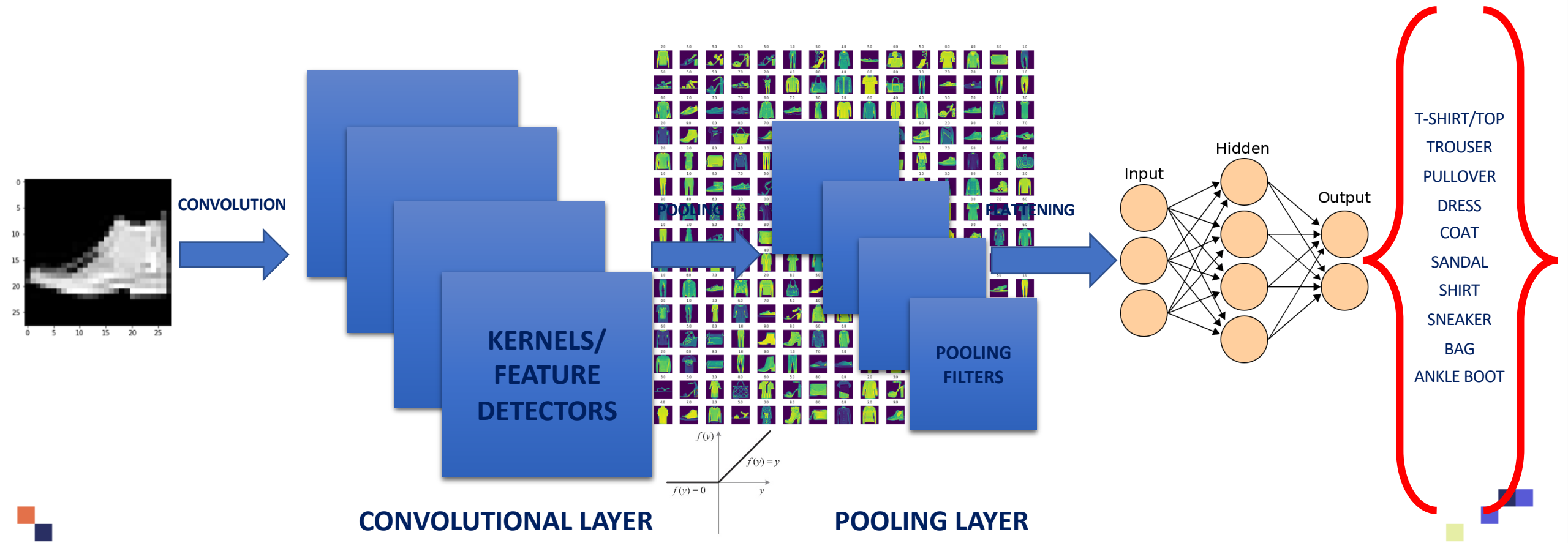
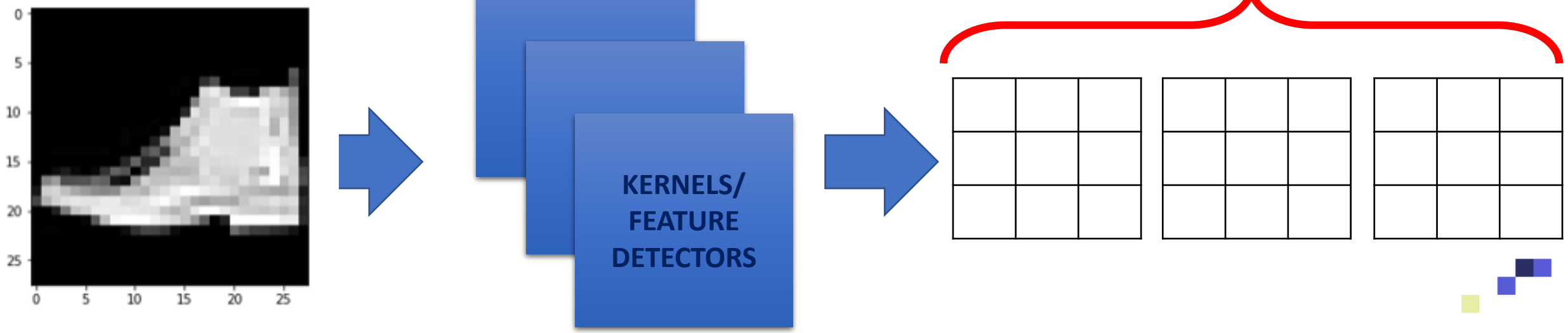


Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

FEATURE DETECTORS

- Convolutions use a kernel matrix to scan a given image and apply a filter to obtain a certain effect.
- An image Kernel is a matrix used to apply effects such as blurring and sharpening.
- Kernels are used in machine learning for feature extraction to select most important pixels of an image.
- Convolution preserves the spatial relationship between pixels.



FEATURE DETECTORS

- Live Convolution: http://setosa.io/ev/image-kernels/

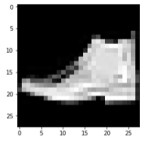
FEATURE DETECTOR

1	0	0
0	1	0
0	1	0

0	1	1	0	1
1	0	0	0	1
1	1	0	0	0
0	1	1	1	0
0	0	1	0	1

1	1	1
3	1	1
2	3	1

FEATURE MAP



IMAGE

RELU (RECTIFIED LINEAR UNITS)

- RELU Layers are used to add non-linearity in the feature map.
- It also enhances the sparsity or how scattered the feature map is.

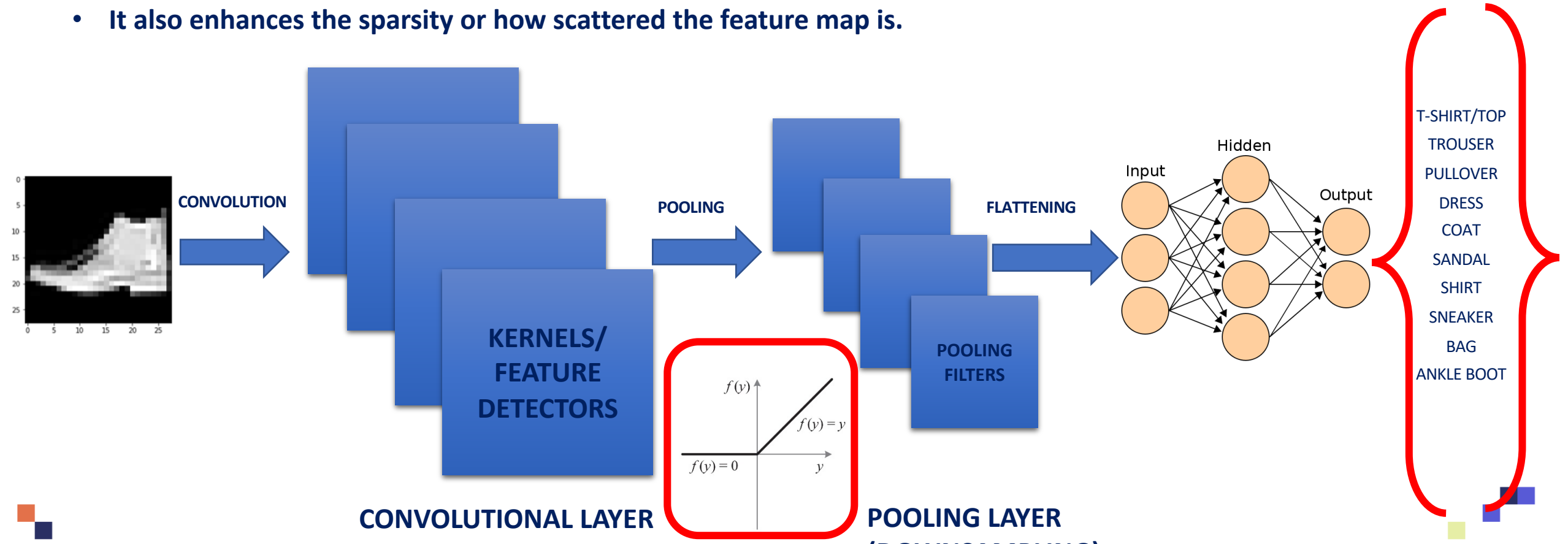
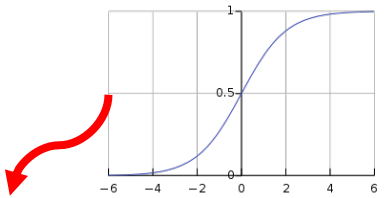


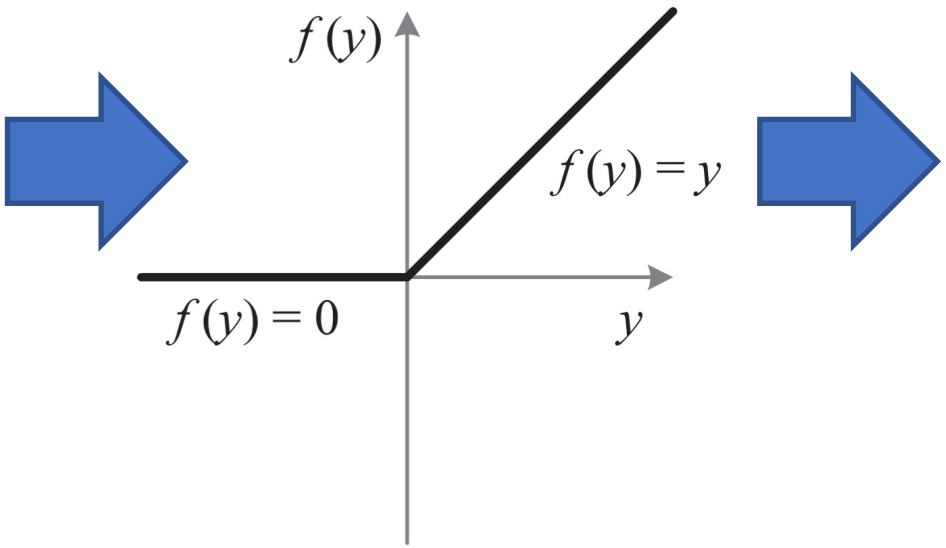
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

RELU (RECTIFIED LINEAR UNITS)

- RELU Layers are used to add non-linearity in the feature map.
- It also enhances the sparsity or how scattered the feature map is.
- The gradient of the RELU does not vanish as we increase x compared to the sigmoid function



7	10	-5	2	1
1	0	2	3	-6
1	17	-5	0	0
0	1	1	1	0
0	0	-8	12	1



7	10	0	2	1
1	0	2	3	0
1	17	0	0	0
0	1	1	1	0
0	0	0	12	1

Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

POOLING (DOWNSAMPLING)

- Pooling or down sampling layers are placed after convolutional layers to reduce feature map dimensionality.
- This improves the computational efficiency while preserving the features.
- Pooling helps the model to generalize by avoiding overfitting.
- If one of the pixel is shifted, the pooled feature map will still be the same.
- Max pooling works by retaining the maximum feature response within a given sample size in a feature map.
- Live illustration : http://scs.ryerson.ca/~aharley/vis/conv/flat.html

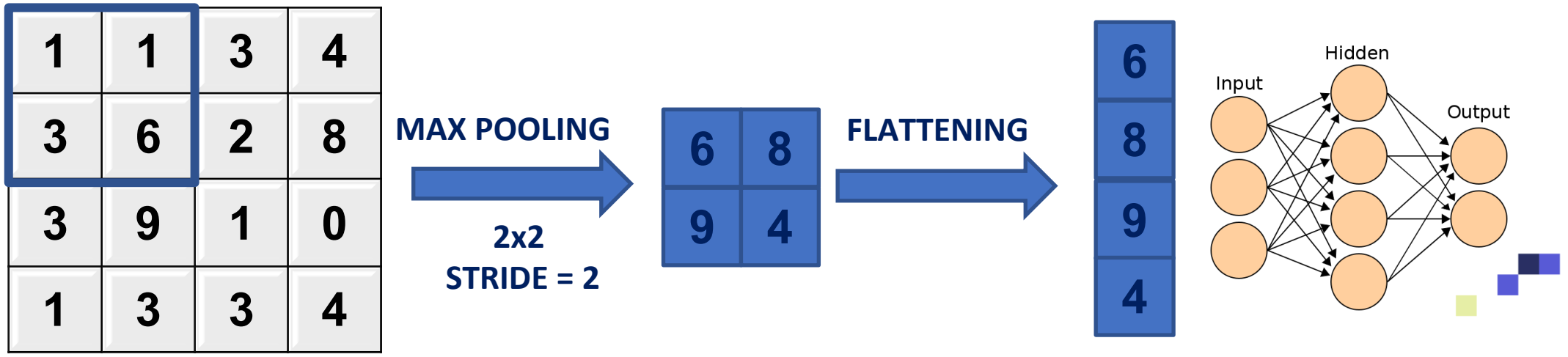


Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

CNN IN KERAS (TENSORFLOW 2.0)

- Let's build a simple CNN in Keras

```
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

# Reshape training data to be = (60000, 28, 28, 1) instead of (60000, 28,28)
train_images = train_images.reshape(60000, 28, 28, 1)
test_images = test_images.reshape(10000, 28, 28, 1)

from tensorflow.keras import datasets, layers, models
model = models.Sequential()

model.add(layers.Conv2D(6, (5,5), activation = 'relu', input_shape = (28,28,1)))
model.add(layers.AveragePooling2D())

model.add(layers.Conv2D(16, (5,5), activation = 'relu'))
model.add(layers.AveragePooling2D())

model.add(layers.Flatten())
model.add(layers.Dense(120, activation = 'relu'))
model.add(layers.Dense(84, activation = 'relu'))
model.add(layers.Dense(10, activation = 'softmax'))
model.summary()

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)
```

CNN SUMMARY

- CNNs are extremely computational expensive requiring GPUs to train
- CNNs include several hyperparameters such as kernel sizes, number of filters, pooling layers, number of neurons in the dense layers.
- The following architecture is the most common:
    1. Conv2D – performs the convolution on a 2D image
    2. MaxPooling2D – downsampling
    3. Dropout – regularization technique
    4. Flatten – convert 2D layer to a 1D to be fed to Dense (fully connected) network
    5. Dense – fully connected ANN
    6. Softmax

- CNN work best with images of the following dimensions:
    - width x length x color channels
    - Example: 32 x 32 x 3

ADVANCED OFF THE SHELVE CNNs

- There are many trained off the shelve convolutional neural networks that are readily available such as:
  - LeNet-5 (1998): 7 level convolutional neural network developed by LeCun that works in classifying hand writing numbers.
  - AlexNet (2012): Offered massive improvement, error reduction from 26% to 15.3%
  - ZFNEt (2013): achieved error of 14.8%
  - Googlenet/Inception (2014): error reduction to 6.67% which is at par with human level accuracy.
  - VGGNet (2014)
  - ResNet (2015): Residual Neural Network includes “skip connection” feature and therefore enabled training of 152 layers without vanishing gradient issues. Error of 3.57% which is superior than humans.

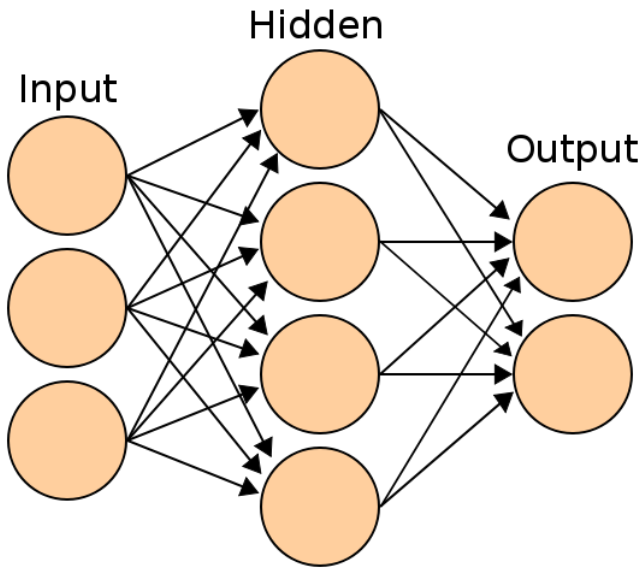
https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

RECURRENT NEURAL NETWORKS INTUITION

RECURRENT NEURAL NETWORKS (RNN): WHAT ARE THEY?

- We covered Feedforward Neural Networks (vanilla networks) that map a fixed size input (such as image) to a fixed size output (classes or probabilities).
- A drawback in Feedforward networks is that they do not have any time dependency or memory effect.
- A RNN is a type of ANN that is designed to take temporal dimension into consideration by having a memory (internal state) (feedback loop).

