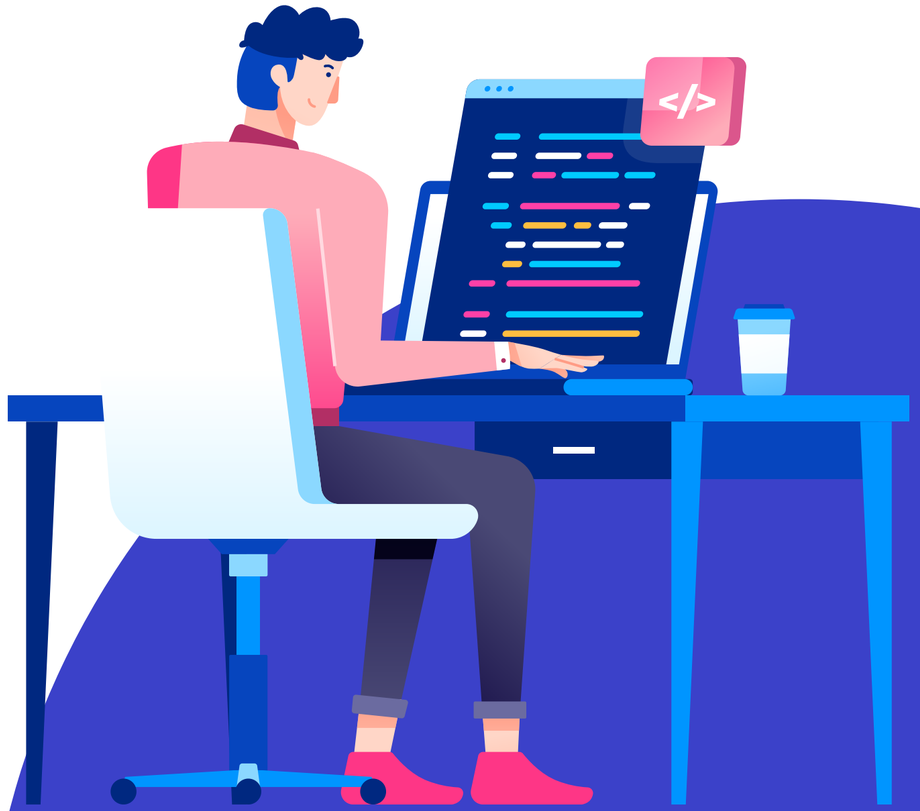
MODEL TUNING

SDS X IEEE



Search . . .

MEET THE TEAM

MARCUS

Y1 Data Science & Analytics

HUI XIN

Y2 Data Science & Analytics

SHANKAR

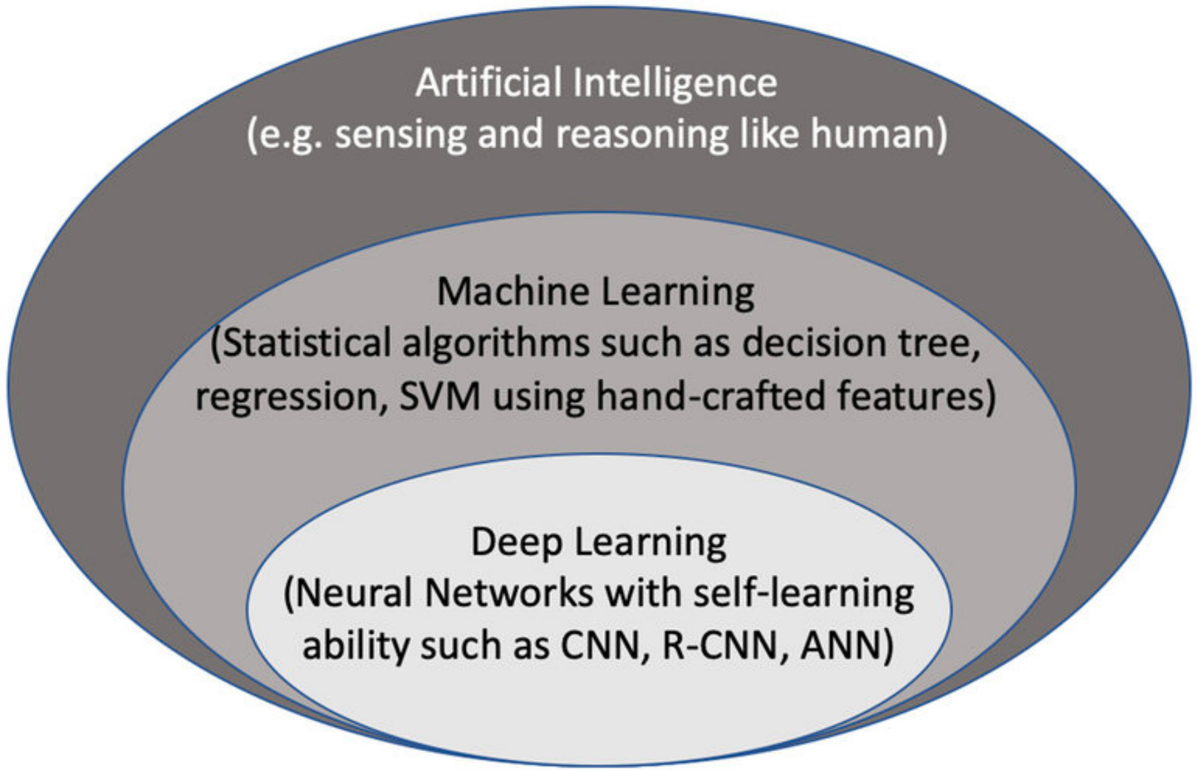
Y3 Electrical Engineering

OUTLINE OF THE WORKSHOP

1. Intro to Deep Learning
2. Intro to Dataset
3. Model Evaluation Methods
4. Model Training / Tuning
5. End of Workshop + Q&A

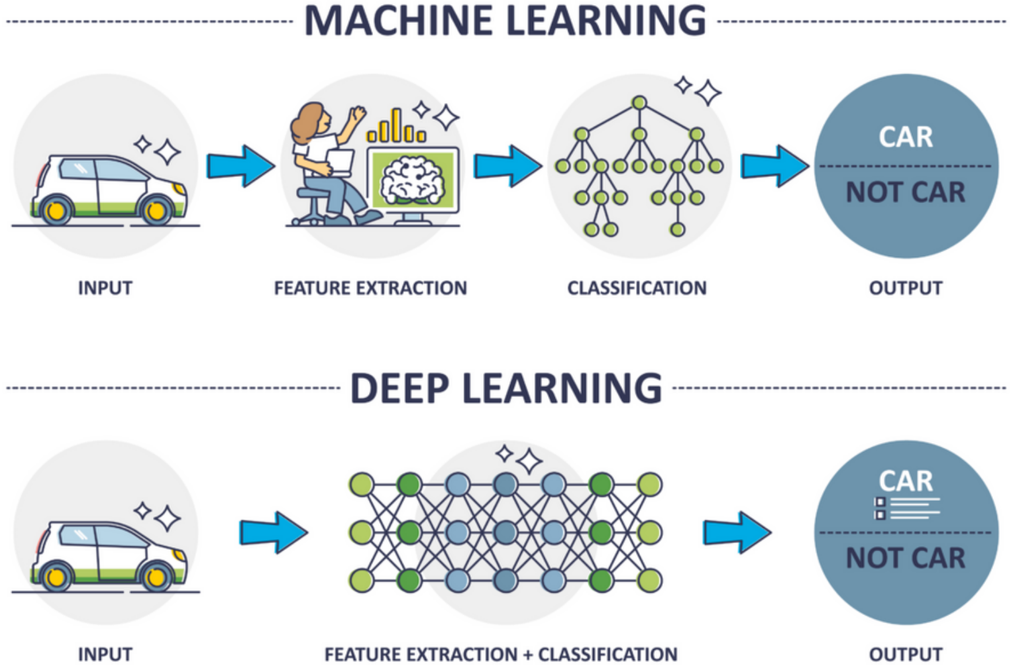
1

INTRO TO DEEP LEARNING



WHAT IS DEEP LEARNING?

