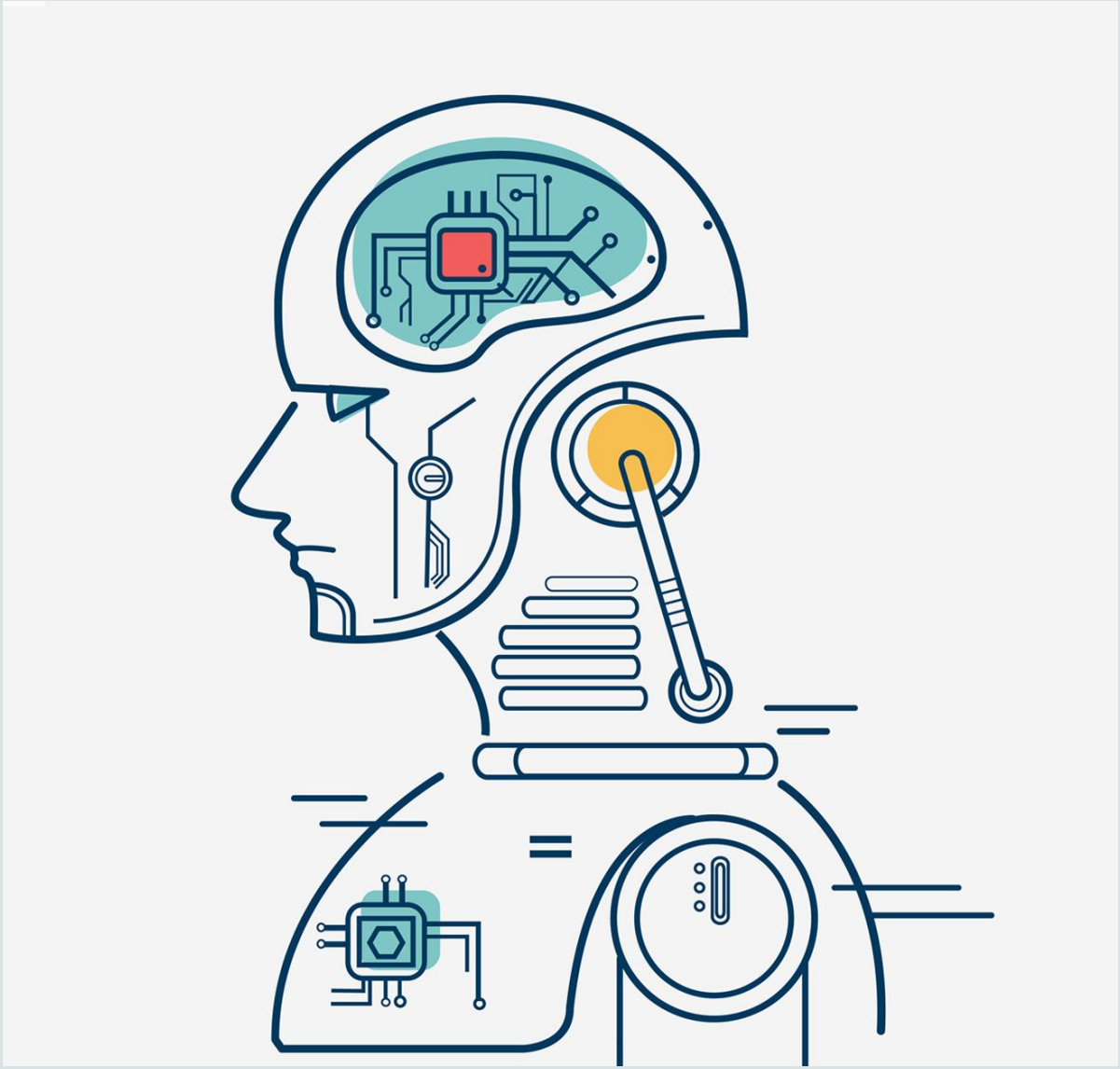
Hands on Session

Ruchira and Rahul

Agenda

- Introduction
- Preprocessing
- Splitting the dataset
- Modelling
- Optimizing the model



Introduction

The Dataset

So today we will be using Pokemon dataset from Kaggle and we will be classifying the images of the pokemon into 5 distinct classes, which are: -

1. Mewto
2. Charmander
3. Pikachu
4. Bulbasaur
5. Squirtle

Hence, by the end of this presentation we will have successfully developed a model which can identify whether the given image of the Pokemon belongs to any of the formentioned classes or doesn't

How will we achieve the task?