FEED FORWARD ANN



RECURRENT NEURAL NETWORK

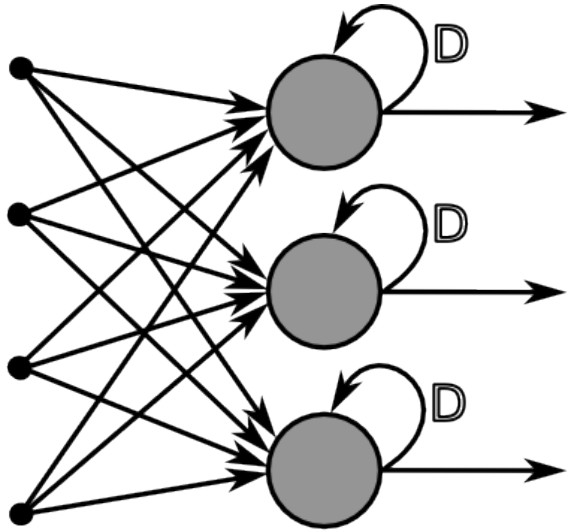


Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

RNN ARCHITECTURE

- A RNN contains a temporal loop in which the hidden layer not only gives an output but it feeds itself as well.
- An extra dimension is added which is time!
- RNN can recall what happened in the previous time stamp so it works great with sequence of text.

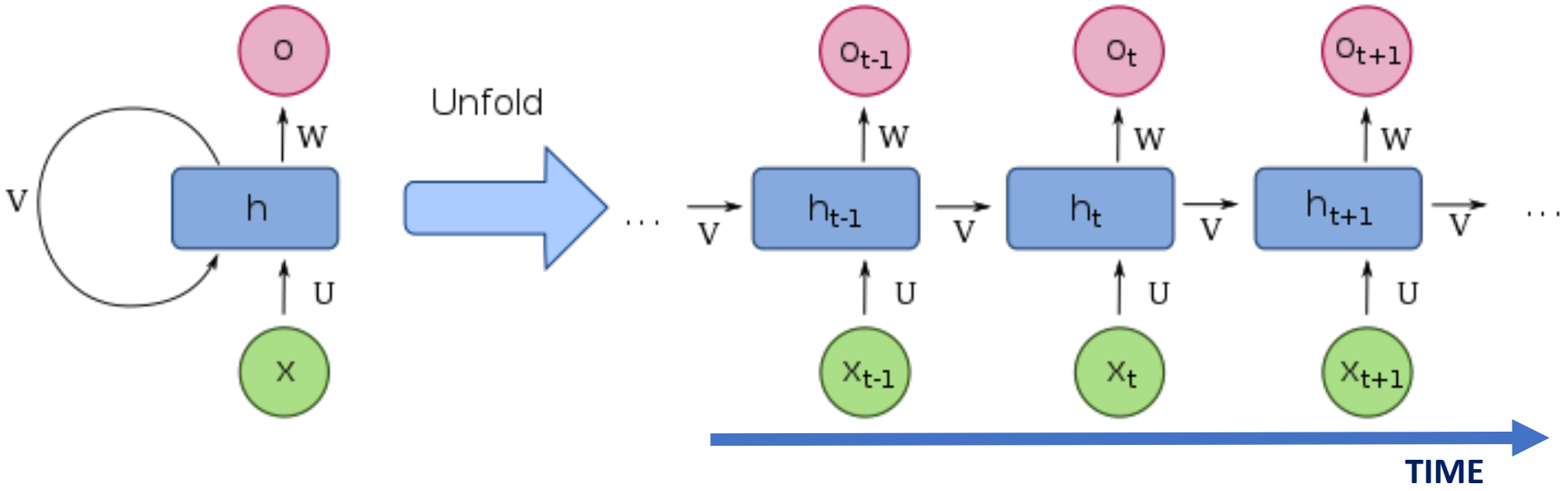


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

RNNs WORK LIKE MAGIC!

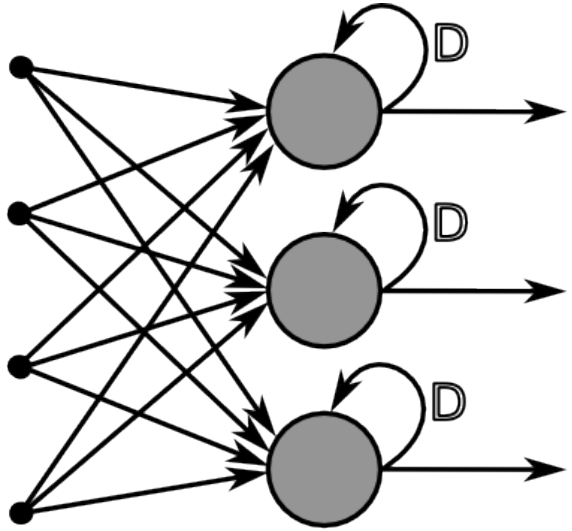
We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"

Source: The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

WHAT MAKES RNNs SO SPECIAL?

- Feedforward ANNs are so constrained with their fixed number of input and outputs.
- For example, a CNN will have fixed size image (28x28) and generates a fixed output (class or probabilities).
- Feedforward ANN have a fixed configuration, i.e.: same number of hidden layers and weights.
- Recurrent Neural Networks offer huge advantage over feedforward ANN and they are much more fun!
- RNN allow us to work with a sequence of vectors:
  - Sequence in inputs
  - Sequence in outputs
  - Sequence in both!

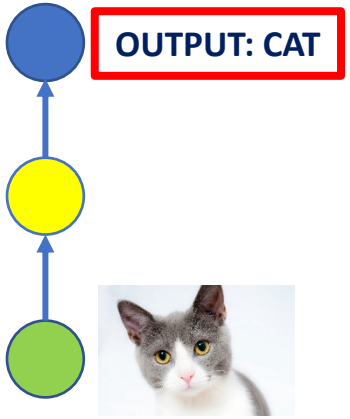
RECURRENT NEURAL NETWORK



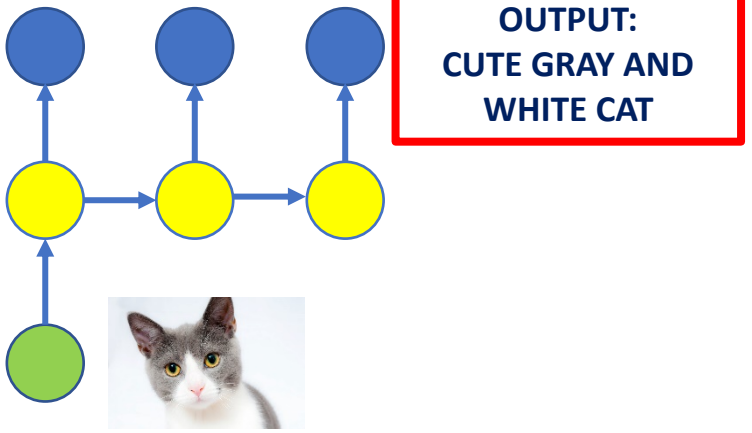
Source: The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png

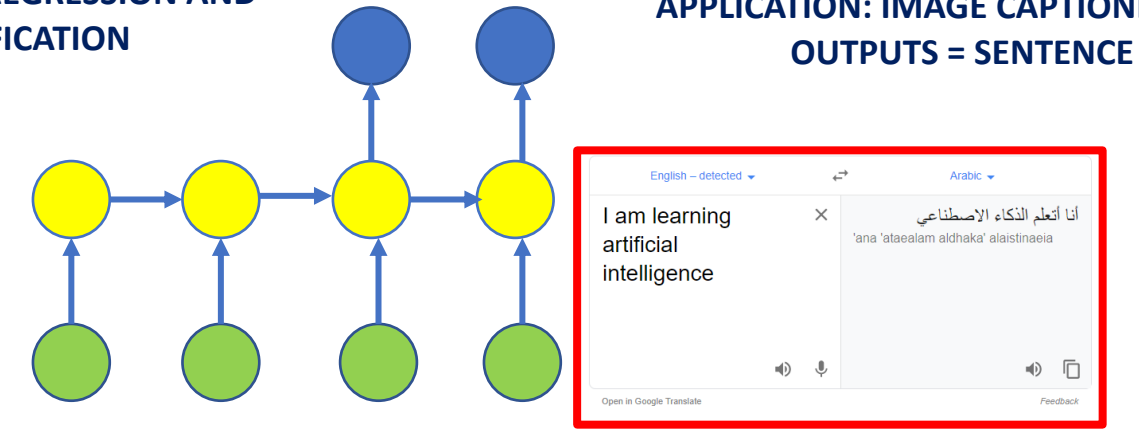
WHAT MAKES RNNs SO SPECIAL?



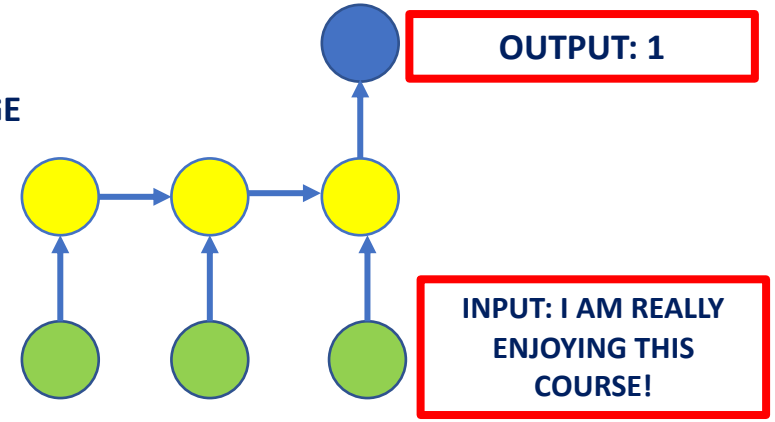
ONE TO ONE (VANILLA)
APPLICATION: REGRESSION AND CLASSIFICATION



ONE TO MANY (SEQUENCE OUTPUT)
APPLICATION: IMAGE CAPTIONING, INPUT = IMAGE OUTPUTS = SENTENCE OF WORDS)



MANY TO MANY (SEQUENCE INPUT AND OUTPUT)
APPLICATION: LANGUAGE TRANSLATION,



MANY TO ONE (SEQUENCE INPUT)
APPLICATION: SENTIMENT ANALYSIS,
EX: IS A REVIEW POSITIVE OR NEGATIVE?

Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

RNN MATH

- A RNN accepts an input x and generate an output o.
- The output o does not depend on the input x alone, however, it depends on the entire history of the inputs that have been fed to the network in previous time steps.
- Two equations that govern the RNN are as follows:

  - **INTERNAL STATE UPDATE:**

$$h_t = \tanh(X_t * U + h_{t-1} * V)$$

  - **OUTPUT UPDATE:**

$$o_t = softmax(W * h_t)$$

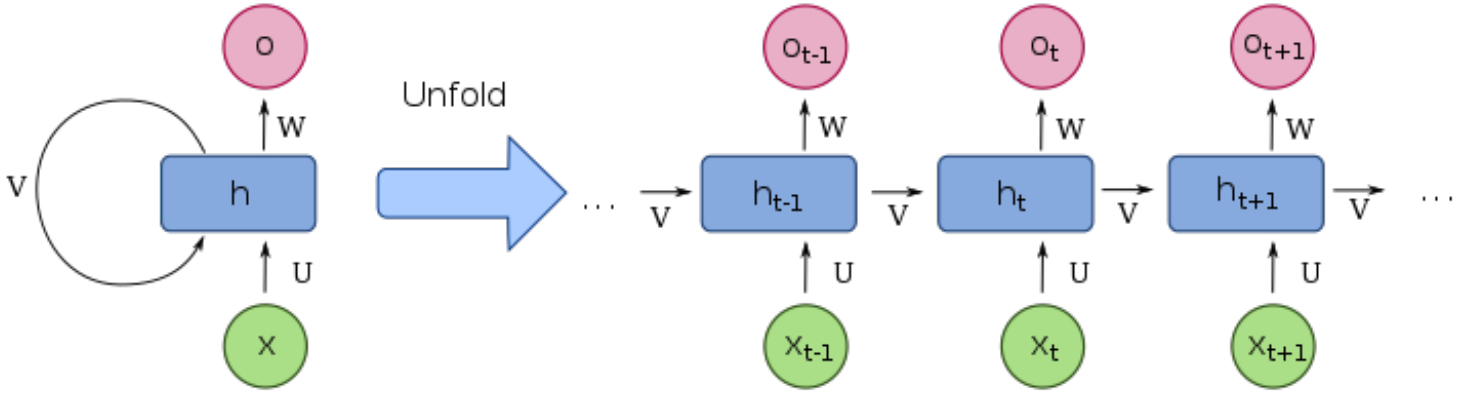


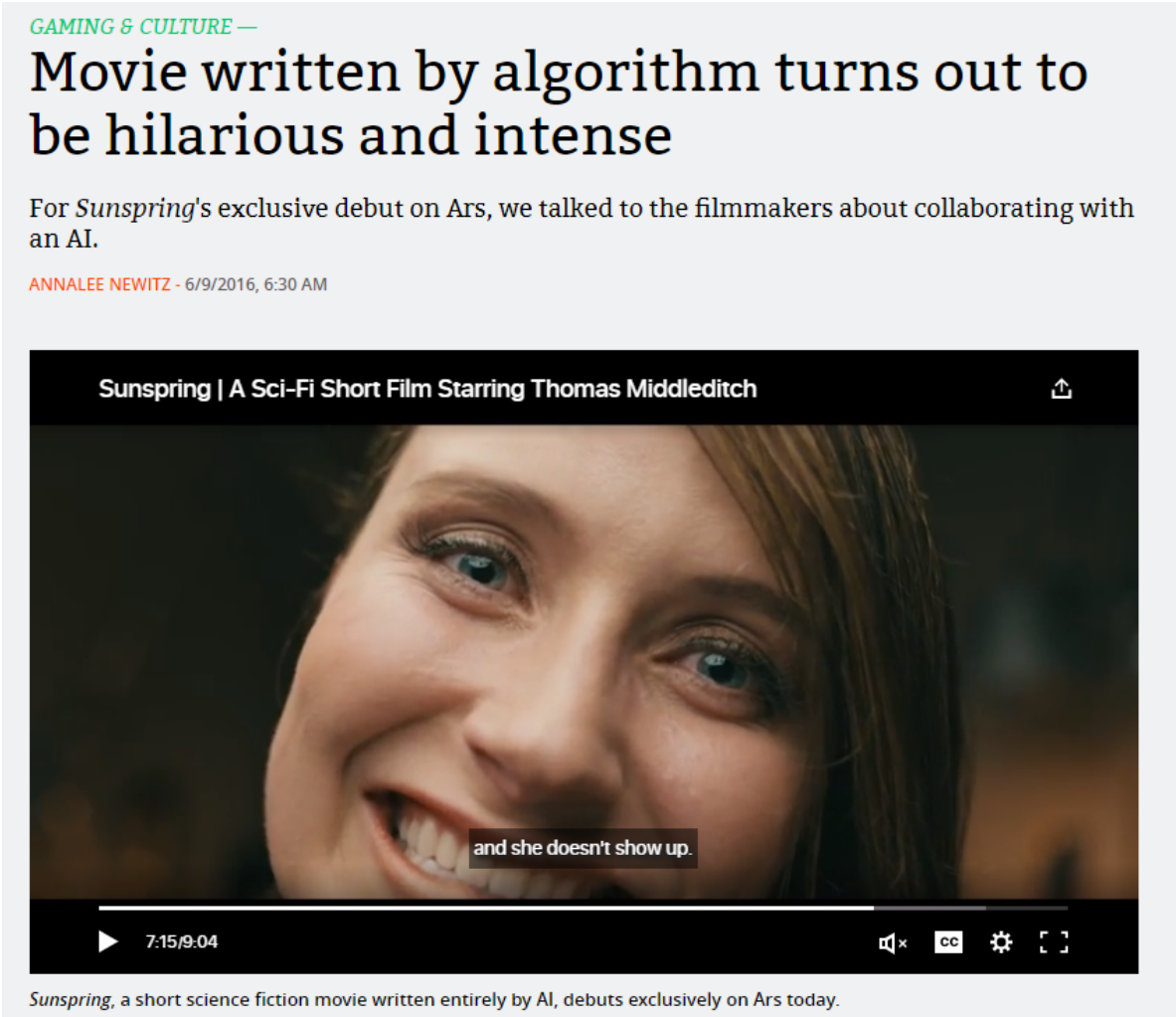
Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

LET'S WATCH THIS MOVIE WRITTEN BY AN RNN!

- Let's watch a movie written by AI!

https://arstechnica.com/gaming/2016/06/an-ai-wrote-this-movie-and-its-strangely-moving/

- The movie was written by an LSTM recurrent neural network
- The LSTM network was trained with a corpus of dozens of sci-fi screenplays from movies from the 1980s and 90s.



GAMING & CULTURE —

Movie written by algorithm turns out to be hilarious and intense

For *Sunspring*'s exclusive debut on Ars, we talked to the filmmakers about collaborating with an AI.

ANNALEE NEWITZ - 6/9/2016, 6:30 AM

Sunspring, a short science fiction movie written entirely by AI, debuts exclusively on Ars today.

Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

VANISHING GRADIENT PROBLEM

VANISHING GRADIENT PROBLEM

- LSTM networks work much better compared to vanilla RNN since they overcome the vanishing gradient problem.
- The error has to propagate through all the previous layers resulting in a vanishing gradient.
- As the gradient goes smaller, the network weights are no longer updated.
- As more layers are added, the gradients of the loss function approaches zero, making the network hard to train.