- Subset of machine learning that is designed to **handle large and complex datasets**.
- Involves the use of **artificial neural networks** that are inspired by the structure and function of the human brain.
- Deep learning algorithms are able to learn patterns and features from data **without explicit programming or human intervention**. (use of multiple layers in the neural network)

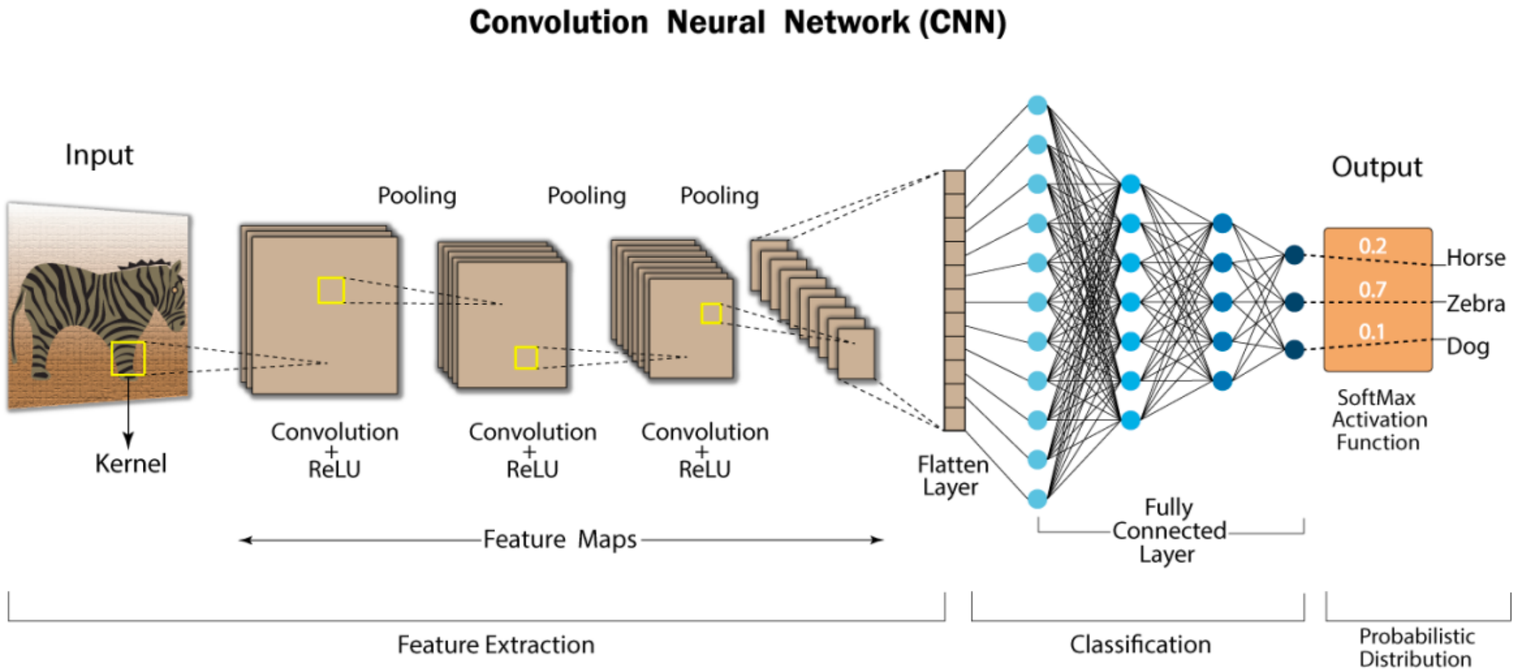


TYPES OF NEURAL NETWORKS

	ANN	CNN	RNN
Basics	One of the simplest types of neural networks.	One of the most popular types of neural networks.	The most advanced and complex neural network.
Structural Layout	Its simplicity comes from its feed forward nature — information flows in one direction only.	Its structure is based on multiple layers of nodes including one or more convolutional layers.	Information flows in different directions, which gives it its memory and self-learning features.
Data Type	Fed on tabular and text data.	Relies on image data.	Trained with sequence data.
Complexity	Simple in contrast with the other two models.	Considered more powerful than the other two.	Fewer features than CNN but powerful due to its self-learning & memory potential.
Commendable Feature	Ability to work with incomplete knowledge and high fault tolerance.	Accuracy in recognizing images.	Memory and self-learning.
Feature type: spatial recognition	No	Yes	No
Feature type: Recurrent connections	No	No	Yes
Main Drawback	Hardware dependence.	Large training data required.	Slow and complex training and gradient concerns.
Uses	Complex problem solving such as predictive analysis.	Computer vision including image recognition	Natural language processing including sentiment analysis and speed recognition.

Taken from: https://levity.ai/blog/neural-networks-cnn-ann-rnn

WHAT IS CNN?



1 Convolutional Layer

The convolution layer is the building block of CNN carrying the main responsibility for computation.

2 Pooling Layer

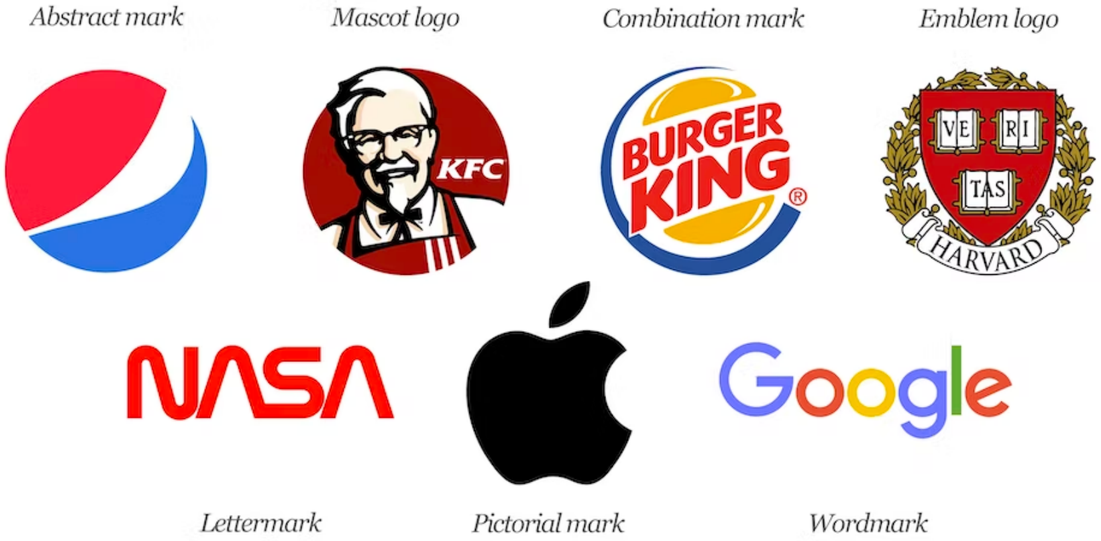
Pooling reduces the spatial size of the representation and lessens the number of computations required

3 Fully Connected Layer

The FC layer helps to map the representation between the input and the output

SOME APPLICATIONS OF CNN

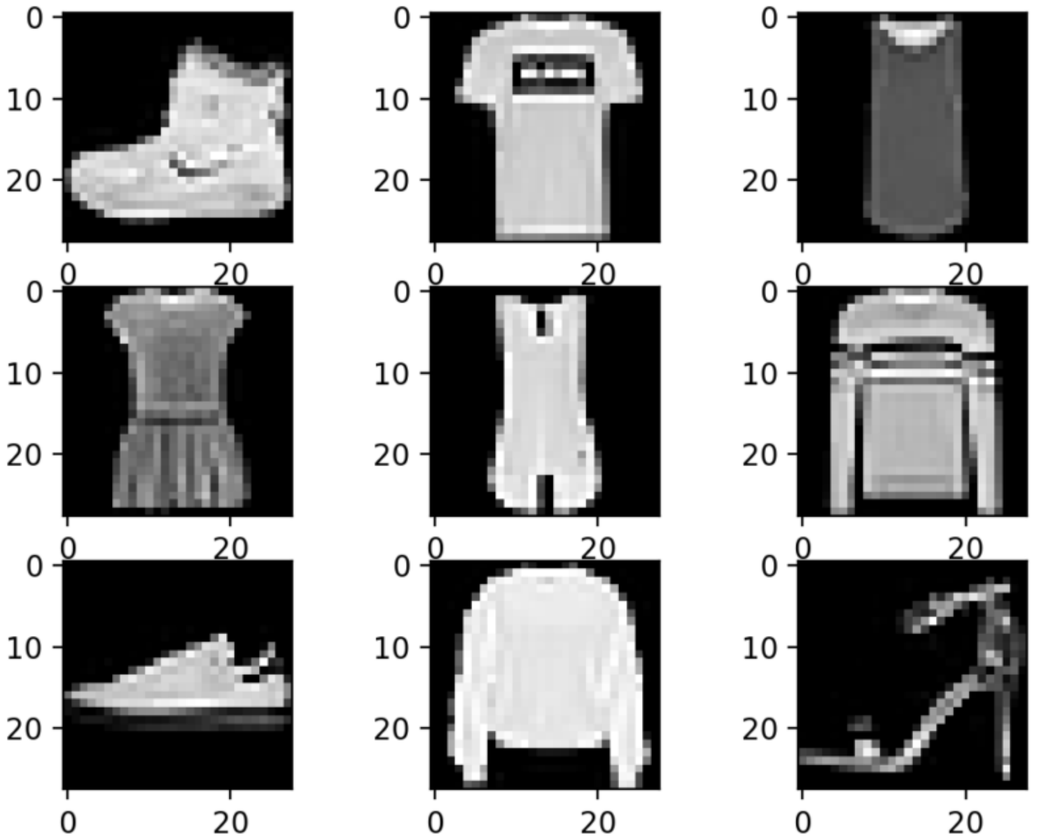
Diagnose health diseases from medical scans



Detect a company logo in social media to better understand joint marketing opportunities

2

INTRO TO FASHION-MNIST DATASET



FASHION-MNIST DATASET

Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms.