For the purposes of our project we will be using the PyTorch library which provides us with ample of tools to create the best possible ML model. First, we will be using PyTorch to create a custom model and check its accuracy and then with the help of transfer learning we will use a pre-trained instance of Resnet to achieve the same task and see if the accuracy of the predictions improve or not.

Preprocessing

Preprocessing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Preprocessing in image data

The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task. This could include increasing the brightness, reducing contrast and pretty much anything which makes our image easier to analyze. In our project we have used Pytorch's transforms class to resize all images to a common dimension and then convert them into tensors

Image Augmentation

Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc. Image Augmentation helps us in making our dataset resemble a real world dataset. For example, in our Pokemon dataset, when we get the images from a real world source, its not guaranteed that the data will always be consistent or be perfect, hence we add augmentations in order to enable our model to learn to identify any kind of images.

Splitting the dataset

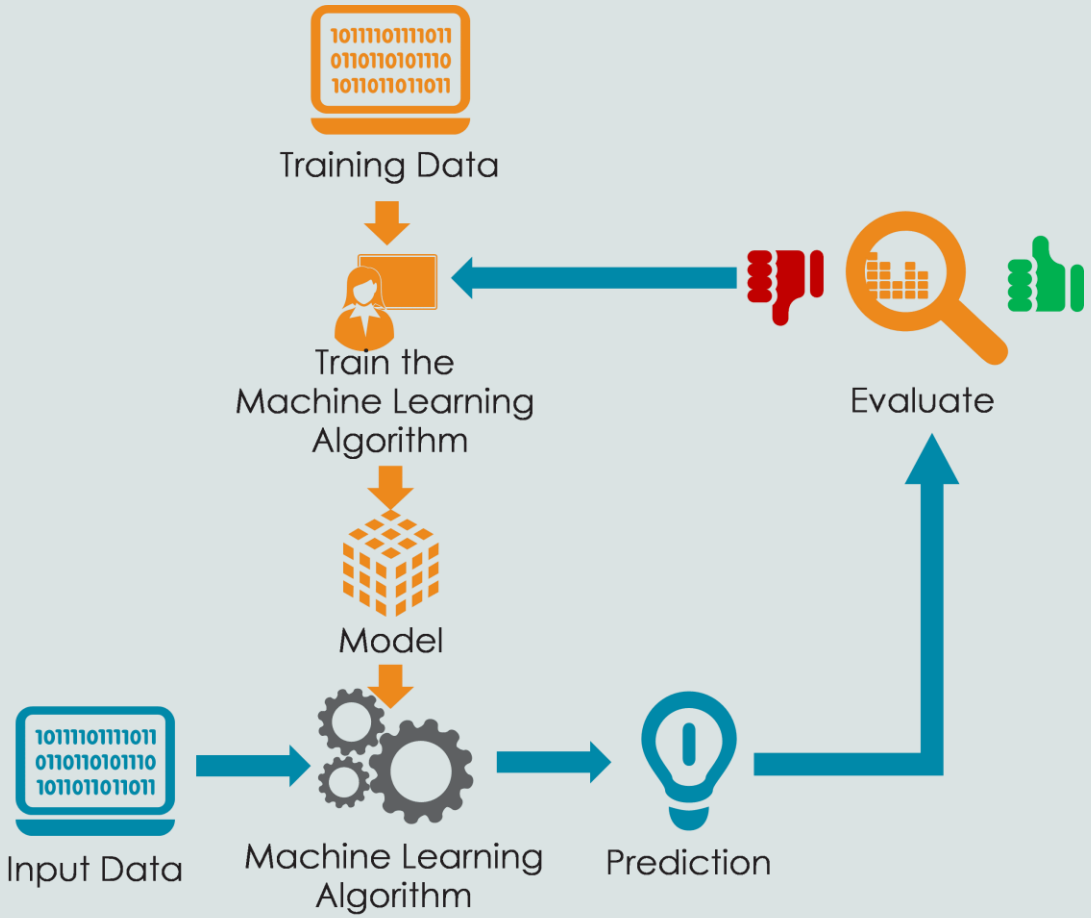
Why are we splitting the dataset?