EACH LAYER DEPENDS ON THE OUTPUT FROM THE PREVIOUS LAYERS, THE "V" IS MULTIPLIED SEVERAL TIMES RESULTING IN VANISHING GRADIENT

0.1 * 0.1 * 0.1 *…..*0.1 =1e-10

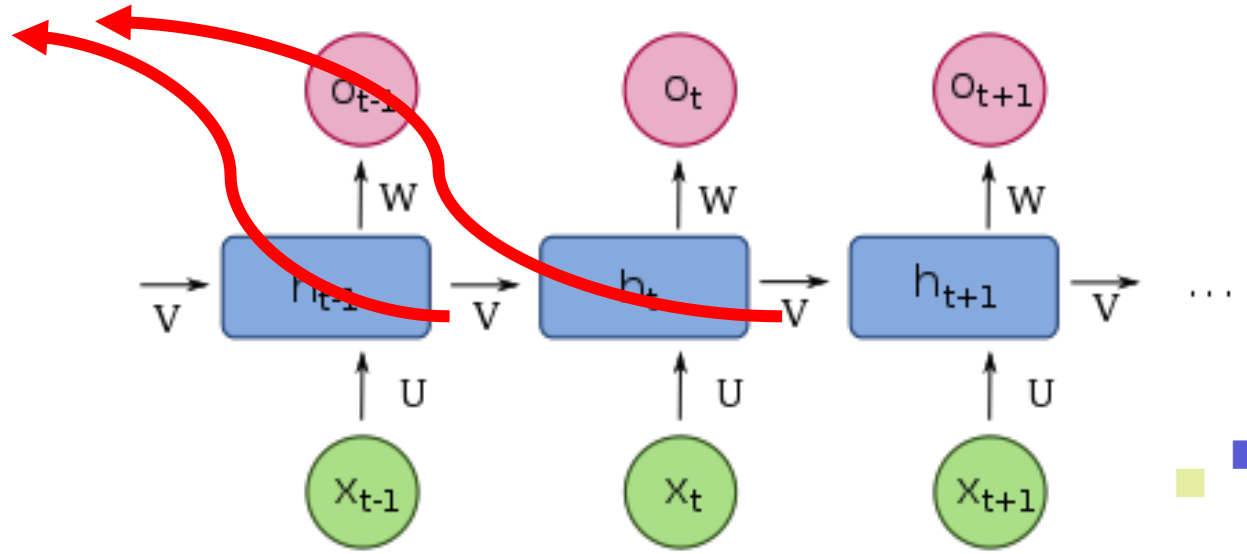
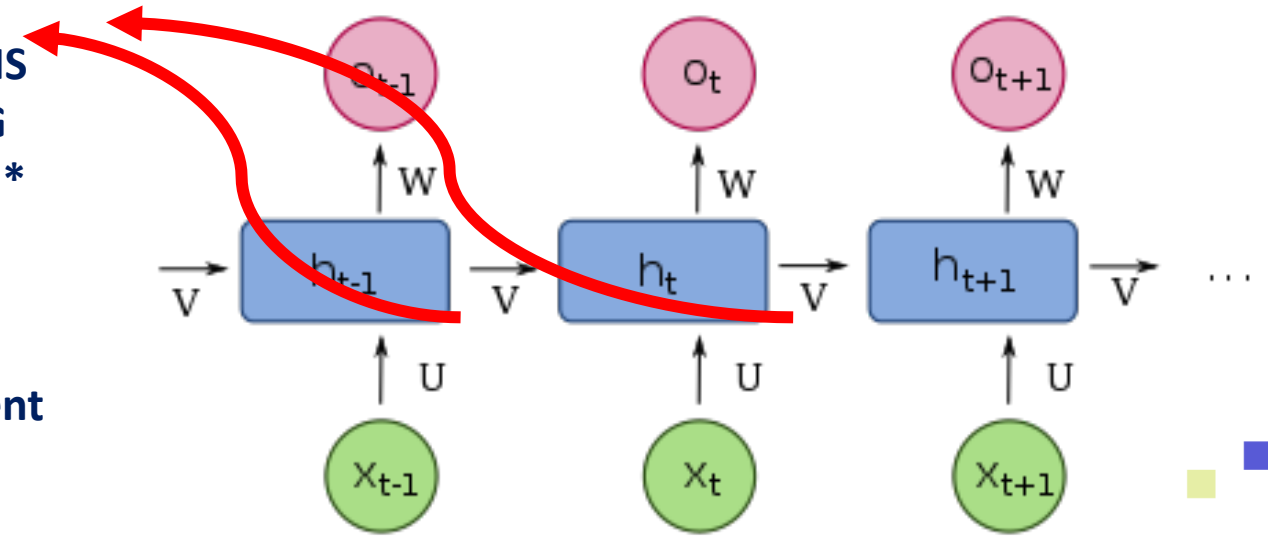


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

VANISHING GRADIENT PROBLEM

- ANN gradients are calculated during backpropagation.
- In backpropagation, we calculate the derivatives of the network by moving from the outermost layer (close to output) back to the initial layers (close to inputs).
- The chain rule is used during this calculation in which the derivatives from the final layers are multiplied by the derivatives from early layers.
- The gradients keeps diminishing exponentially and therefore the weights and biases are no longer being updated.

EACH LAYER DEPENDS ON THE OUTPUT FROM THE PREVIOUS LAYERS, THE "V" IS MULTIPLIED SEVERAL TIMES RESULTING IN VANISHING GRADIENT, (ex: 0.1 * 0.1 * 0.1 *…..*0.1 =1e-10)



New Weight = Old Weight – Learning rate * gradient
9.09999 = 10.1 – 1 * 0.001

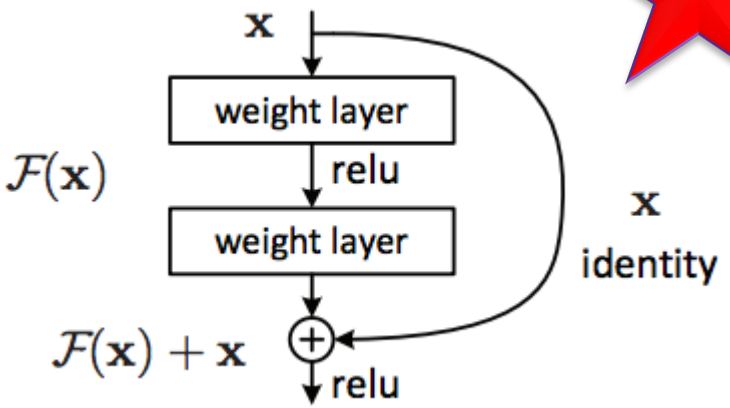
Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

VANISHING GRADIENT PROBLEM SOLUTION

- **Choose Proper Activation Function:** Use RELU activation function instead of Tanh or sigmoid activation functions
- **Use Long short term memory networks (LSTM):** LSTM could be used instead of pure vanilla Recurrent neural networks
- **Use ResNet (residual networks)**
- **Multi-level hierarchy:** multi-level hierarchy of networks pre-trained one level at a time through unsupervised learning, fine-tuned through backpropagation.

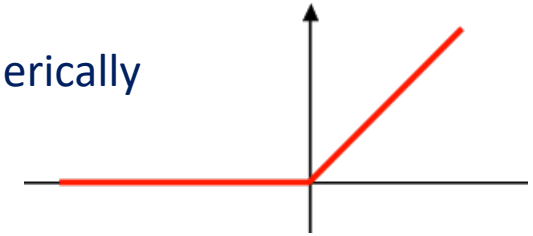
- **Gradient Checking:** a debugging strategy used to numerically track and assess gradients during training

RESNET



RELU

$$\mathrm{ReLU}(x) \triangleq \max(0, x)$$



LSTM

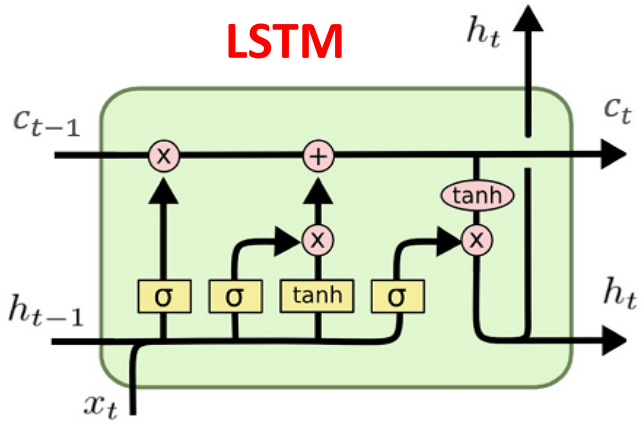


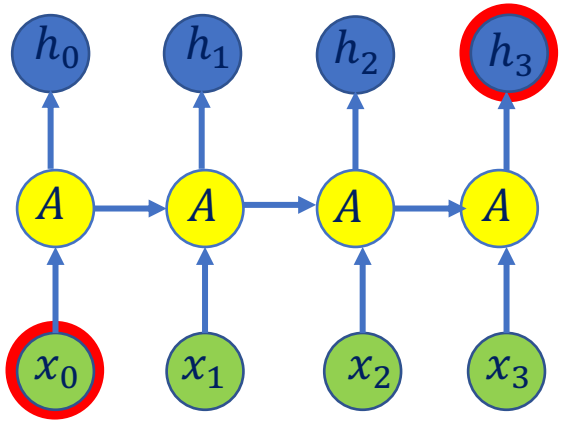
Photo Credit: https://commons.wikimedia.org/wiki/File:Resnet.png
Reference and Photo Credit: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Photo Credit: https://commons.wikimedia.org/wiki/File:LSTM.png

LONG SHORT TERM MEMORY (LSTM) NETWORKS

LSTM INTUITION

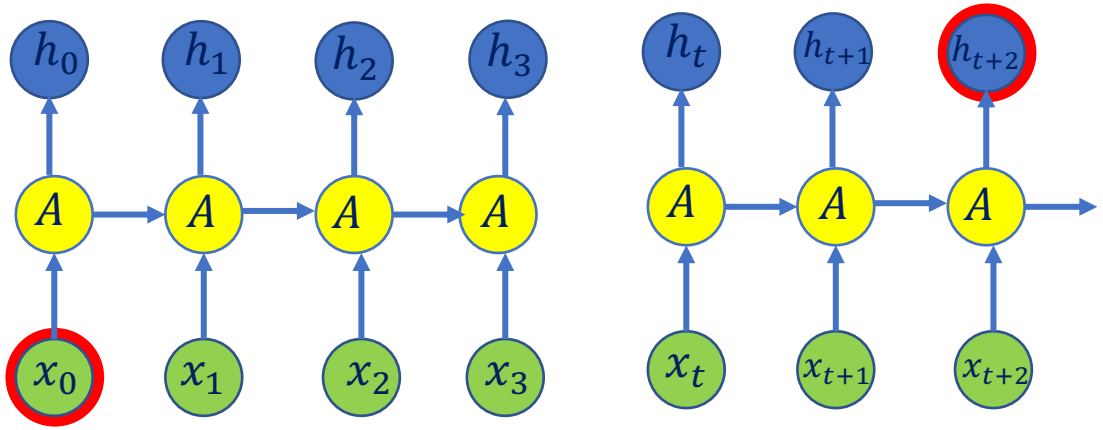
- LSTMs work better compared to vanilla RNN since they overcome vanishing gradient problem.
- In practice, RNN fail to establish long term dependencies.
- Reference: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The tree color is "green"



RNN PERFORMS WELL SINCE THE GAP BETWEEN THE PREDICTION "GREEN" AND THE NECESSARY CONTEXT INFORMATION "TREE" IS SMALL

I live in Quebec in Northern Canada……where I live, the weather is generally "cold" most of the year



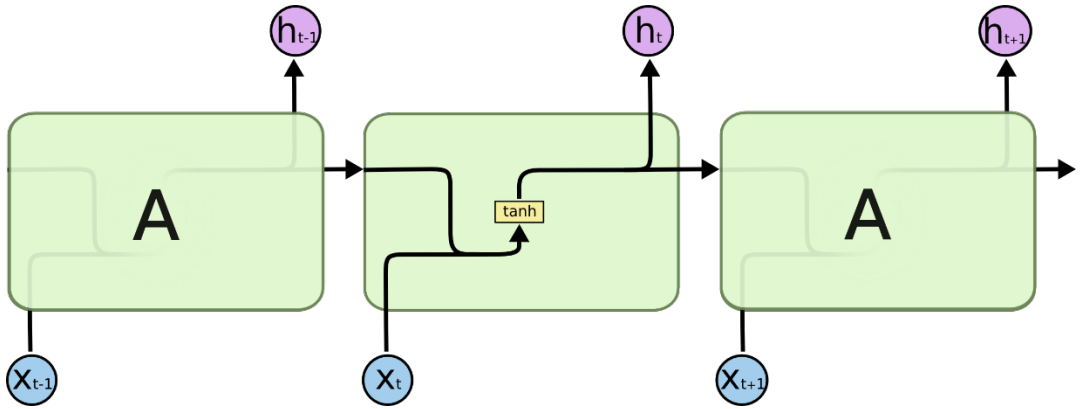
RNN PERFORMS POORLY WHEN THE GAP BETWEEN THE PREDICTION "COLD" AND THE NECESSARY CONTEXT INFORMATION "CANADA" IS LARGE

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

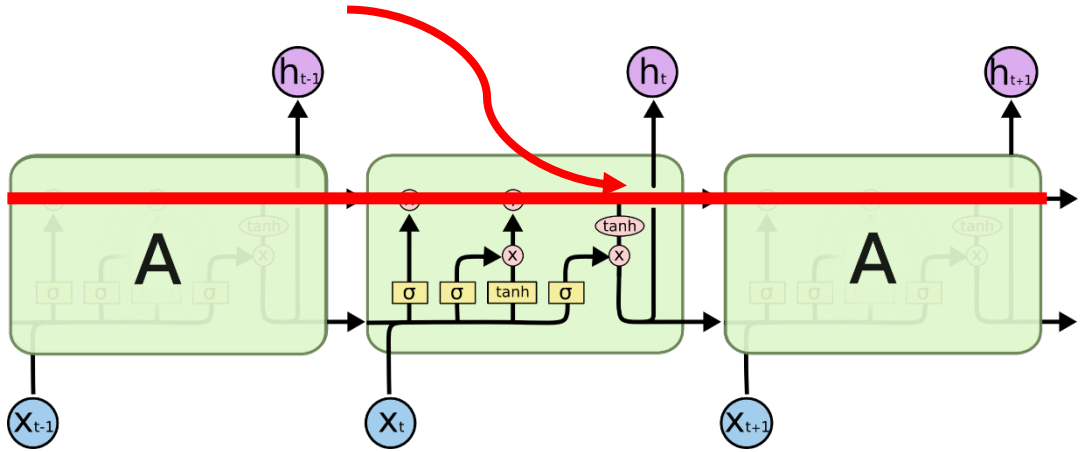
LSTM INTUITION

- LSTM networks are type of RNN that are designed to remember long term dependencies by default.
- LSTM can remember and recall information for a prolonged period of time.
- Recall that each line represents a full vector.