60000

Training Set

10000

Test Set

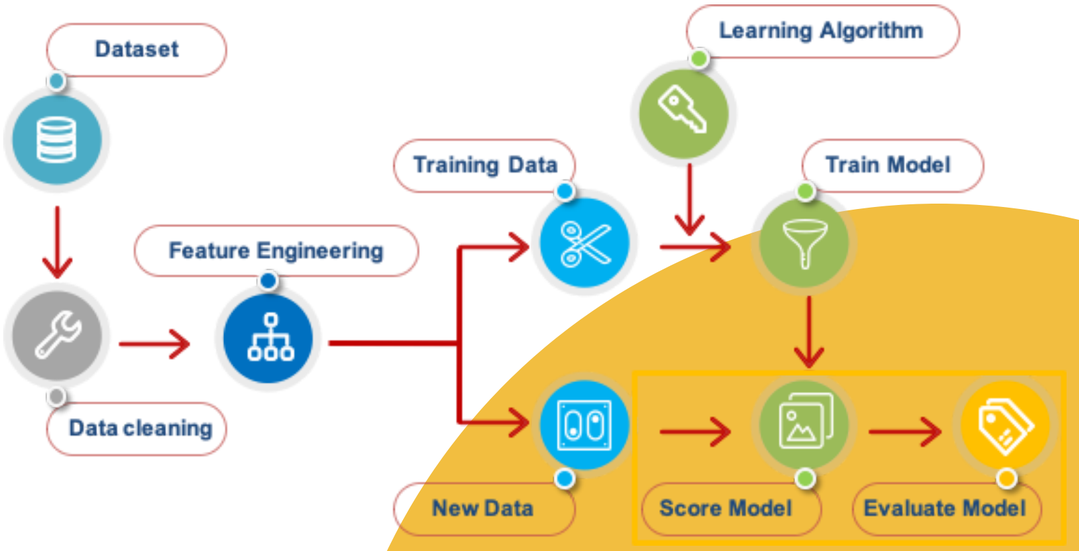
10

Labelled Clases

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

3

MODEL EVALUATION METHODS



DIFFERENCE BETWEEN THE TRAINING, TEST AND VALIDATION SET

~70%

TRAINING SET

Utilised to train the model

~20%

VALIDATION SET

Utilised when you tune your hyperparameters

~10%

TEST SET

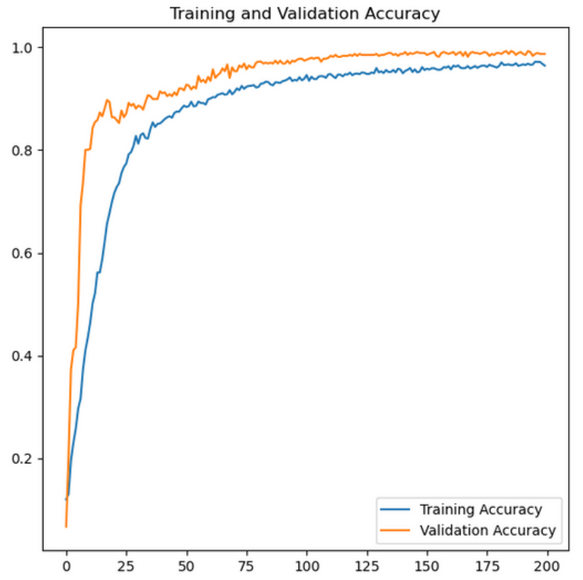
To score and report your model performance

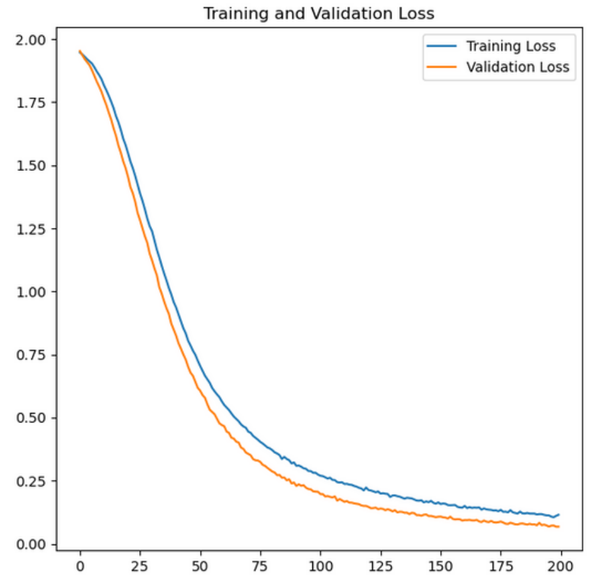
UNDERSTANDING THE DIFFERENT PERFORMANCE METRICS

CLASSIFICATION REPORT

	precision	recall	f1-score	support
Koi_Fish	1.00	0.99	0.99	100
Scattered_Light	0.99	1.00	1.00	100
Tomte	0.99	1.00	0.99	68
Whistle	1.00	0.97	0.98	100
Air_Compressor	1.00	1.00	1.00	32
Blip	0.95	1.00	0.98	100
Chirp	1.00	0.92	0.96	36
accuracy			0.99	536
macro avg	0.99	0.98	0.99	536
weighted avg	0.99	0.99	0.99	536

TRAINING AND VALIDATION CURVES





PERFORMANCE METRICS SUMMARY

01 **CONFUSION MATRIX (BINARY + MULTI-CLASS)**

02 **ACCURACY**
= (TP + TN) / (TP + TN + FP + FN)

03 **PRECISION**
= TP / (TP + FP)

04 **RECALL**
= TP / (TP + FN)

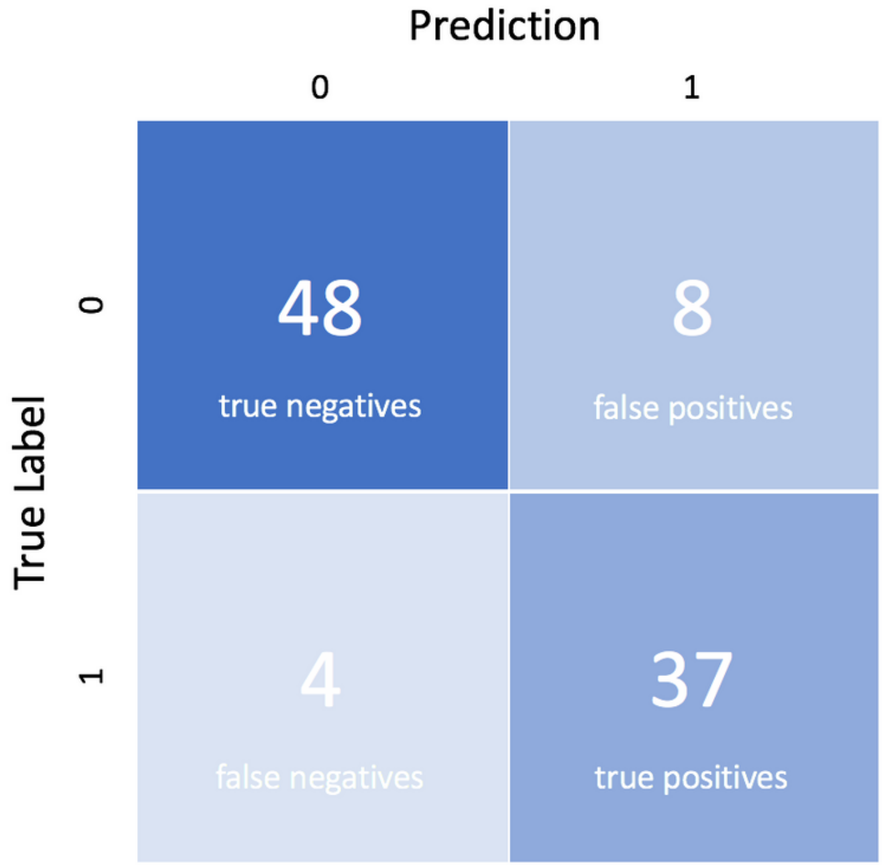
05 **SPECIFICITY**
= TN / (TN + FP) = 1 - FPR