THIS HORIZONTAL LINE (MEMORY) OR CELL STATE ENABLES LSTM TO REMEMBER VERY OLD INFORMTION



VANILLA RECURRENT NEURAL NETWORK

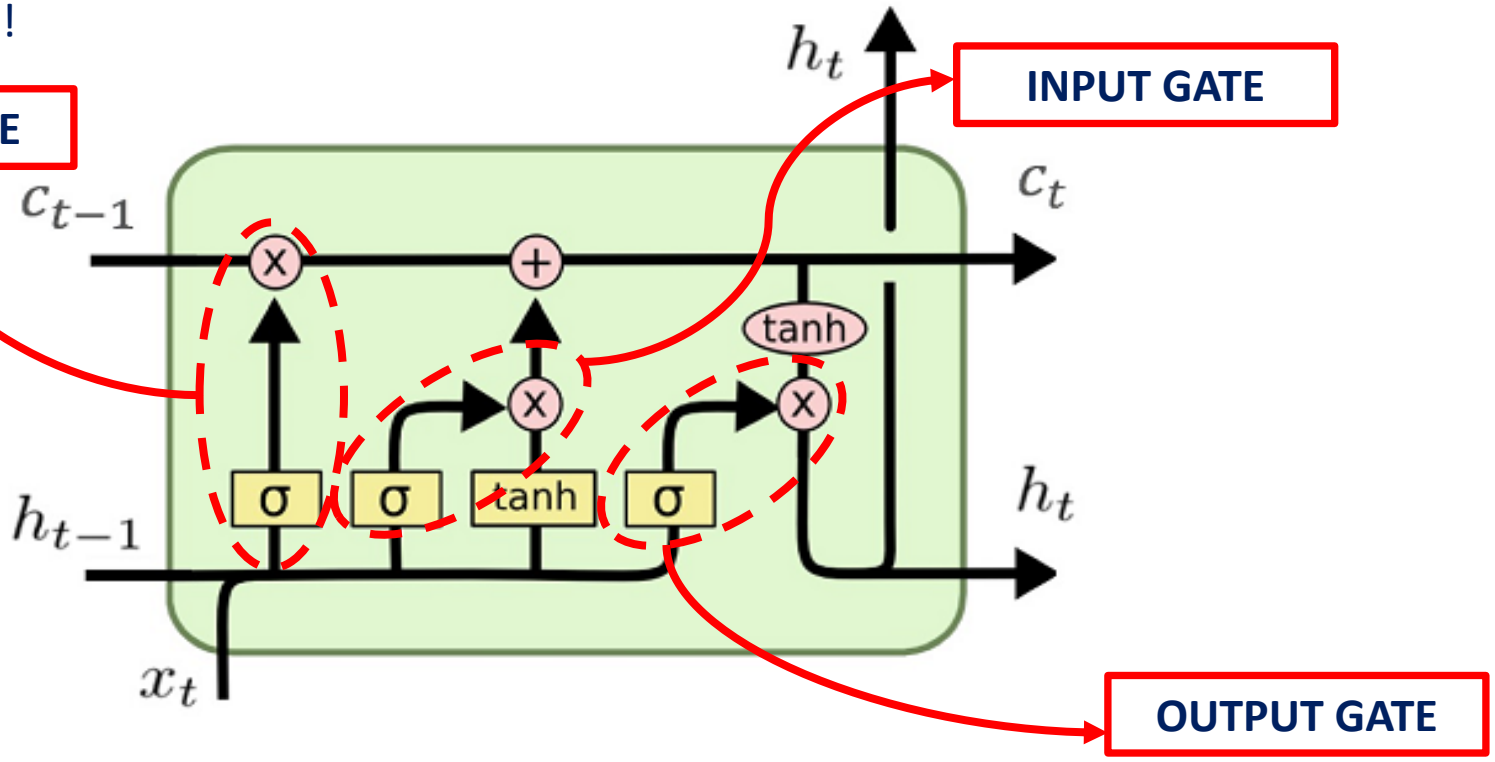


LONG SHORT TERM MEMORY NETWORK

Reference and Photo Credit:
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM INTUITION – GATES

- LSTM contains gates that can allow or block information from passing by.
- Gates consist of a sigmoid neural net layer along with a pointwise multiplication operation.
- Sigmoid output ranges from 0 to 1:
  - 0 = Don't allow any data to flow
  - 1 = Allow everything to flow!



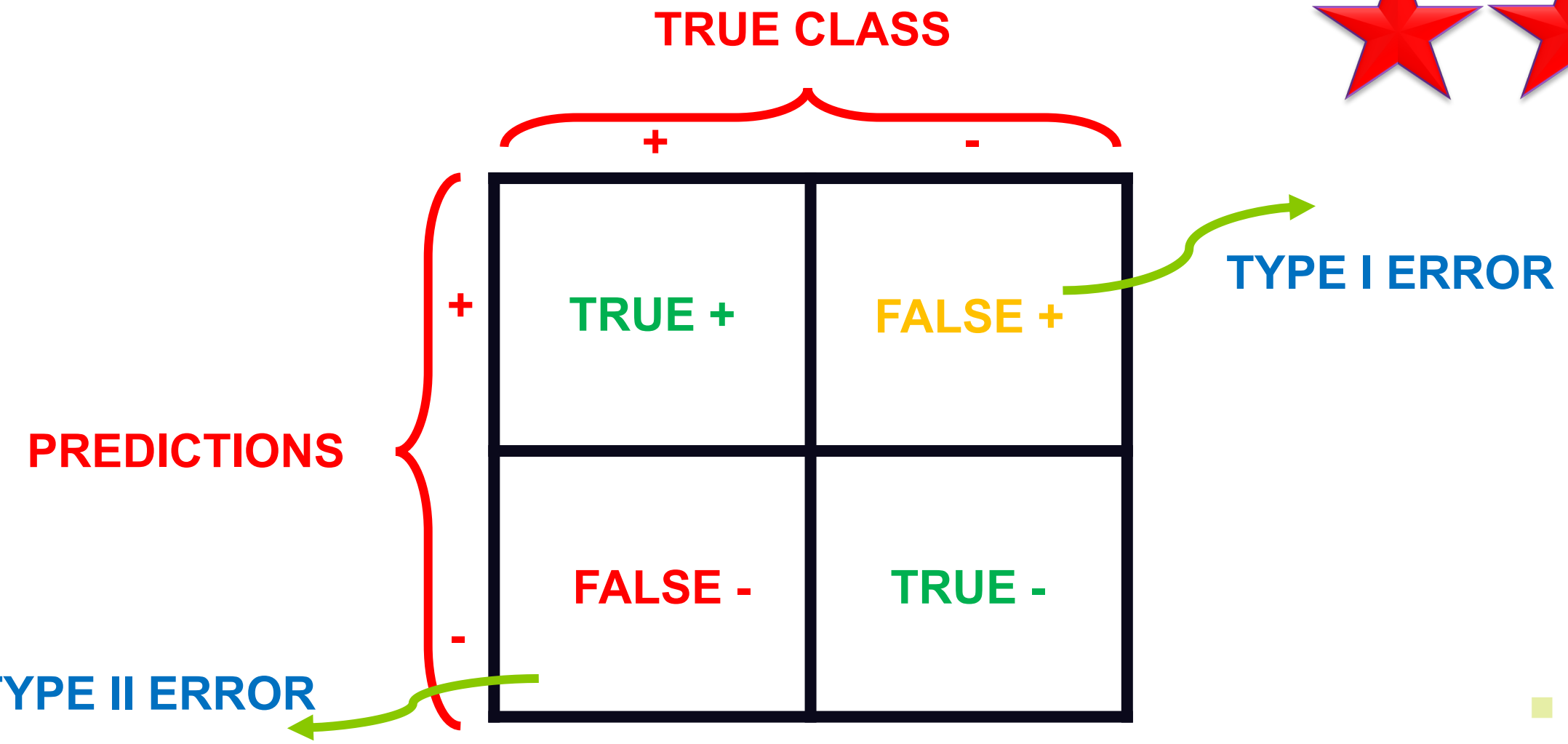
Reference and Photo Credit: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Photo Credit: https://commons.wikimedia.org/wiki/File:LSTM.png

RNN Training

- RNNs are trained using Backpropagation through time
- Similar to backpropagation on feedforward ANN, but applied to each time step.
- Since we have several time steps, RNNs training is computationally expensive
- Backpropagation can be limited to a certain window in time by truncating backpropagation
- RNN training is sensitive to network topologies and hyperparameters.

MODEL PERFORMANCE ASSESSMENT – CONFUSION MATRIX

CONFUSION MATRIX



CONFUSION MATRIX

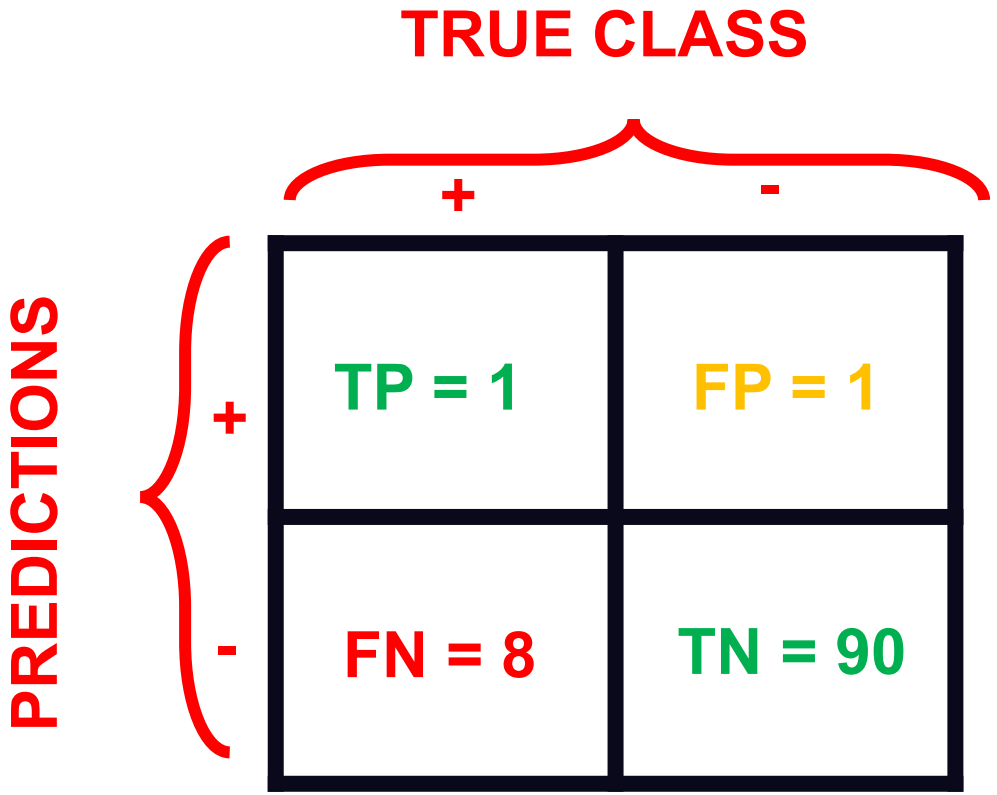
- A confusion matrix is used to describe the performance of a classification model:
  - True positives (TP): cases when classifier predicted TRUE (they have the disease), and correct class was TRUE (patient has disease).
  - True negatives (TN): cases when model predicted FALSE (no disease), and correct class was FALSE (patient do not have disease).
  - False positives (FP) (Type I error): classifier predicted TRUE, but correct class was FALSE (patient did not have disease).
  - False negatives (FN) (Type II error): classifier predicted FALSE (patient do not have disease), but they actually do have the disease

KEY PERFORMANCE INDICATORS (KPI)

- Classification Accuracy = (TP+TN) / (TP + TN + FP + FN)
- Misclassification rate (Error Rate) = (FP + FN) / (TP + TN + FP + FN)
- Precision = TP/Total TRUE Predictions = TP/ (TP+FP) (When model predicted TRUE class, how often was it right?)
- Recall = TP/ Actual TRUE = TP/ (TP+FN) (when the class was actually TRUE, how often did the classifier get it right?)

MODEL PERFORMANCE ASSESSMENT – PRECISION, RECALL AND F1-SCORE

PRECISION Vs. RECALL EXAMPLE



FACTS:
100 PATIENTS TOTAL
91 PATIENTS ARE HEALTHY
9 PATIENTS HAVE CANCER

- Accuracy is generally misleading and is not enough to assess the performance of a classifier.
- Recall is an important KPI in situations where:
  - Dataset is highly imbalanced; cases when you have small cancer patients compared to healthy ones.

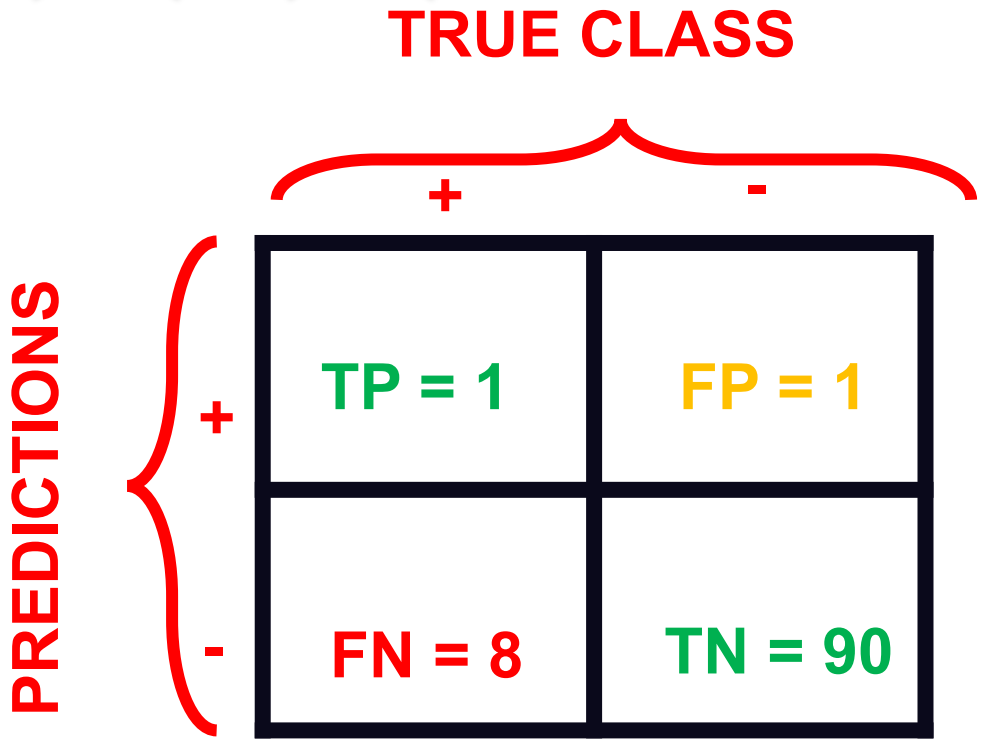
- Classification Accuracy = (TP+TN) / (TP + TN + FP + FN) = 91%
- Precision = TP/Total TRUE Predictions = TP/ (TP+FP) = ½=50%
- Recall = TP/ Actual TRUE = TP/ (TP+FN) = 1/9 = 11%

PRECISION DEEP DIVE

$$Precision = \frac{TRUE\ POSITIVES}{TOTAL\ TRUE\ PREDICTIONS}$$

$$Precision = \frac{TRUE\ POSITIVES}{TRUE\ POSITIVES + FALSE\ POSITIVES}$$

$$Precision = \frac{1}{1+1} = 50\%$$



NOTES:

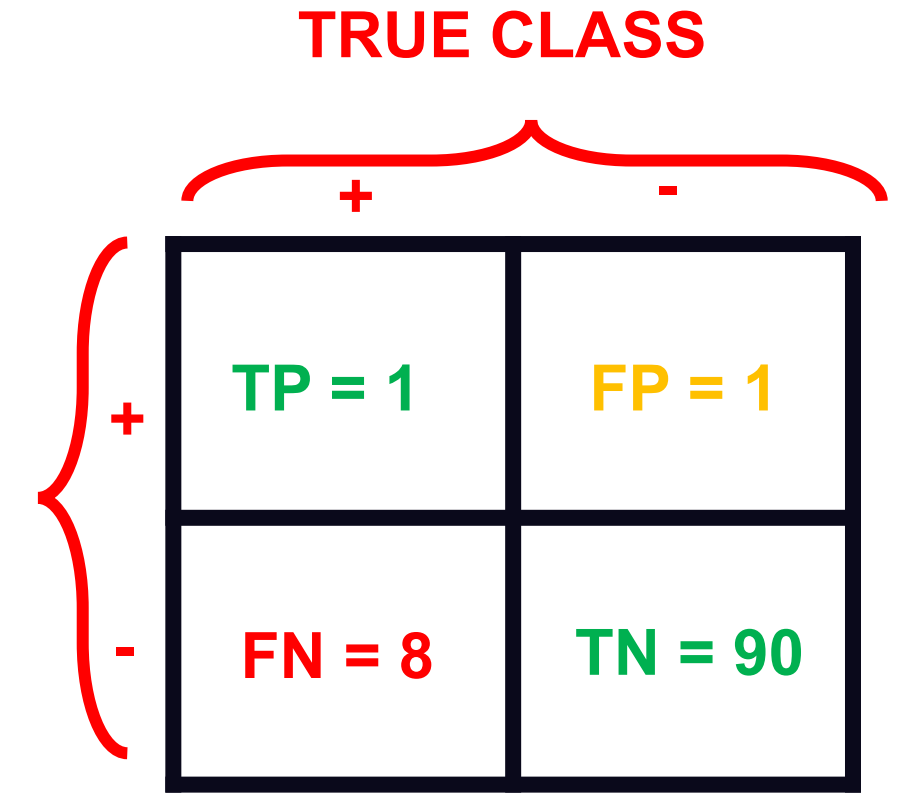
- Precision is a measure of Correct Positives, in this example, the model predicted two patients were positive classes (has cancer), only one of the two was correct.
- Precision is an important metric when False positives are important (how many times a model says a pedestrian was detected and there was nothing there!
- Examples include self driving cars and drug testing

RECALL DEEP DIVE

$$Recall = \frac{TRUE\ POSITIVES}{ACTUAL\ TRUE}$$

$$Recall = \frac{TRUE\ POSITIVES}{TRUE\ POSITIVES + FALSE\ NEGATIVES}$$

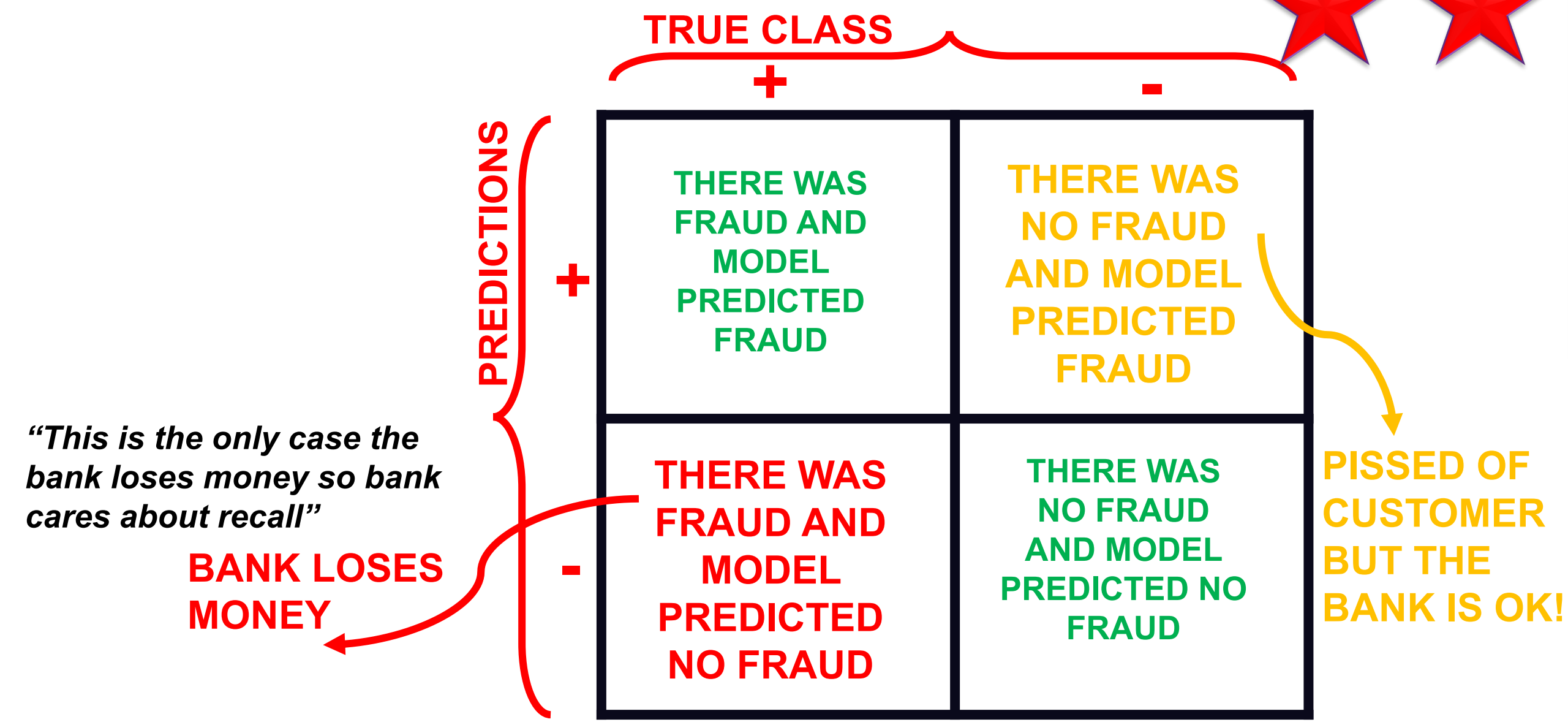
$$Recall = \frac{1}{1+8} = 11\%$$



NOTES:

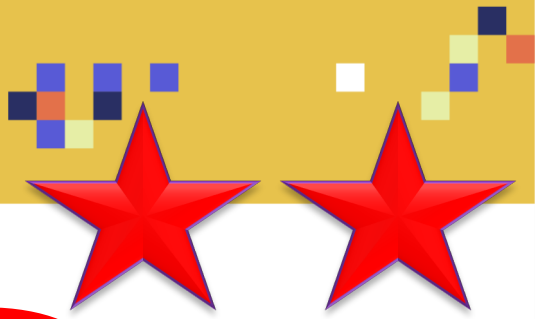
- Recall is also called True Positive rate or sensitivity
- In this example, I had 9 cancer patients but the model only detected 1 of them
- Important metric when we care about false negatives
- Example: Self driving cars and fraud detection

EX1: BANK FRAUD DETECTION



"This is the only case the bank loses money so bank cares about recall"

EX2: SPAM EMAIL DETECTION



TRUE CLASS

PREDICTIONS	+	-
+	THERE WAS SPAM EMAIL AND MODEL PREDICTED SPAM (BLOCKED IT)	THERE WAS NO SPAM EMAIL AND MODEL PREDICTED SPAM (BLOCKED IT)
-	THERE WAS A SPAM EMAIL AND MODEL PREDICTED NO SPAM (WENT TO INBOX)	THERE WAS NO SPAM EMAIL AND MODEL PREDICTED NO SPAM (WENT TO INBOX)

"This is a case when we care about precision and it's OK if we mess up recall a little bit"

NOT A BIG DEAL!

BLOCKED IMPORTANT EMAILS (DREAM JOB!)

F1 SCORE

$$F1\ Score\ = \frac{2\ *(PRECISION\ *RECALL)}{(PRECISION + RECALL)}$$

$$F1\ Score\ = \frac{2\ *TP}{2 * TP + FP + FN}$$

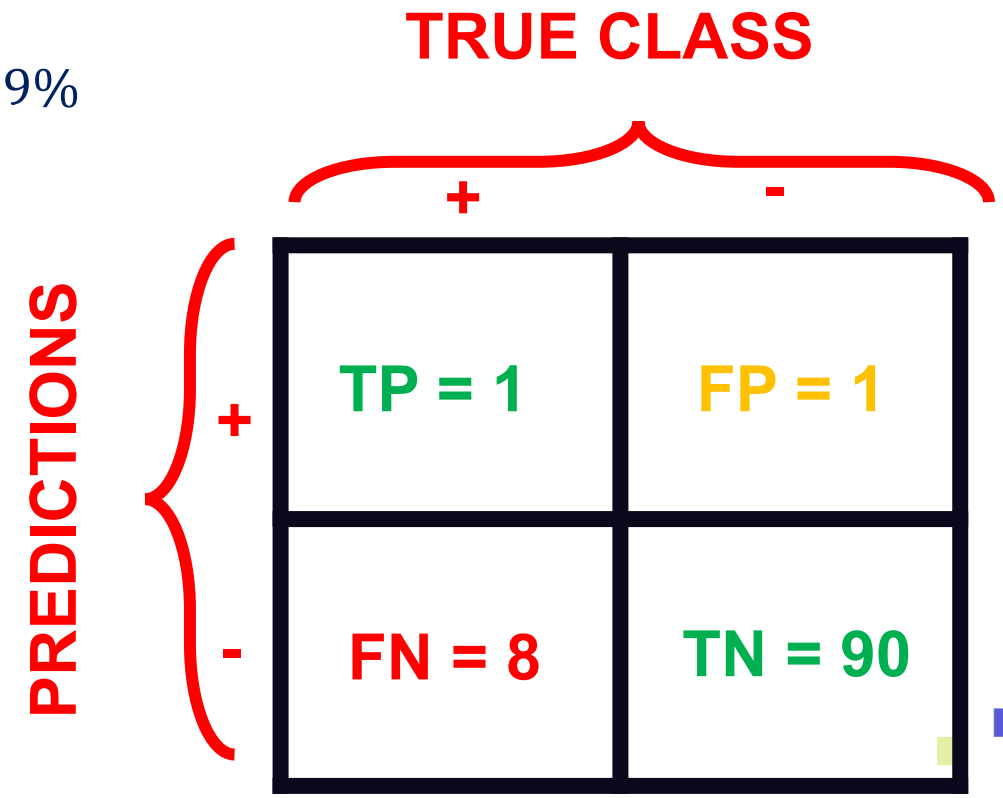
- F1 Score is an overall measure of a model's accuracy that combines precision and recall.
- F1 score is the harmonic mean of precision and recall.
- What is the difference between F1 Score and Accuracy?
- In unbalanced datasets, if we have large number of true negatives (healthy patients), accuracy could be misleading. Therefore, F1 score might be a better KPI to use since it provides a balance between recall and precision in the presence of unbalanced datasets.

SPECIFICTY

$$SPECIFICITY = TRUE\ NEGATIVE\ RATE = \frac{TRUE\ NEGATIVES}{TRUE\ NEGATIVES + FALSE\ POSITIVES}$$

$$SPECIFICITY = \frac{90}{90 + 1} = 98.9\%$$

- Specificity measures the proportion of actual negatives that are correctly identified as such
- Example: true negative rate indicates the percentage of healthy people who are correctly classified as healthy!



F1-SCORE PER CLASS: CANCER CLASSIFICATION DATASET

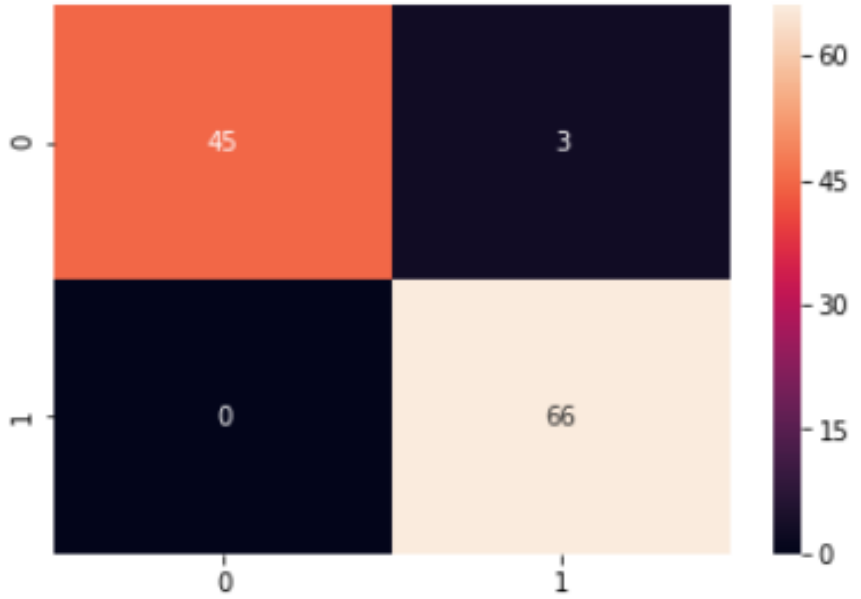
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	comp
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	

5 rows × 31 columns

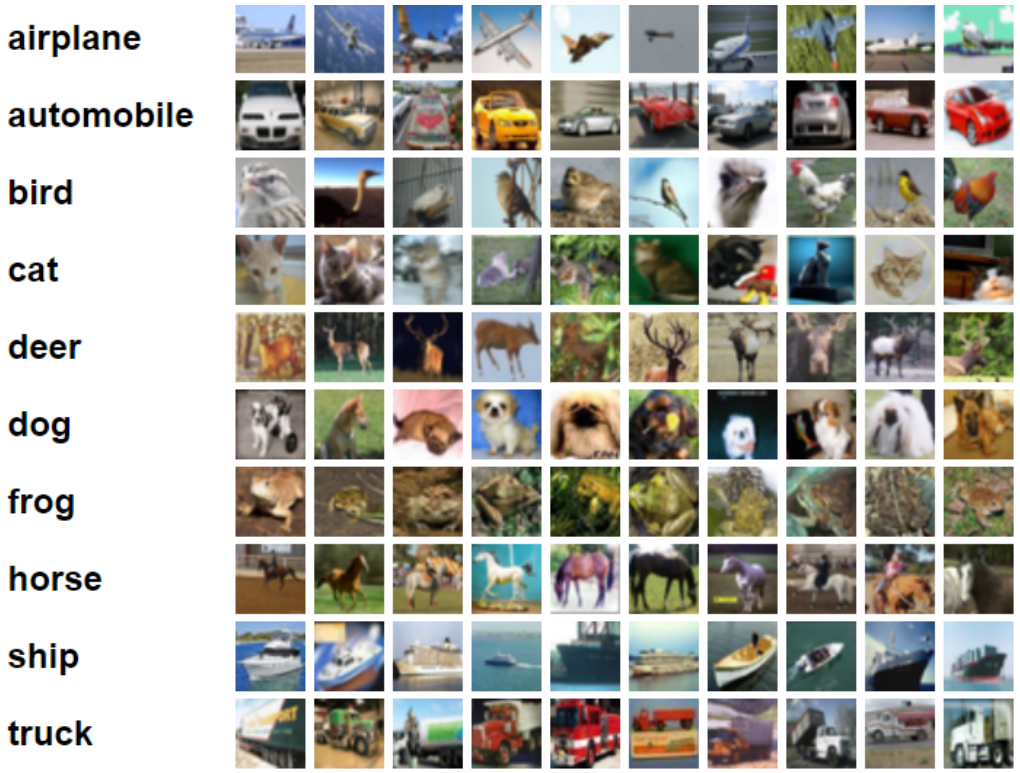
	precision	recall	f1-score
0.0	1.00	0.94	0.97
1.0	0.96	1.00	0.98
avg / total	0.97	0.97	0.97