06 **F1-SCORE**
= 2 * (PRECISION * RECALL) / (PRECISION + RECALL)

CONFUSION MATRIX

01

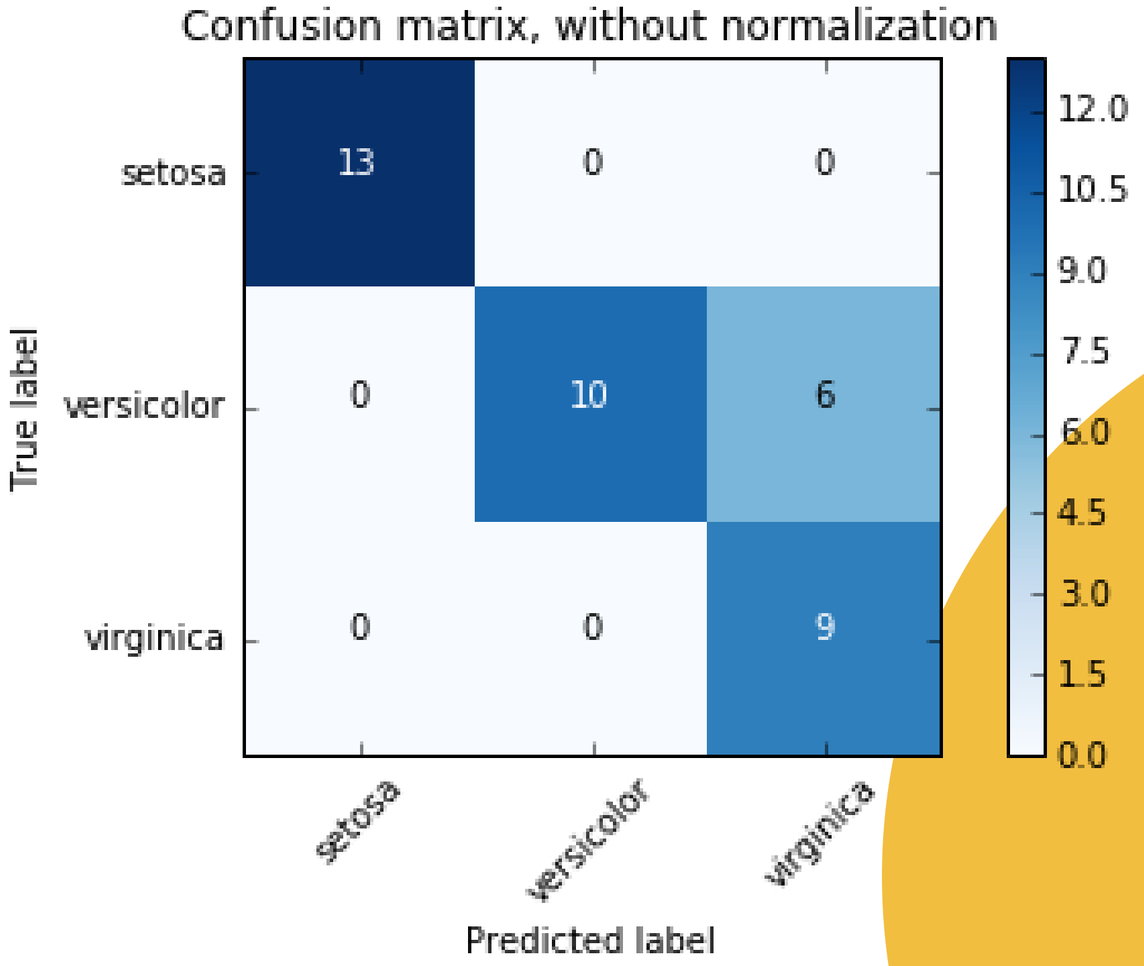
BINARY CLASS CONFUSION MATRIX



CONFUSION MATRIX

02

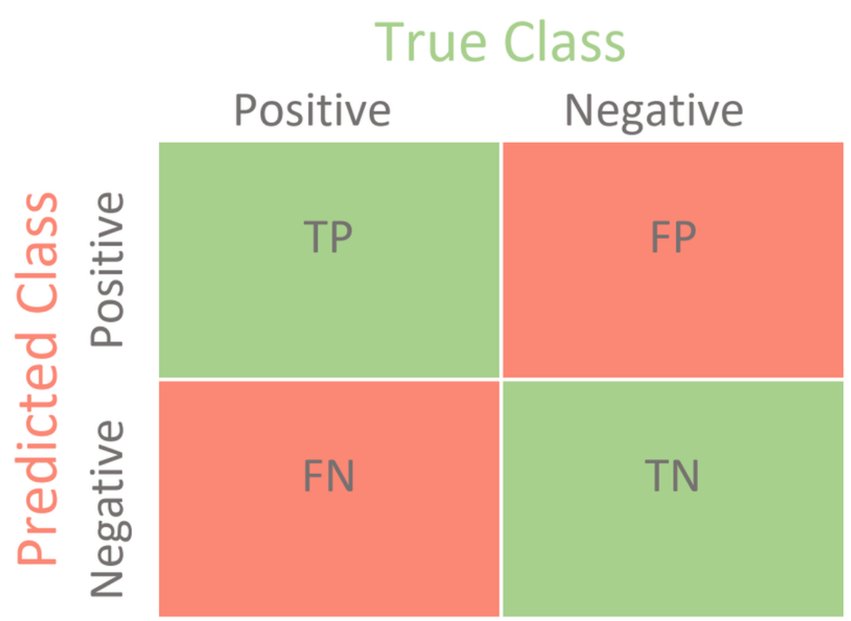
MULTI-CLASS CONFUSION MATRIX



ACCURACY

ACCURACY
= CORRECT PREDICTIONS / ALL PREDICTIONS
= (TP + TN) / (TP + TN + FP + FN)

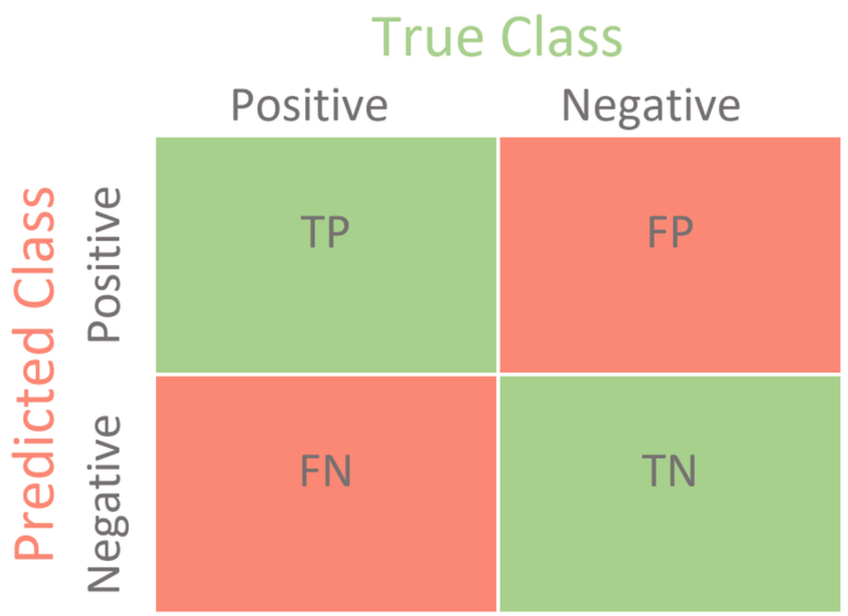
- PERCENTAGE OF CORRECT PREDICTIONS FOR THE TEST DATA



PRECISION

PRECISION
= NO. OF TP / TOTAL NO. OF POSITIVE PREDICTIONS
= TP / (TP + FP)