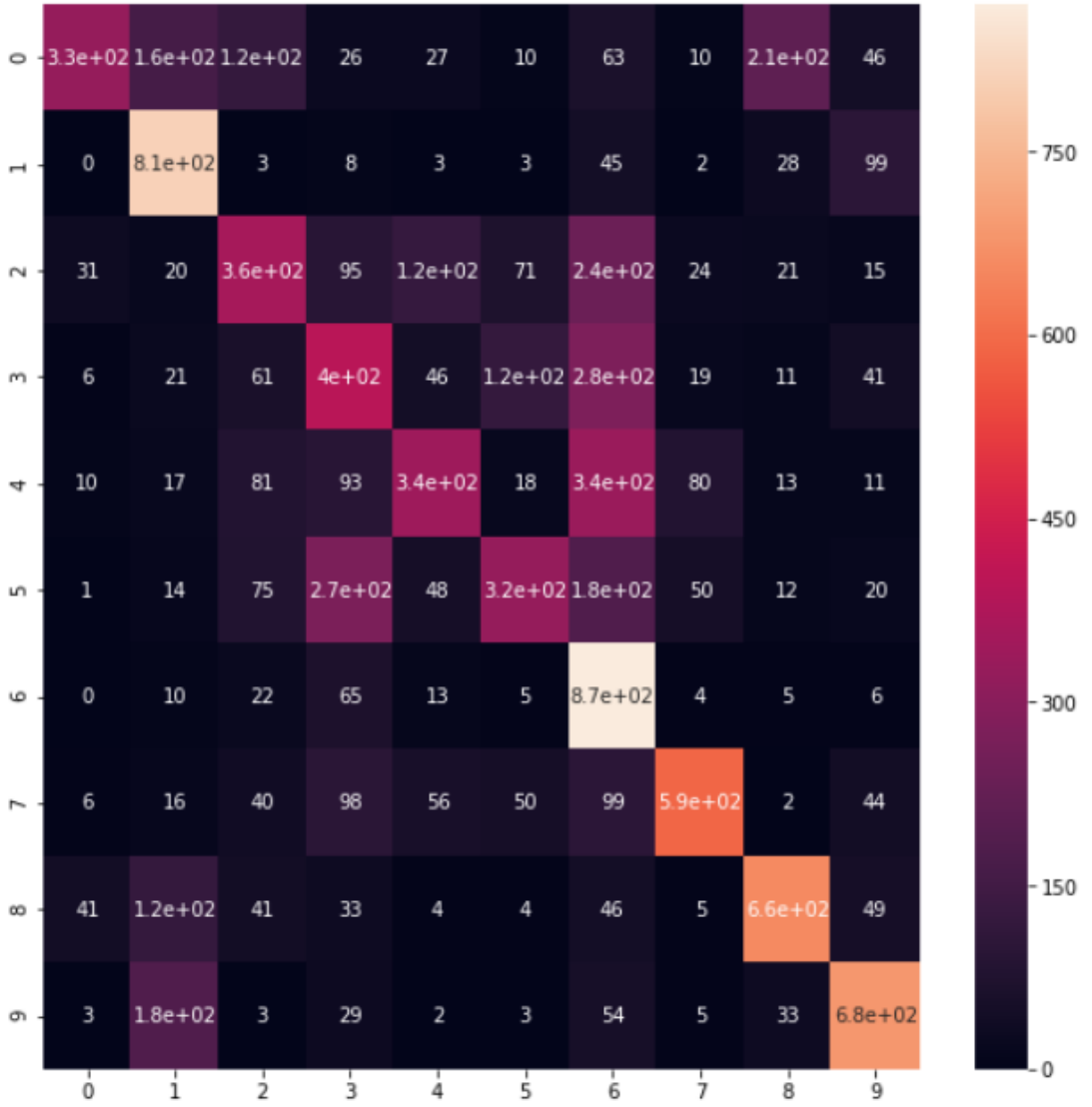
F1-SCORE PER CLASS

AVERAGE F1-SCORE



MULTICLASS CLASSIFICATION

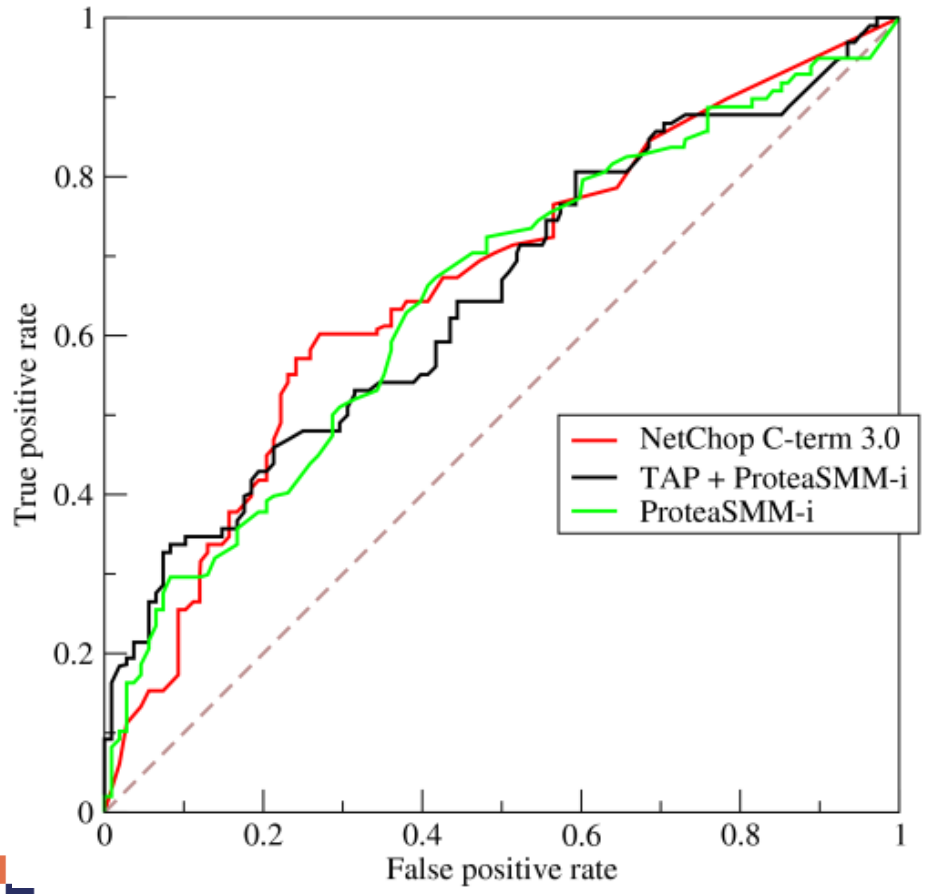




https://www.cs.toronto.edu/~kriz/cifar.html

MODEL PERFORMANCE ASSESSMENT – ROC, AUC, HEATMAP, RMSE

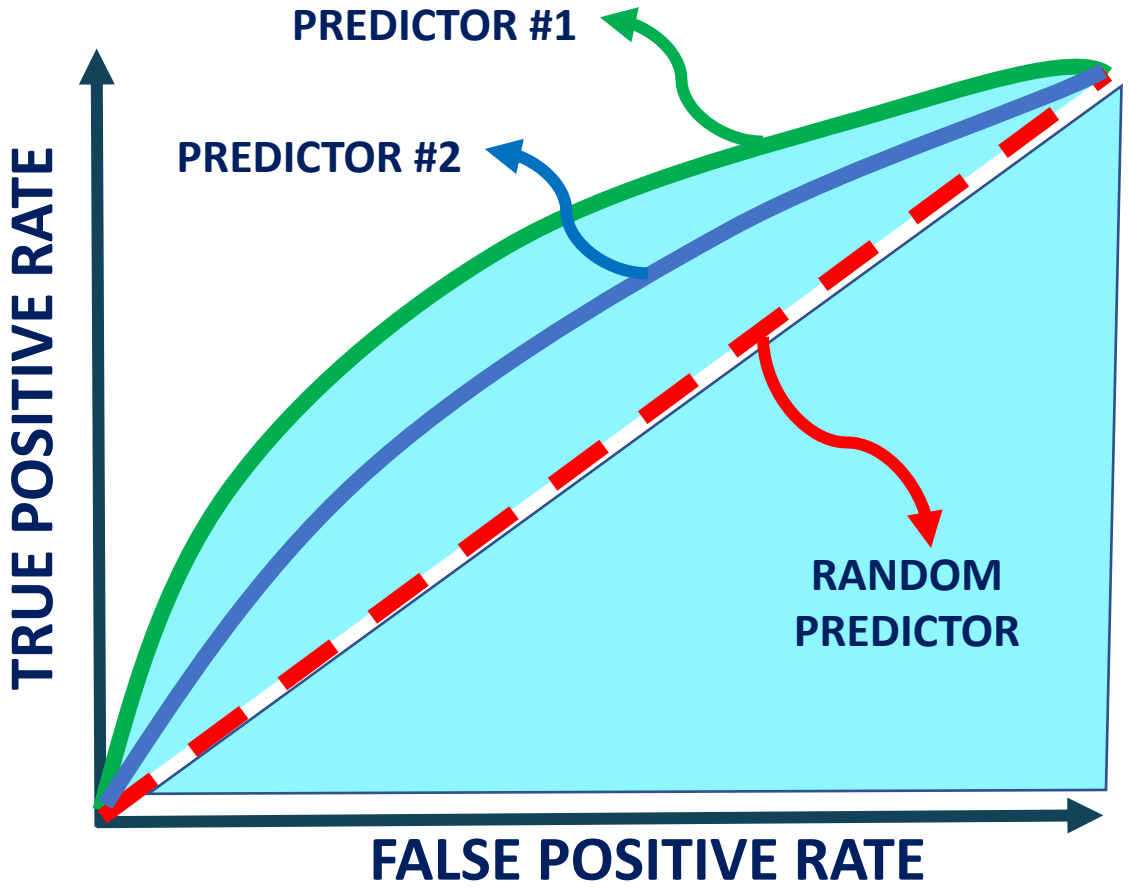
ROC (RECEIVER OPERATING CHARACTERISTIC CURVE)



- ROC Curve is a metric that assesses the model ability to distinguish between binary (0 or 1) classes.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
- The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning.
- The false-positive rate is also known as the probability of false alarm and can be calculated as (1 − specificity).
- Points above the diagonal line represent good classification (better than random)
- The model performance improves if it becomes skewed towards the upper left corner.

Photo Credit: https://commons.wikimedia.org/wiki/File:Roccurves.png

AUC (AREA UNDER CURVE)



- The light blue area represents the area Under the Curve of the Receiver Operating Characteristic (AUROC).
- The diagonal dashed red line represents the ROC curve of a random predictor with AUROC of 0.5.
- If ROC AUC = 1, perfect classifier
- Predictor #1 is better than predictor #2
- Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s.

CONFUSION MATRIX AND HEAT MAP

REGRESSION METRICS: ROOT MEAN SQUARE ERROR (RMSE)

- Root Mean Square Error (RMSE) represents the **standard deviation of the residuals** (i.e.: differences between the model predictions and the true values (training data)).
- RMSE can be **easily interpreted** compared to MSE because RMSE units match the units of the output.
- RMSE provides an estimate of how large the residuals are being dispersed.
- The RMSE is calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2}$$

- RMSE is calculated by following these steps:
  1. Calculate the residual for every data point
  2. Calculate the squared value of the residuals
  3. Calculate the average of the squared residuals
  4. Obtain the square root of the result

TRANSFER LEARNING BASICS

WHAT IS TRANSFER LEARNING?

- Transfer learning is a machine learning technique in which a network that has been trained to perform a specific task is being **reused (repurposed)** as a starting point for another similar task.
- Transfer learning is widely used since starting from a pre-trained models can dramatically **reduce the computational time** required if training is performed from scratch.

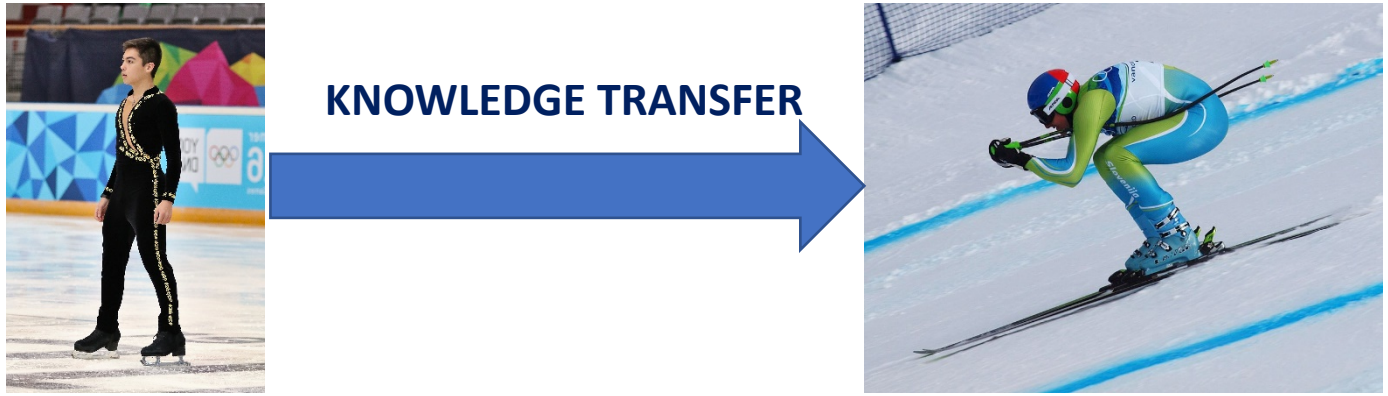
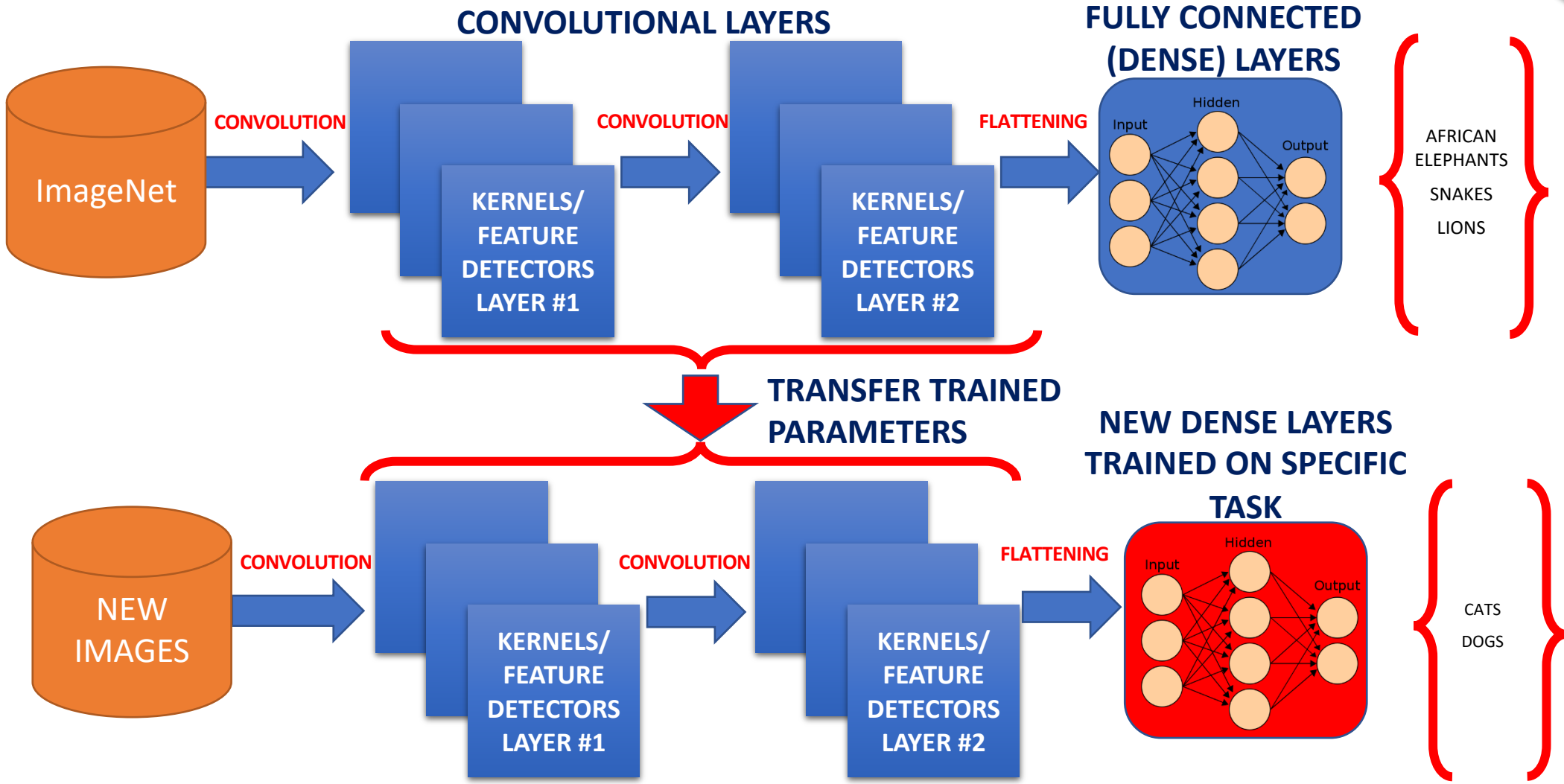


Photo Credit: https://commons.wikimedia.org/wiki/File:Lillehammer_2016_-_Figure_Skating_Men_Short_Program_-_Camden_Pulkinen_2.jpg
Photo Credit: https://commons.wikimedia.org/wiki/Alpine_skiing#/media/File:Andrej_%C5%A0porn_at_the_2010_Winter_Olympic_downhill.jpg
Citations: Olga Russakovsky, *Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei.
ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575, 2014.

WHAT IS TRANSFER LEARNING

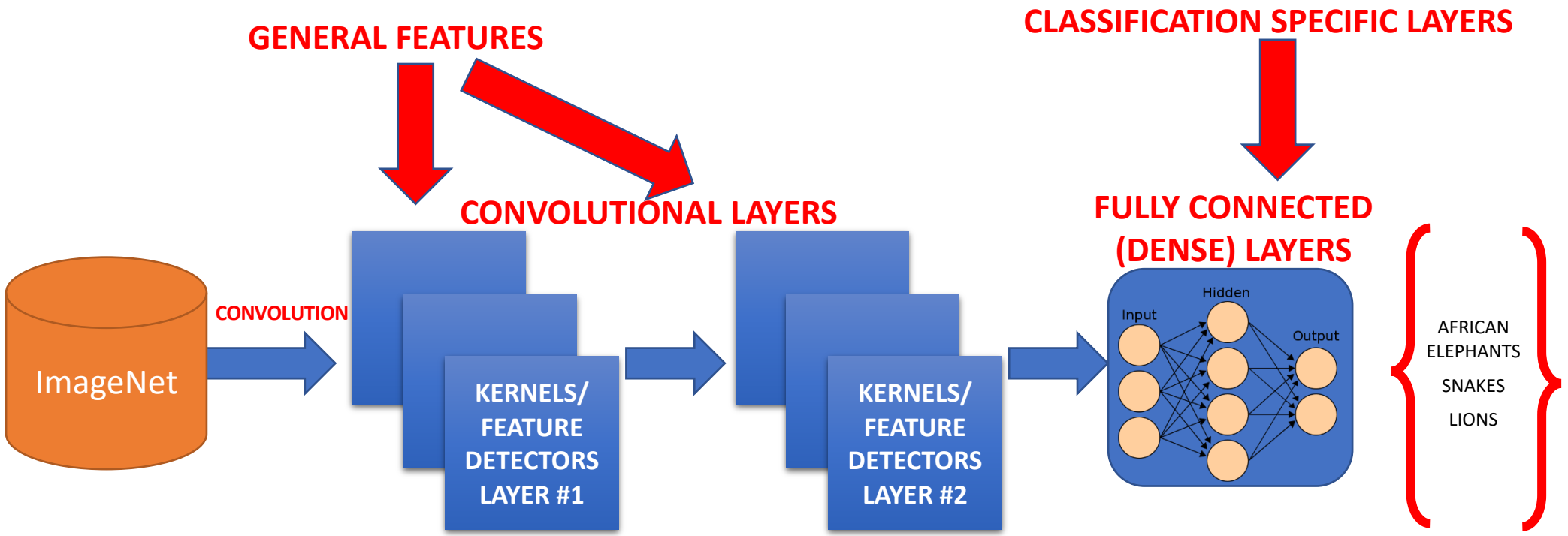
- “Transfer learning is the improvement of learning in a new task through the **transfer of knowledge** from a related task that has already been learned”—Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.
- In transfer learning, **a base (reference)** Artificial Neural Network on a base dataset and function is being trained. Then, this trained network weights are then **repurposed in a second ANN** to be trained on a new dataset and function.
- Transfer learning works great if the **features are general**, such that trained weights can effectively repurposed.
- Intelligence is being transferred from the base network to the newly target network.

TRANSFER LEARNING PROCESS



WHY DO WE KEEP THE FIRST LAYERS?

- The first CNN layers are used to **extract high level general features.**
- The last couple of layers are used to perform classification (on a specific task).
- So we copy the first trained layers **(base model)** and then we **add a new custom layers** in the output to perform classification on a specific new task.



TRANSFER LEARNING TRAINING STRATEGIES

- **Strategy #1 Steps:**
  - Freeze the trained CNN network weights from the first layers.
  - Only train the newly added dense layers (with randomly initialized weights).

- **Strategy #2 Steps:**
  - Initialize the CNN network with the pre-trained weights
  - Retrain the **entire CNN** network while setting the learning rate to be very small, this is critical to ensure that you do not aggressively change the trained weights.

TRANSFER LEARNING ADVANTAGES

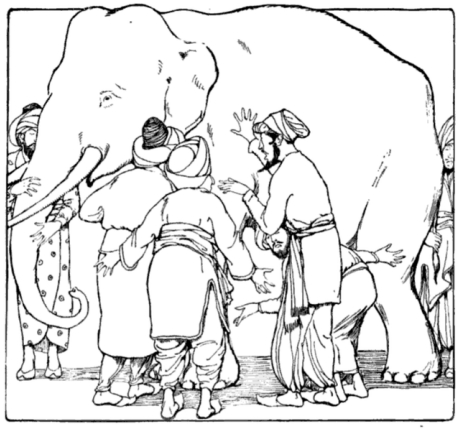
- Transfer learning advantages are:
  - Provides fast training progress, you don’t have to start from scratch using randomly initialized weights
  - You can use small training dataset to achieve incredible results

- When to use transfer learning?
  - When there is a small training dataset available for your new task but there exist a large dataset in a different domain (such as ImageNet)
  - When you have limited computational resources (GPU, TPU)

ENSEMBLE LEARNING METHODS

ENSEMBLE LEARNING

- Ensemble techniques such as bagging and boosting can offer an extremely powerful algorithm by combining a group of relatively weak/average ones.
- For example, you can combine several decision trees to create a powerful random forest algorithm
- By Combining votes from a pool of experts, each will bring their own experience and background to solve the problem resulting in a better outcome.
- Bagging and Boosting can reduce variance and overfitting and increase the model robustness.
- Example: Blind men and the elephant!



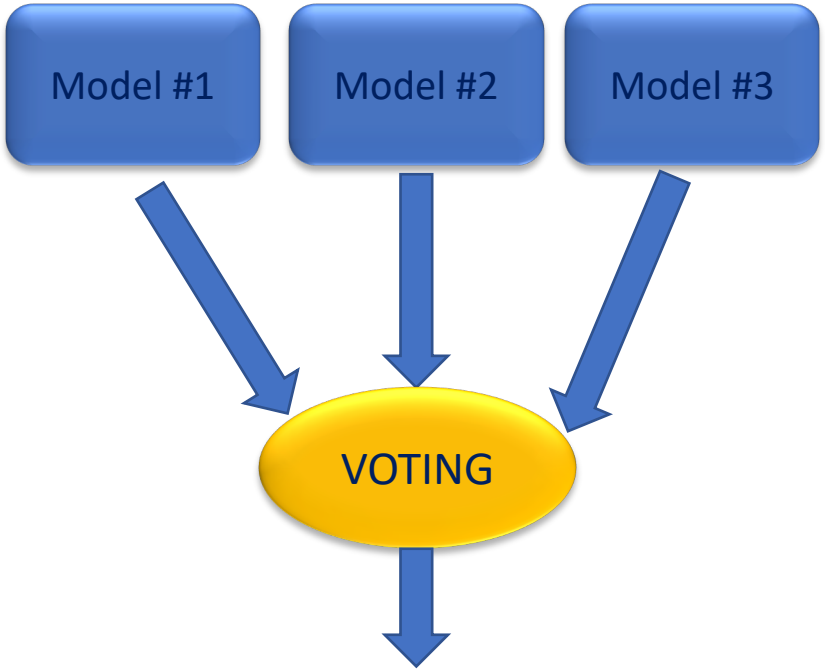
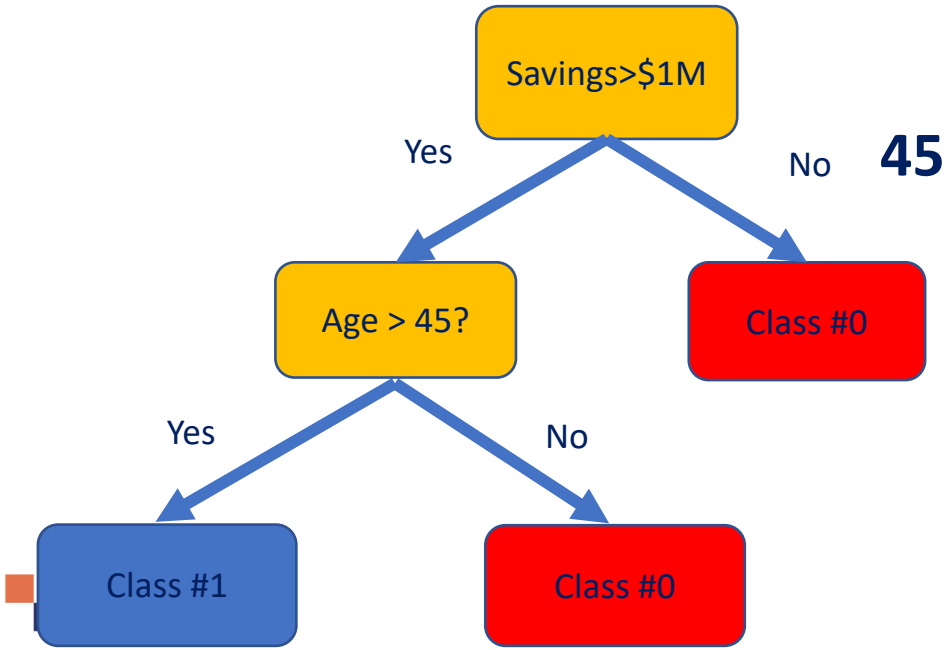
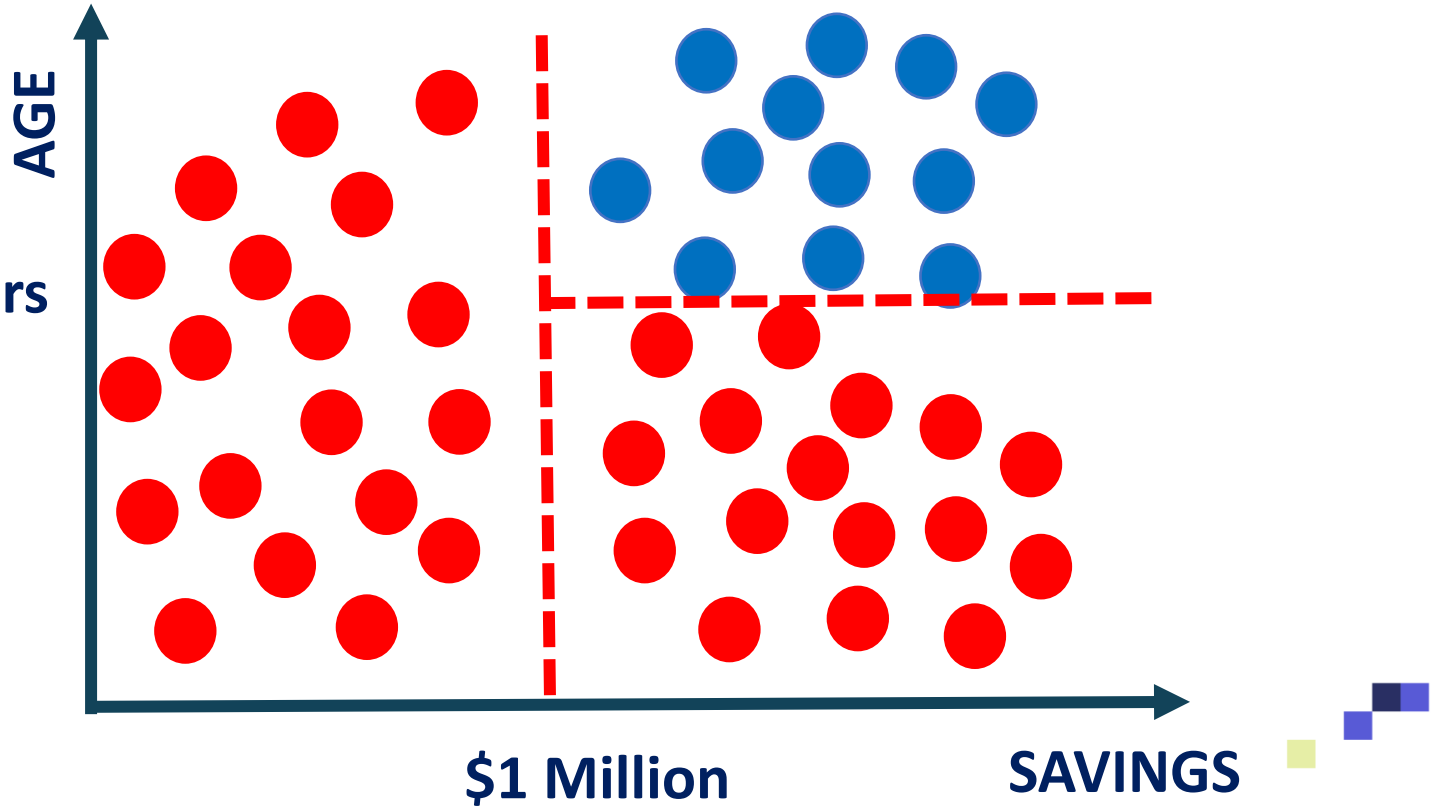


Photo Credit: https://commons.wikimedia.org/wiki/File:Blind_men_and_elephant.png

DECISION TREES: INTUITION

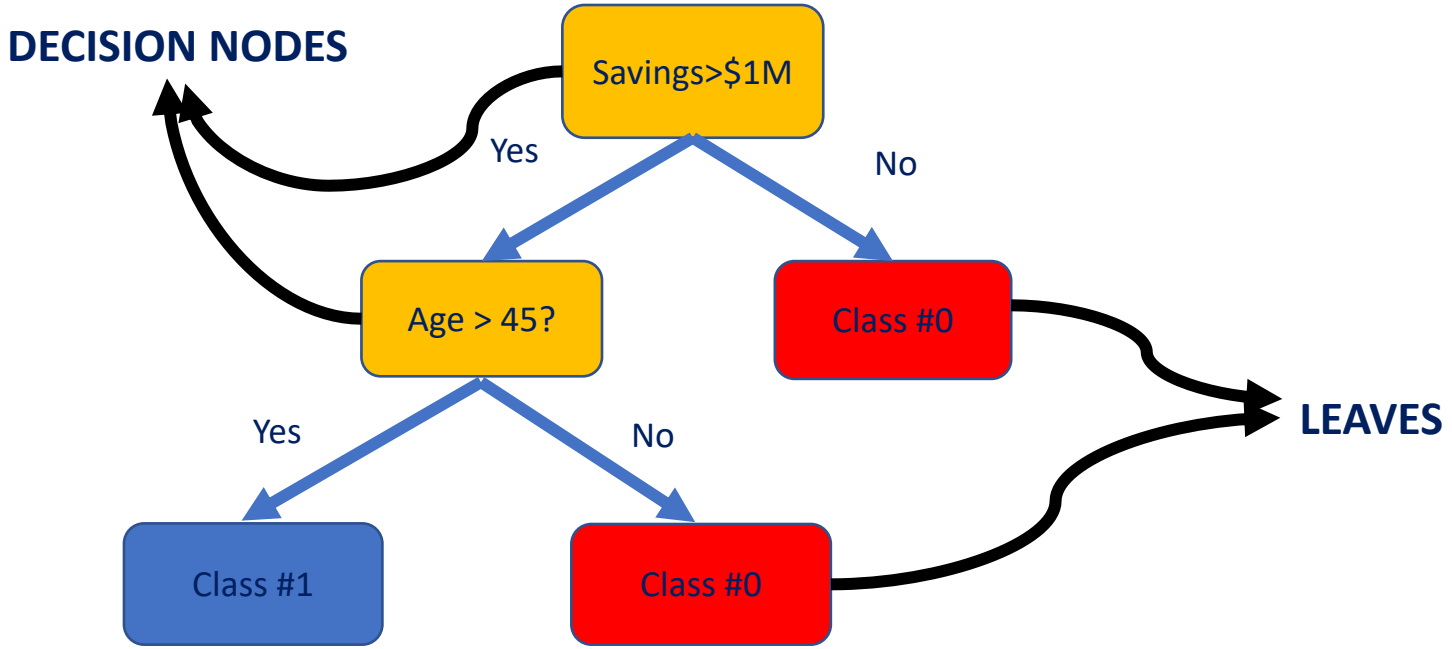
- Decision Trees are supervised Machine Learning technique where the data is split according to a certain condition/parameter.
- Let's assume we want to classify whether a customer could retire or not based on their savings and age.