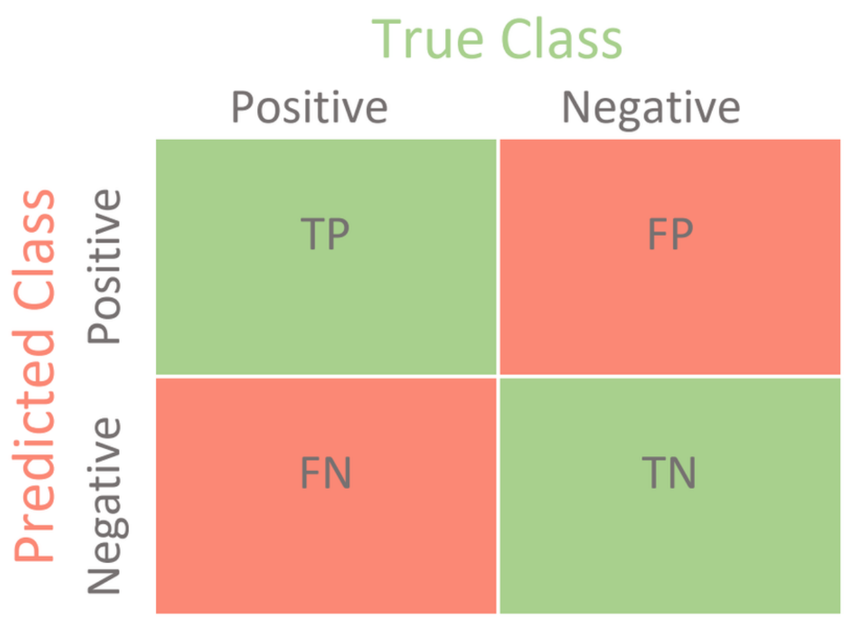
- TELLS US THE QUALITY OF A POSITIVE PREDICTION MADE BY THE MODEL
- ENSURES THAT WE DO NOT MISCLASSIFY TOO MANY PEOPLE AS POSITIVE WHEN THEY ARE NEGATIVE



RECALL

RECALL/ SENSITIVITY/ TRUE POSITIVE RATE (TPR)
= TP / (TP + FN)

- ENSURES THAT THE POSITIVE SAMPLES ARE ACCOUNTED FOR BY THE MODEL
- IMPORTANT TO EVALUATE PRECISION + RECALL

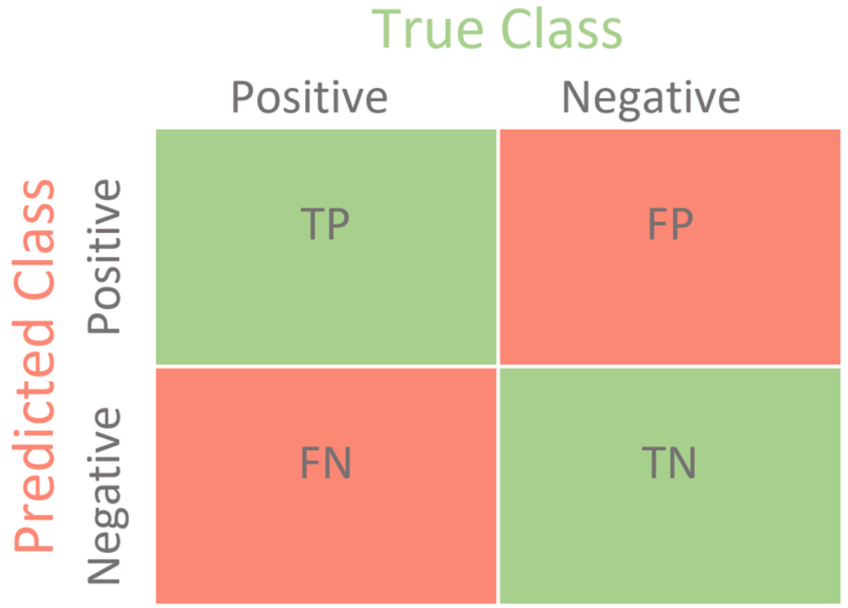


SPECIFICITY

SPECIFICITY/SELECTIVITY/ TRUE NEGATIVE RATE (TNR)
= ACTUAL NEGATIVES / PREDICTED NEGATIVES
= TN / (TN + FP) = 1 - FPR

- MEASURES HOW OFTEN THE MODEL GIVES A NEGATIVE RESULT WHEN SAMPLE DATA IS NEGATIVE

- LOW SPECIFICITY MAY MEAN HIGHER FPR
- MORE WRONGY PREDICTED AS POSITIVE



F1-SCORE

F1-SCORE/ F-SCORE/ F-MEASURE
= 2 * (PRECISION * RECALL) / (PRECISION + RECALL)

- ACCOUNTS FOR BOTH PRECISION AND RECALL
- USED WHEN WE ARE NOT SURE WHETHER PRECISION OR RECALL IS MORE IMPORTANT

BUT WHY IS MY MODEL PERFORMING POORLY?

OVERFITTING VS UNDERFITTING

WHAT?

- OCCURS WHEN YOUR MODEL FITS TOO CLOSELY TO YOUR TRAINING DATA

WHEN?

- NO. OF EPOCHS TOO HIGH
- SMALL TRAINING DATA SIZE, ETC

SO WHAT?

- HIGH TRAINING ACCURACY, BUT LOW TESTING ACCURACY

WHAT?

- OCCURS WHEN YOUR MODEL IS TOO SIMPLE

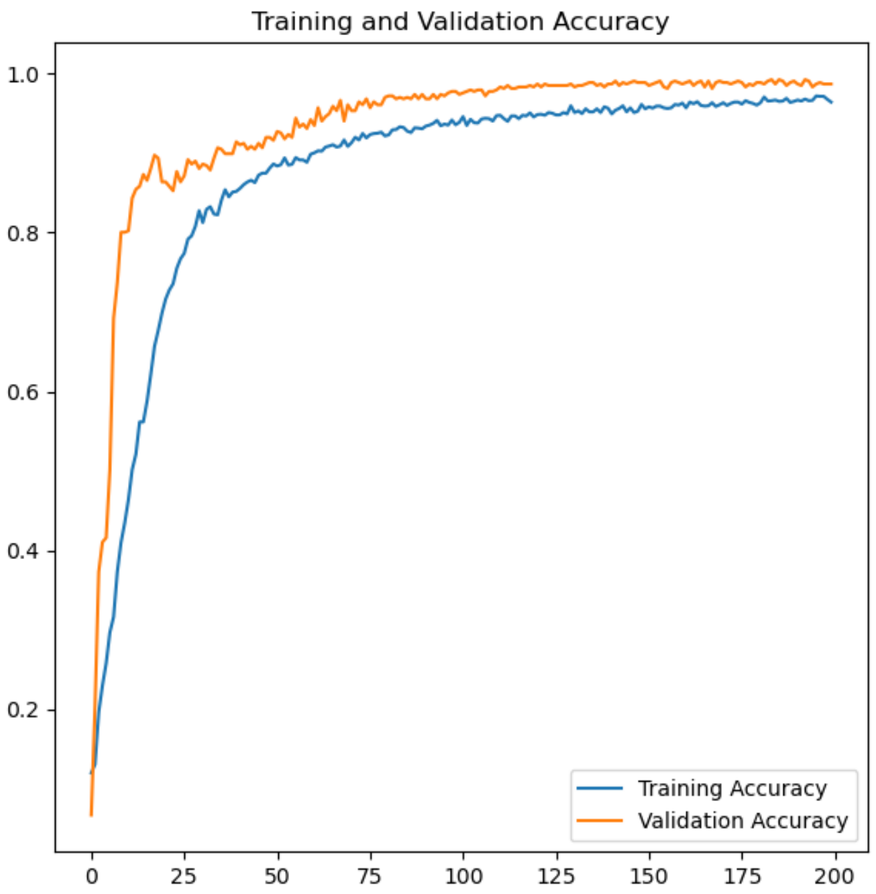
WHEN?

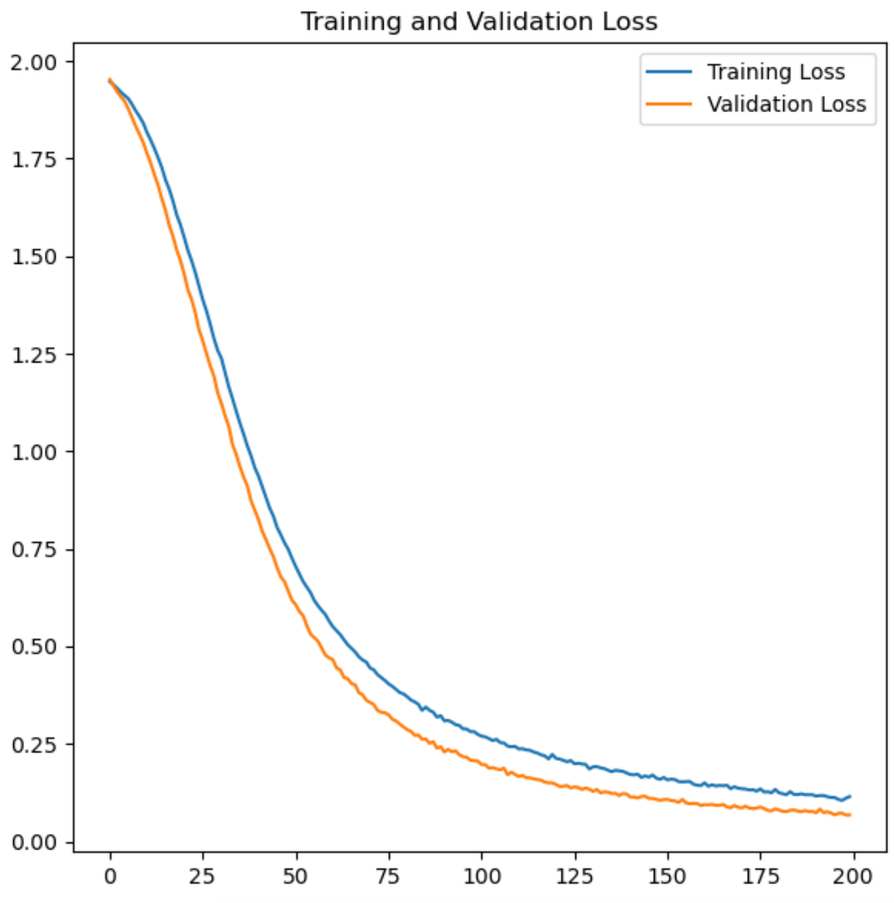
- NO. OF EPOCHS TOO LOW, ETC

SO WHAT?

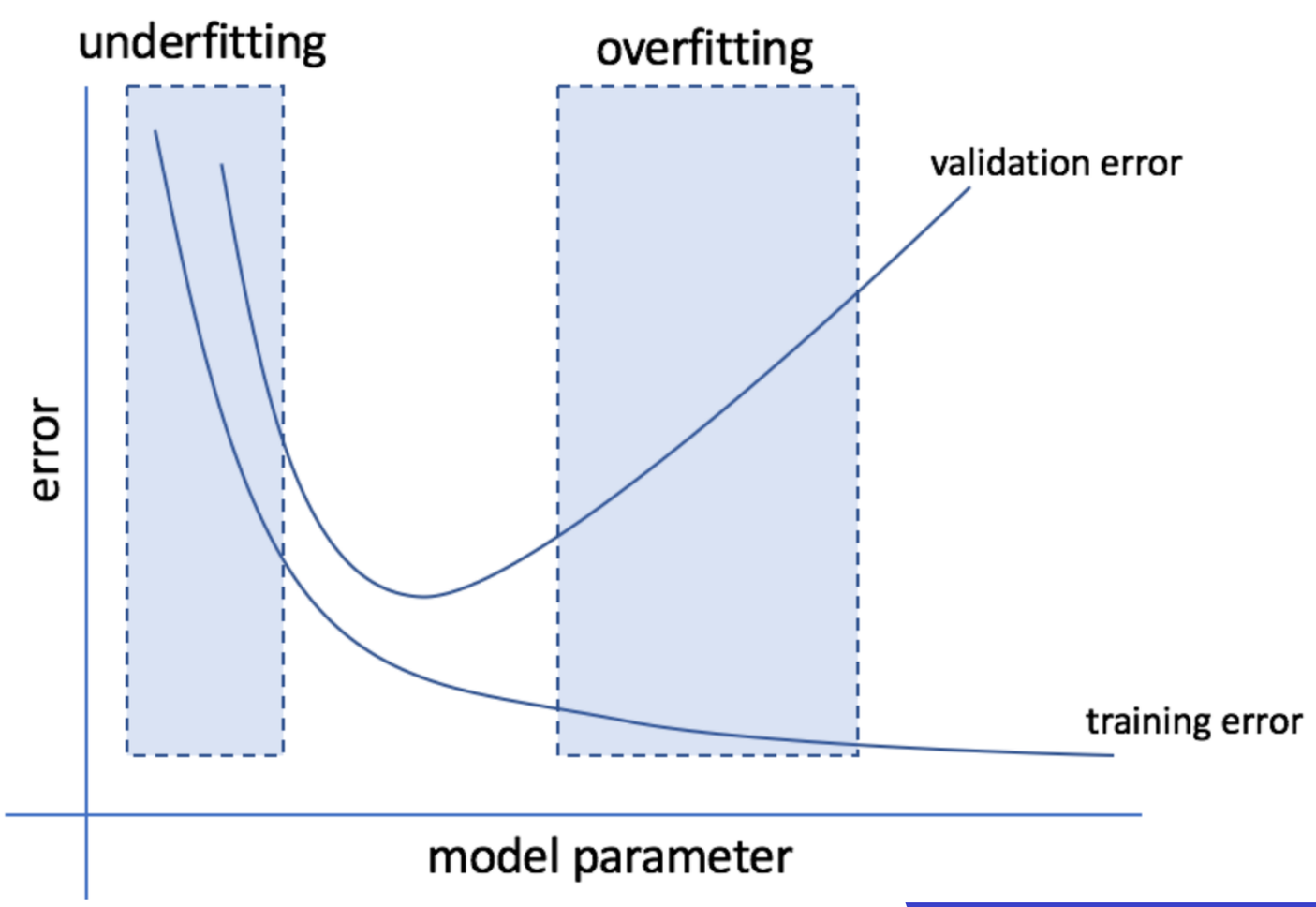
- HIGH ERROR RATE FOR TRAINING AND TESTING SET

UNDERSTANDING YOUR TRAINING AND VALIDATION CURVES





UNDERSTANDING YOUR TRAINING AND VALIDATION CURVES



SOME IMPORTANT DETAILS TO NOTE

01

ACCURACY ISN'T EVERYTHING

- Accuracy alone is not sufficient in evaluating the model
- Many other metrics e.g. precision, recall

02

EARLY STOPPING

- Helps prevent overfitting
- Stops the training process when the improvement stops increasing

03

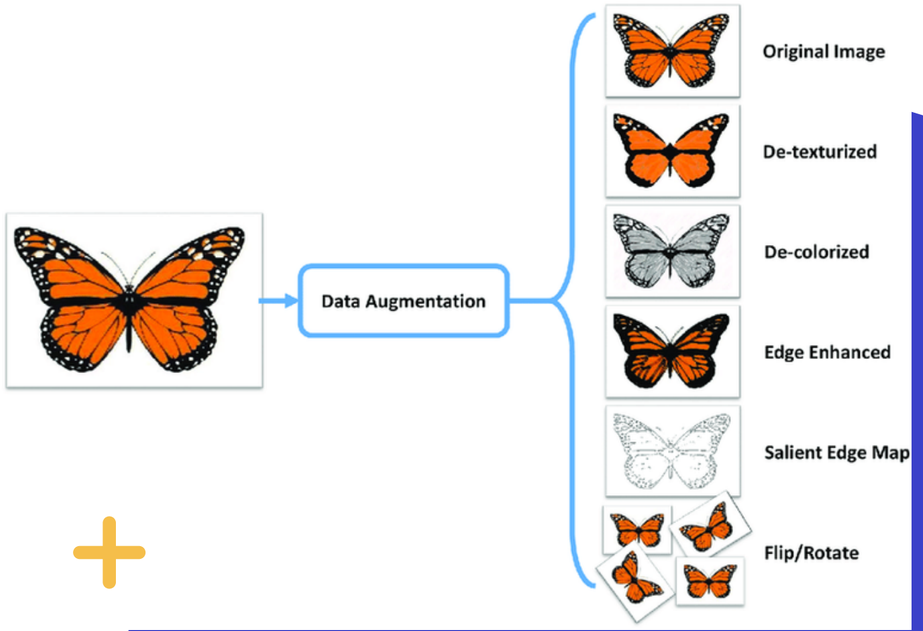
IMBALANCED DATASETS

- Can reduce it by increasing dataset size
- or via Data augmentation
- or via changing the class weights

04

DATA LEAKAGE

- Occurs when you normalise your data before splitting them into train, test and validation sets
- Artificially inflate your score