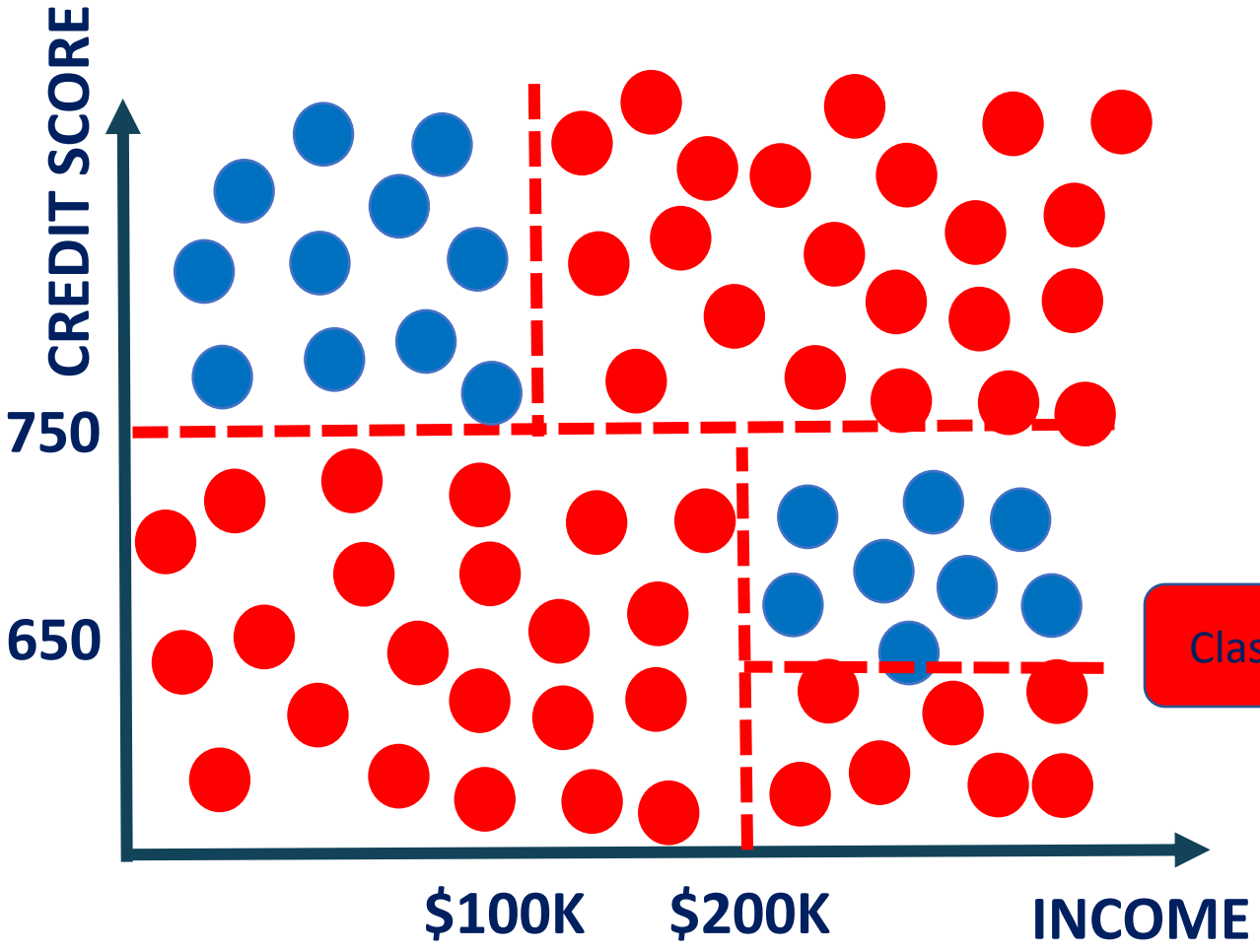


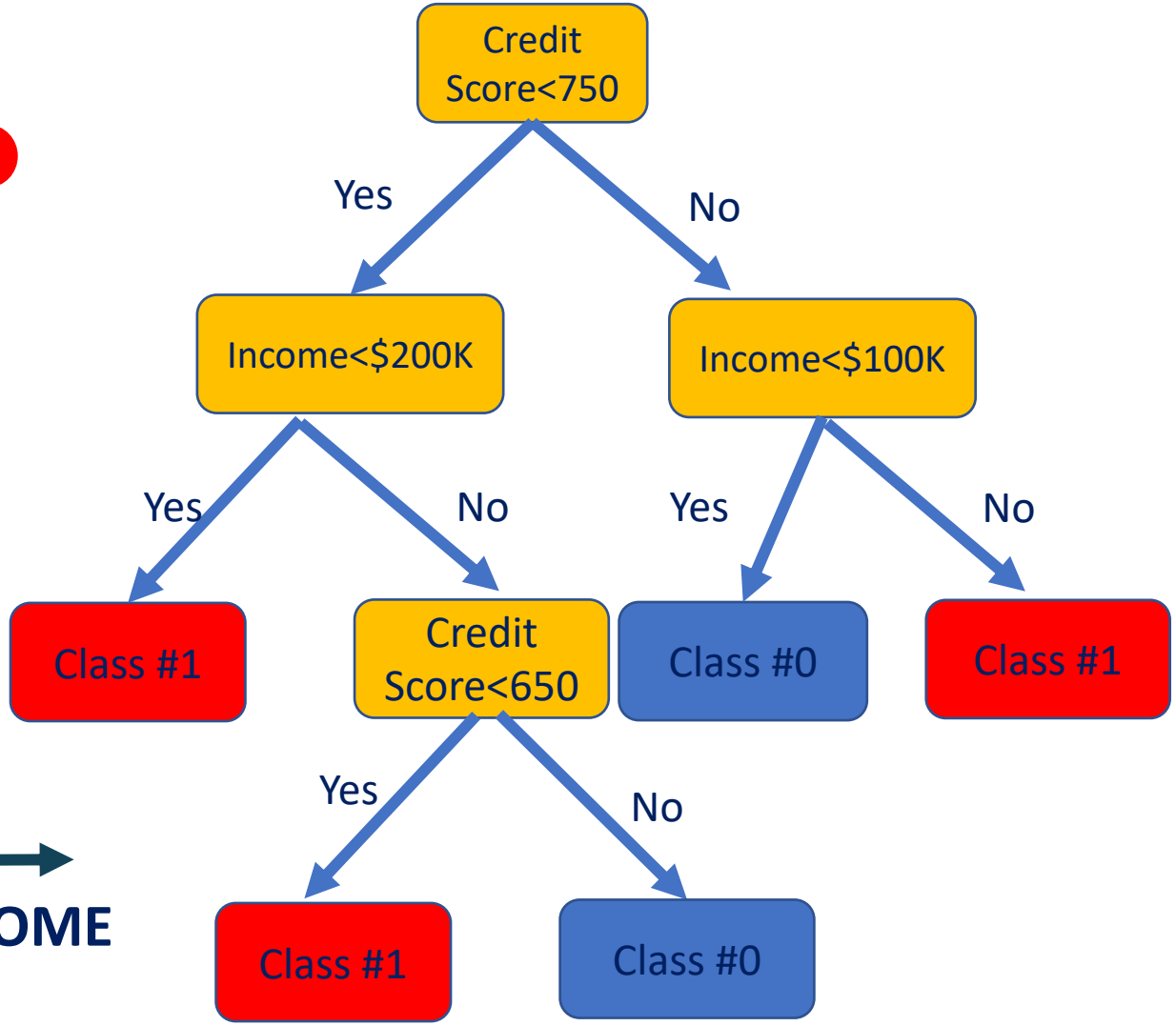
DECISION TREES: DEFINITIONS

- The tree consists of **decision nodes** and **leaves**.
- Leaves are the decisions or the final outcomes.
- Decision nodes are where the data is split based on a certain attribute.
- Objective is to minimize the entropy which provides the optimum split



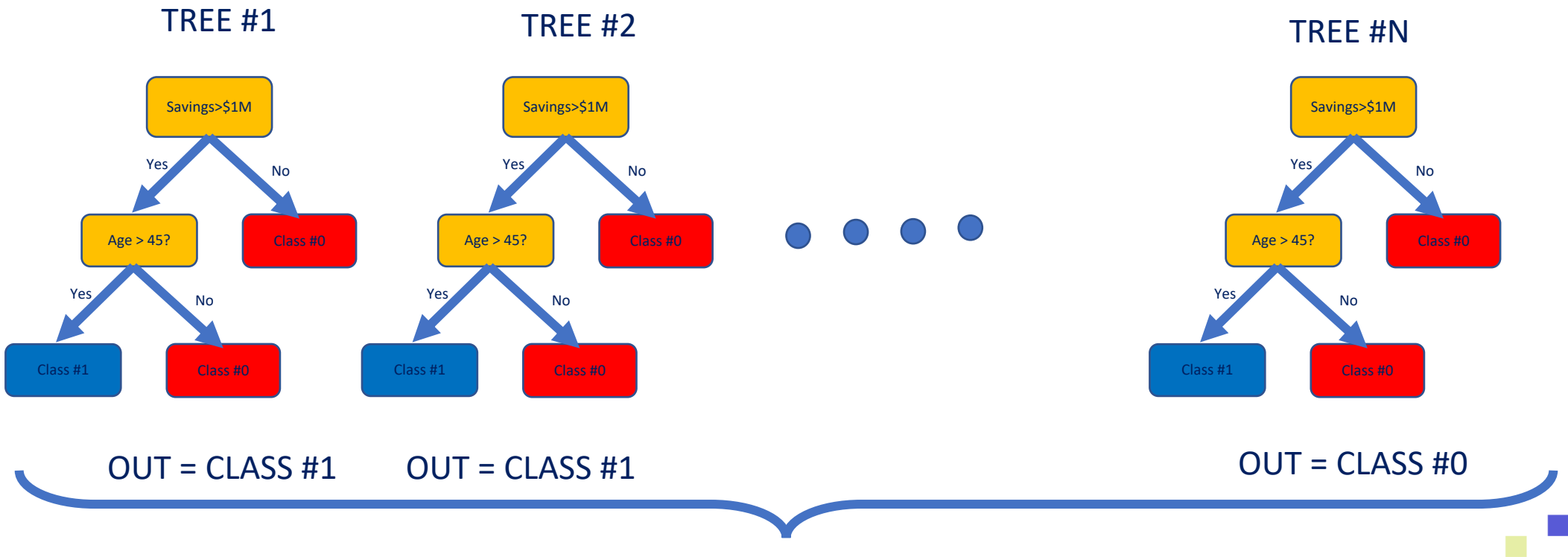
DECISION TREES: CUSTOMER SEGMENTATION





RANDOM FOREST: INTUITION

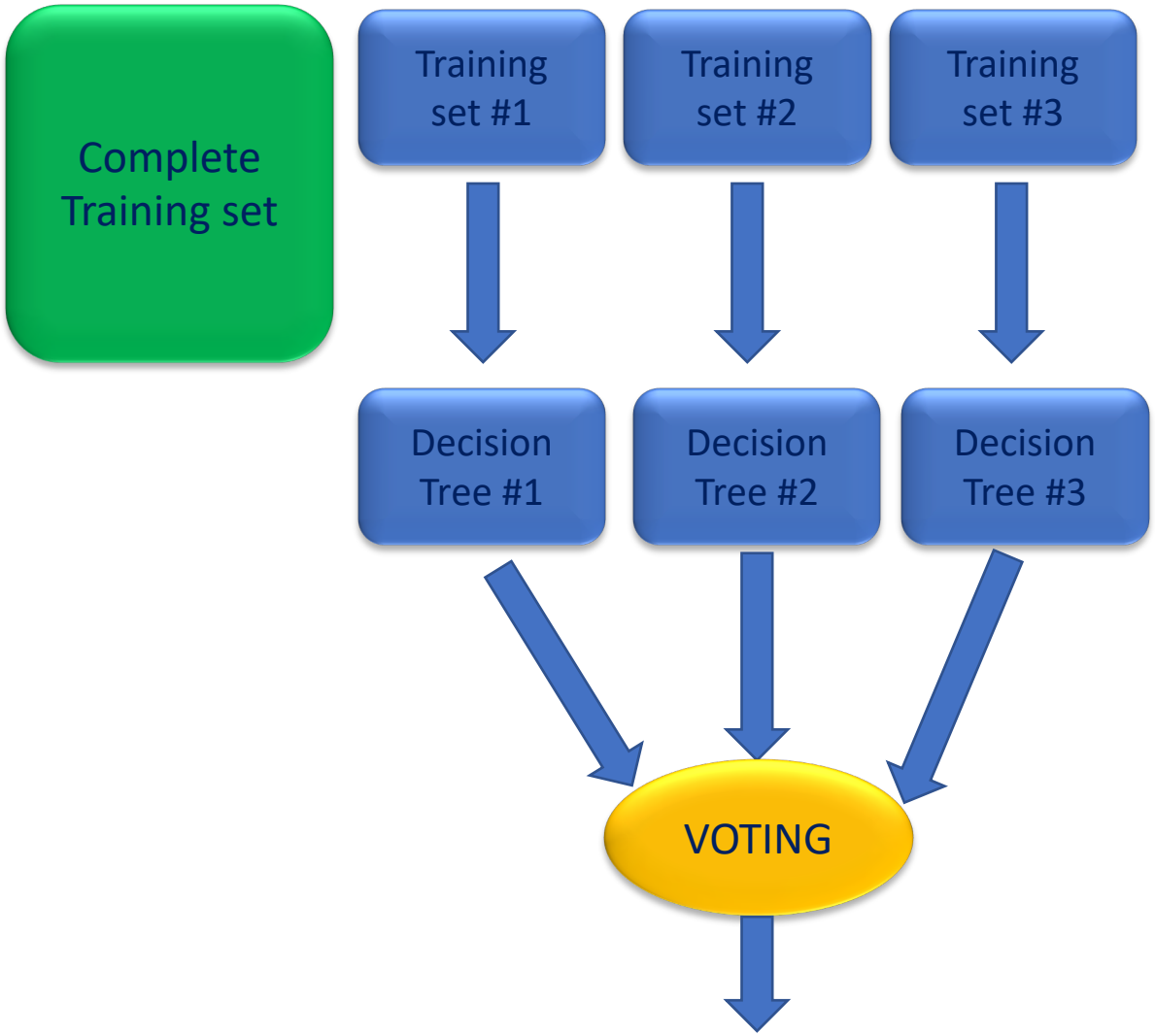
- Random Forest Classifier is a type of **ensemble algorithm**.
- It creates a set of decision trees from randomly selected subset of training set.
- It then **combines votes** from different decision trees to decide the final class of the test object.



MAJORITY VOTE = CLASS #1

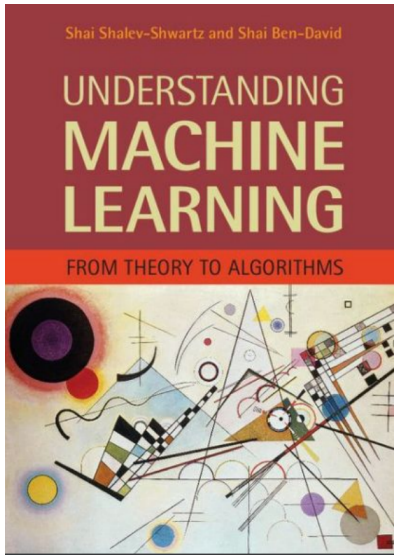
RANDOM FOREST: WHY AND HOW?

- It overcomes the issues with single decision trees by reducing the effect of noise.
- Overcomes **overfitting problem** by taking **average of all the predictions**, canceling out biases.
- Suppose training set: [X1, X2, X3, X4] with labels: [L1, L2, L3, L4]
- Random forest creates three decision trees taking inputs as follows:
  - [X1, X2, X3], [X1, X2, X4], [X2, X3, X4]
- Example: Combining votes from a pool of experts, each will bring their own experience and background to solve the problem resulting in a better outcome.

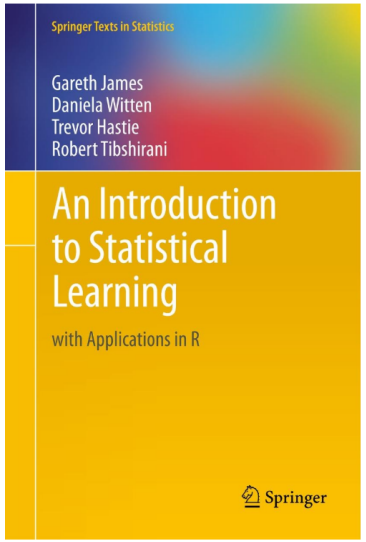


RANDOM FOREST: ADDITIONAL READING MATERIAL

Additional Resources, Page #255:
http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf



Additional Resources, Page #320:
http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf



BAGGING – INDEPENDANTLY AND IN PARALLEL!

- Bagging is a powerful ensemble algorithm in which N new training datasets are generated from a given “main” pool of data.
- Dataset is sampled with replacement.
- Several models are being trained in parallel.
- Prediction based on the aggregate predictions of all models is made.
- Bagging avoids overfitting

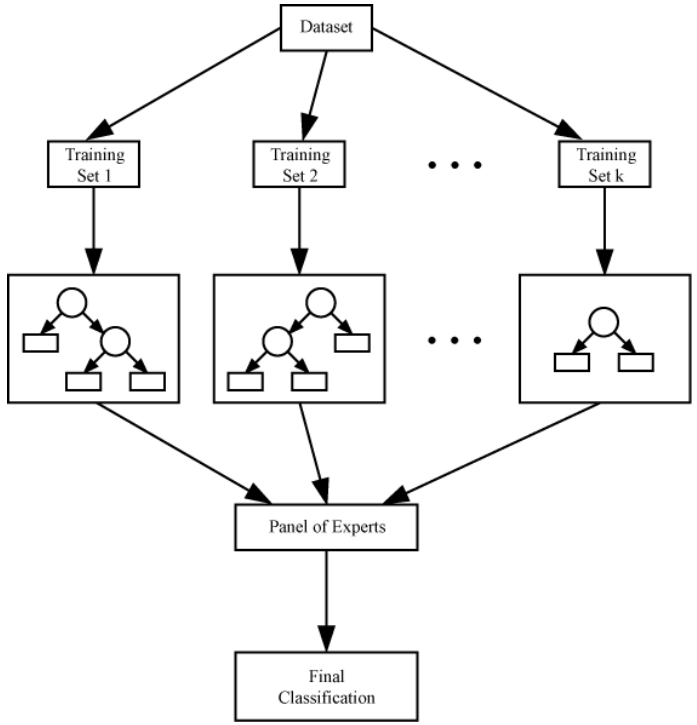


Photo Credit: https://commons.wikimedia.org/wiki/File:DTE_Bagging.png

BOOSTING – SEQUENTIALLY

- Boosting is an ensemble algorithm that combines weighted averages to turn a weak model into an extremely powerful one!
- Boosting differs from bagging in which various models work cooperatively to achieve the best results. The output from the first model is reweighted and fed to another model.
- Recall that in bagging, each model ran independently and then we combined the output at the end!
- XGBoost is an extremely powerful algorithm that won several Kaggle competitions recently.
- Boosting could potentially lead to a better accuracy compared to bagging.

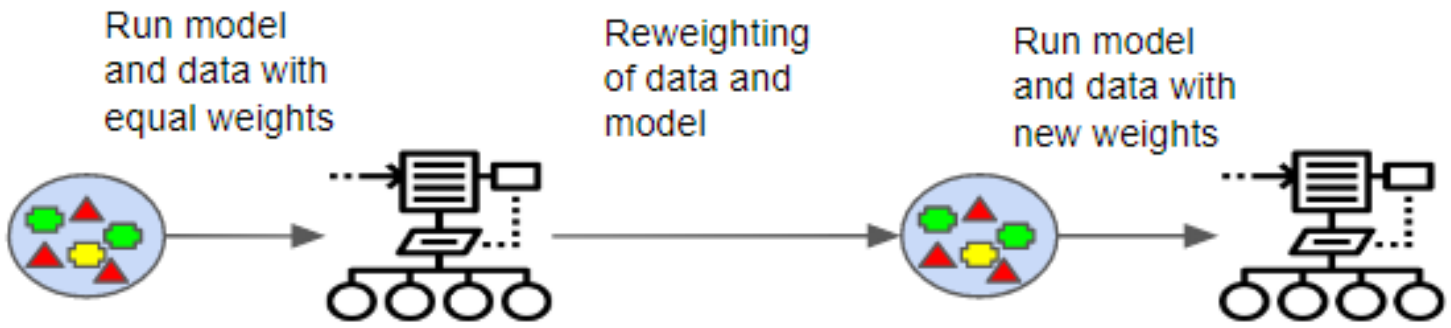


Photo Credit: https://commons.wikimedia.org/wiki/File:Boosting.png

K-FOLD CROSS VALIDATION

K-FOLD CROSS VALIDATION

- Cross-validation is used to assess the generalization capability of the model (with data that has never been seen before).
- K-fold cross validation is a technique used to assess the performance of machine learning models.
- K-Fold works by randomly dividing the data into equal sized K groups (folds).
- First fold is considered the validation dataset, and the model is trained on the remaining k – 1 folds.
- By comparing the performance of the model with these many folds, you can now know if the model is over fitted or not.
- If the error is comparable across various runs then the model is generalized.

Model1: *Trained on Fold1 + Fold2 and Tested on Fold3*
Model2: *Trained on Fold2 + Fold3 and Tested on Fold1*
Model3: *Trained on Fold1 + Fold3 and Tested on Fold2*