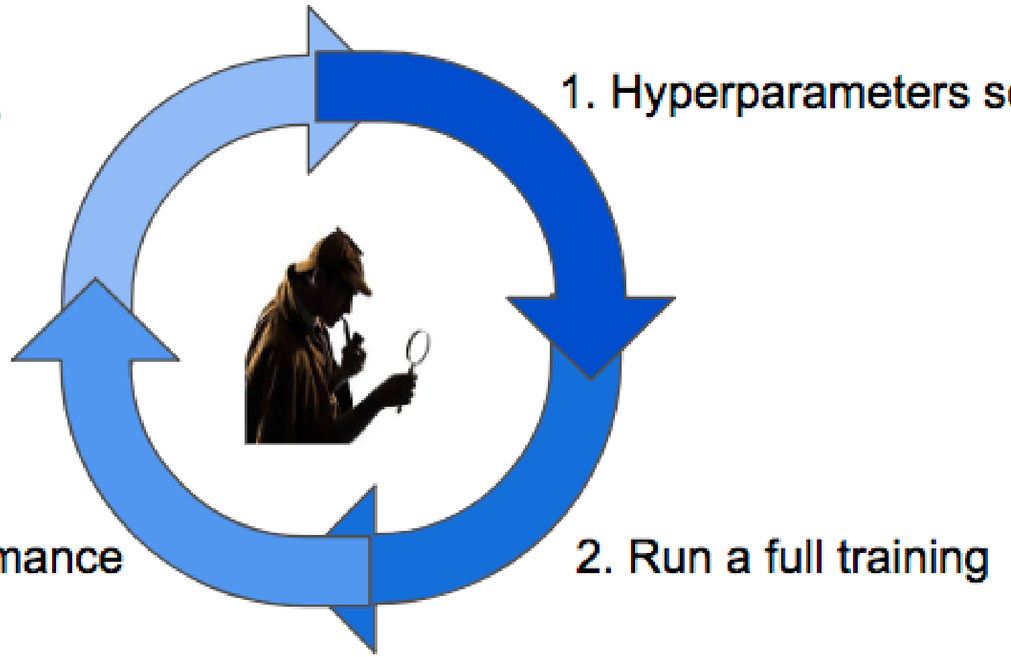


Search . . .

4

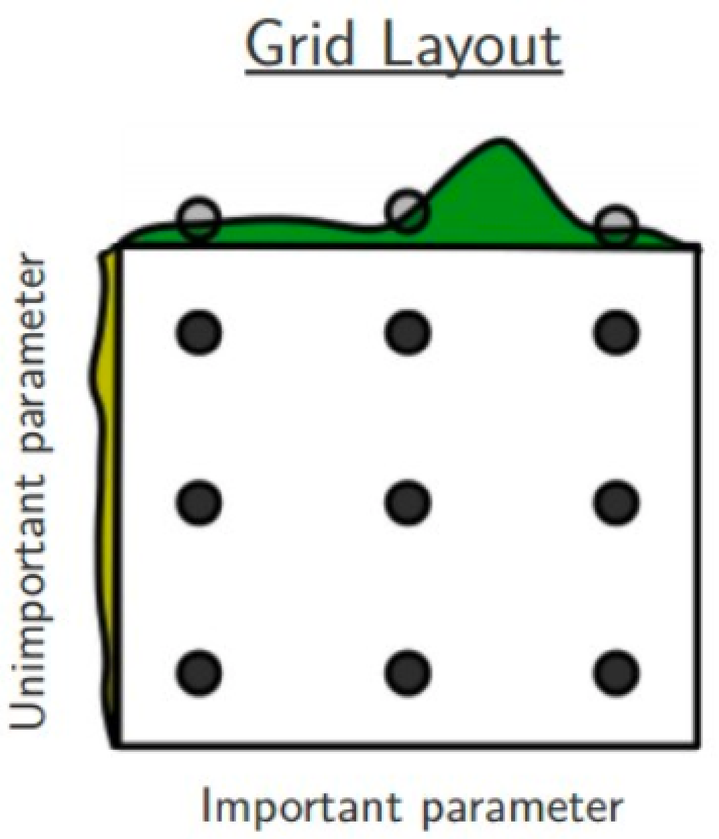
HYPERPARAMETER TUNING

(GRID AND RANDOM SEARCH)



WHAT IS GRID SEARCH?

- Define a grid on n dimensions, where each of these maps is for a hyperparameter. e.g. n = (learning_rate, dropout_rate, batch_size)
- For each dimension, define the range of possible values: e.g. batch_size = [4, 8, 16, 32, 64, 128, 256]
- Search for all the possible configurations and wait for the results to establish the best one: e.g. C1 = (0.1, 0.3, 4) -> acc = 92%, C2 = (0.1, 0.35, 4) -> acc = 92.3%, etc...



WHY USE RANDOM SEARCH INSTEAD?

Grid Search

(+) It will find the best parameters (very high cost)

(-) Can be time-consuming and computationally expensive for large parameter spaces.

(-) May result in overfitting to the training data if the grid search is too fine.

Random Search

(+)Can explore a broader range of hyperparameters more efficiently than grid search, especially in high-dimensional search spaces.

(+) Gives better results in less iterations

(-) Does not guarantee to find best hyperparameters

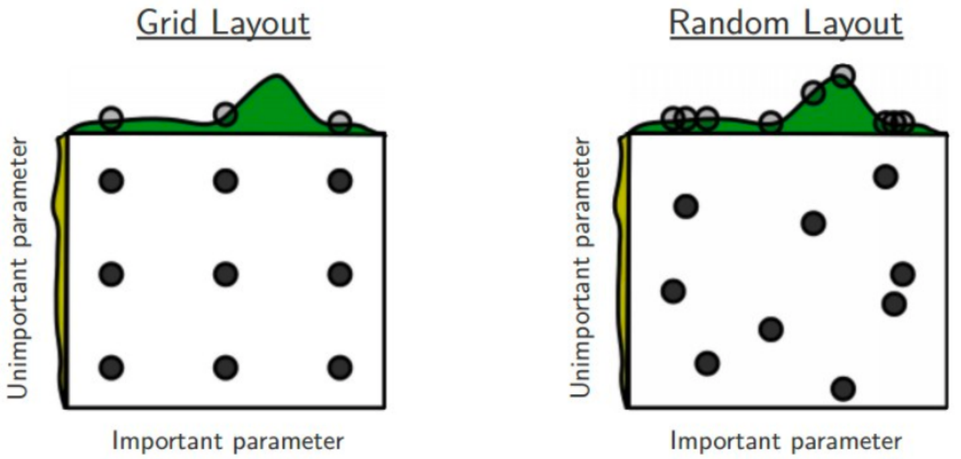


Image from http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

BAYESIAN OPTMIZATION

Imagine you are a researcher investigating mixtures of chemotherapeutic drugs for their ability to kill cancer cells.

C1

C2

C3

LIKELIHOOD
The probability of "B" being True, given "A" is True

PRIOR
The probability "A" being True. This is the knowledge.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

POSTERIOR
The probability of "A" being True, given "B" is True

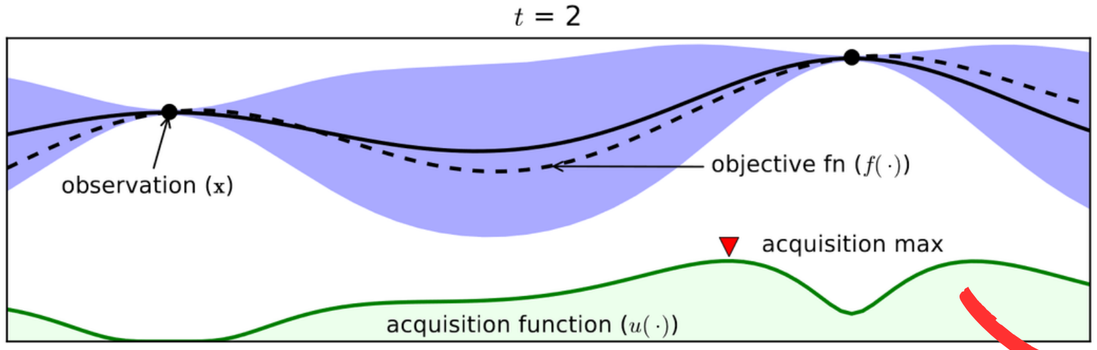
MARGINALIZATION
The probability "B" being True.

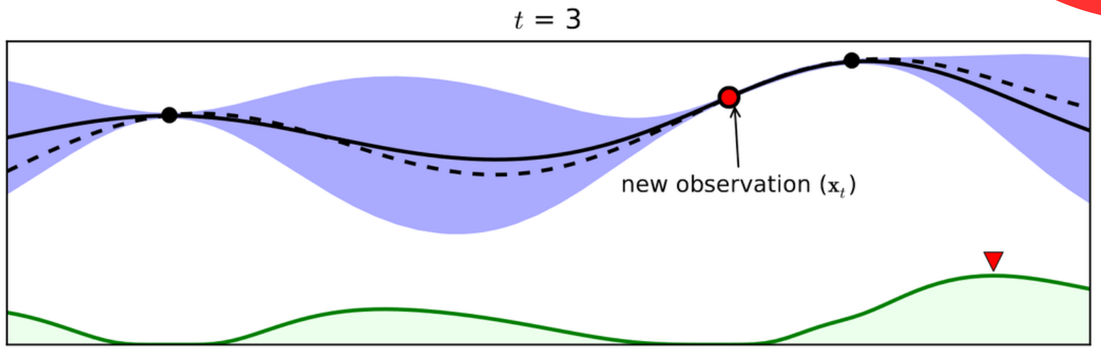
- Uses Bayes Theorem to direct the search in order to find the **minimum** or **maximum** of an objective function.
- **posterior** = **likelihood** * **prior** ≈ objective function
- Objective function is a function that takes a sample and returns a cost where a numeric score for a sample calculated via the objective function.

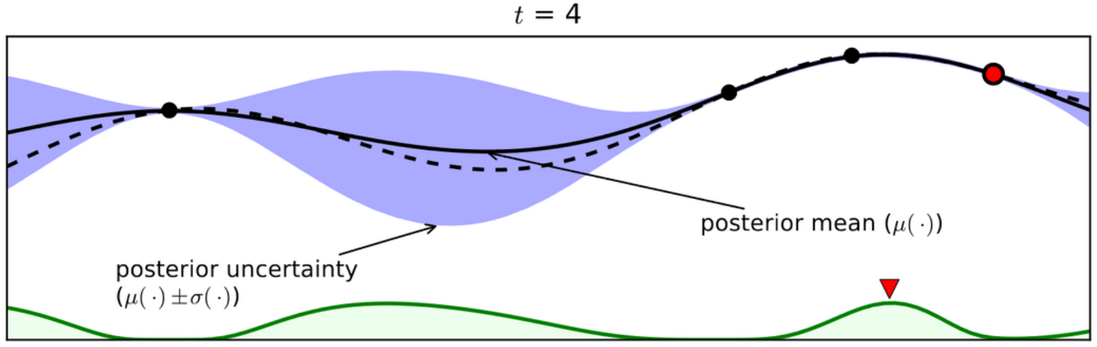
- Surrogate model approximates **f(x), objective function**
- Acquisition Function generates the next best guess of x

Black-box derivative-free non-convex optimization of expensive functions

EXAMPLE OF A 1-DIMENSIONAL OPTIMIZATION PROBLEM







The Bayesian Optimization algorithm can be summarized as follows:

1. Select a Sample by Optimizing the Acquisition Function.
2. Evaluate the Sample With the Objective Function.
3. Update the Data and, in turn, the Surrogate Function.
4. GoTo 1.

ADVANTAGES VS DISADVANTAGES

- Efficient use of resources
- Versatility
  - Can handle noisy or stochastic objective functions and non-convex search spaces.
- Can be combined with parallel computation to further speed up the optimization process such as GPUs and TPUs
- Does not require gradients or second-order derivatives of the objective function.

- Computationally expensive
- Selection of the acquisition function
- Sensitivity to the prior
  - Can be sensitive to the initial hyperparameter values and the quality of the data used to build the probabilistic model.
- May not guarantee finding the global optimum due to the stochastic nature of the algorithm and the model assumptions.

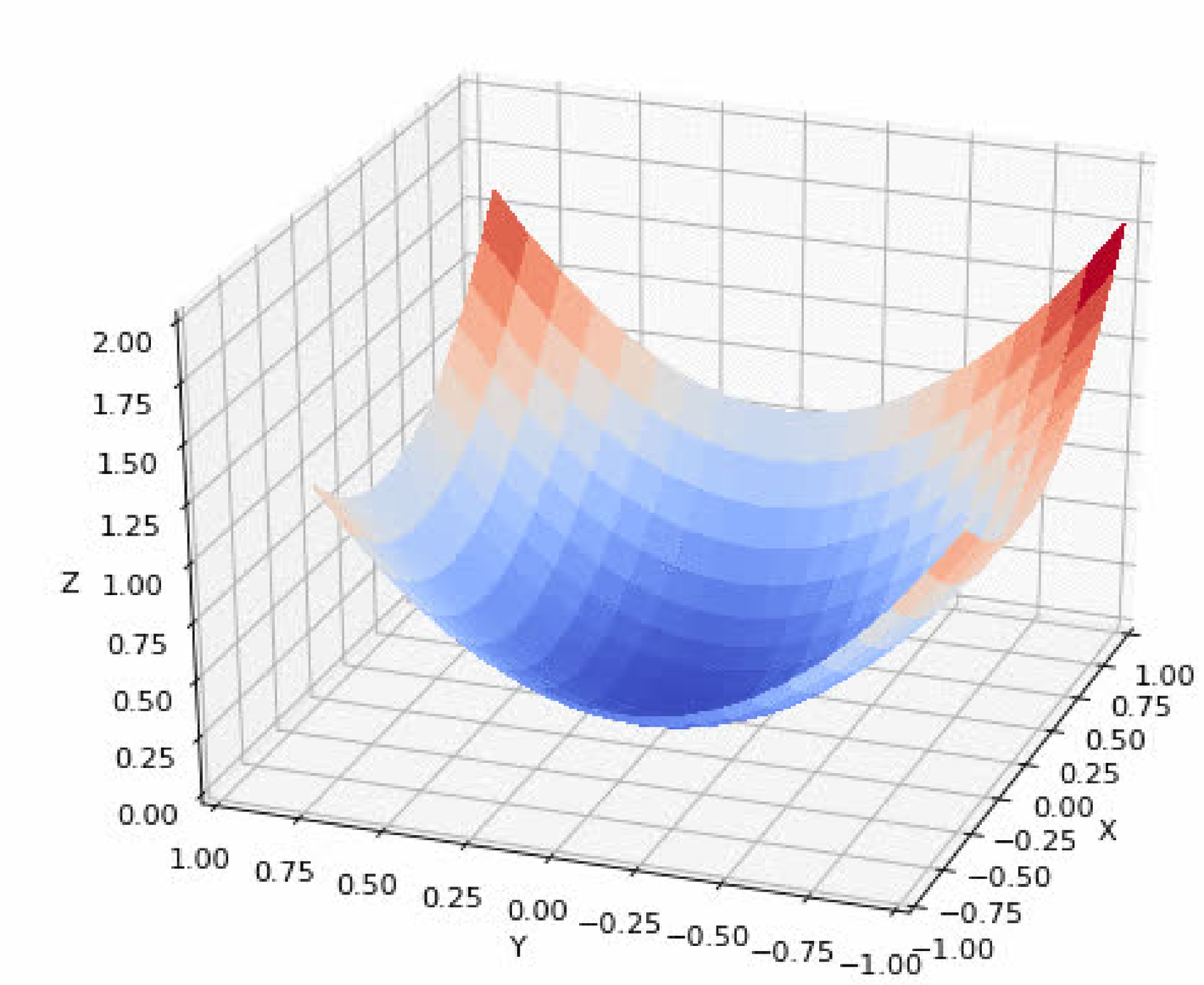
GRADIENT DESCENT ALGORITHM

- Finds the optimal parameters by minimising the error function (differs for different models)

Steps

- Start with a random value for parameters
- Fix a learning rate L to control the rate of change of values for your parameters
- Calculate the partial derivative of the error function with respect to your parameters to update your parameter values
- Repeat the step above until you reach convergence
- (When your error function is minimised to a small value or 0; or when the max number of iterations are reached)

GRADIENT DESCENT ALGORITHM



GRADIENT DESCENT ALGORITHM

```
#compilation of model
model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate', values=[1e-2, 1e-3]))
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
return model
```

```
model.compile(
    optimizer="adam",
    loss="mae",
)
```

ADVANTAGES & DISADVANTAGES OF GRADIENT DESCENT ALGORITHM

- LITTLE COMPUTATIONAL MEMORY
- MANY OTHER DERIVATIONS E.G. STOCHASTIC GRADIENT DESCENT, BATCH GRADIENT DESCENT

- ERROR FUNCTION MUST BE DIFFERENTIABLE AND CONVEX
- STATE OF CONVERGENCE MAY NOT BE THE BEST (LOCAL MINIMA)
- EXPLODING GRADIENT; GRADIENTS KEEP GETTING LARGER UNTIL THE ALGORITHM DIVERGES
- VANISHING GRADIENT: GRADIENTS KEEP GETTING SMALLER UNTIL THE PARAMETER VALUES ARE UNCHANGED

THANK YOU & Q-A