Every time we build a ML model, we will split the dataset into Training, Validating and Testing data. We train our model on the training set, and then use the Validation dataset to measure the accuracy and the performance of our model. We will iteratively add and remove the layers, change the parameters and hyper parameters till we achieve a good accuracy and performance on the validation set. One important thing to remember here is that during this entire process we never introduce the test set to the model, to avoid any kind of output leakages. We have split the data into 85% and 15% ratio.

Modelling

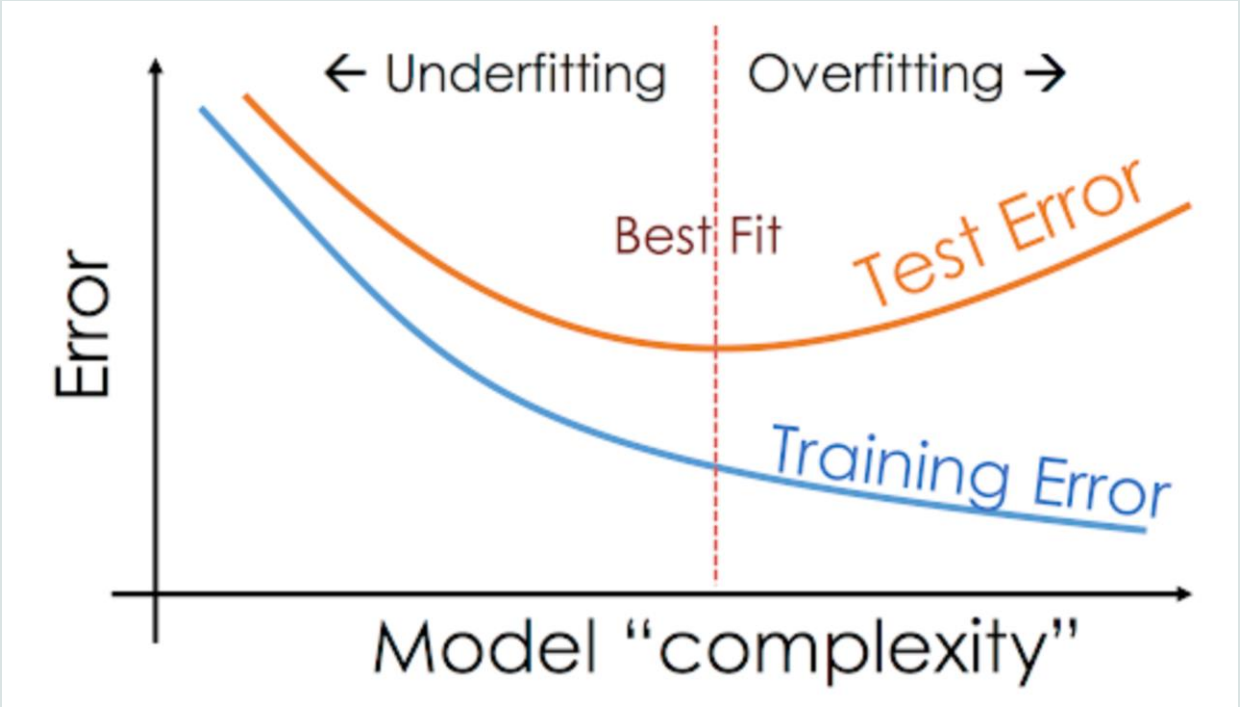
Modelling in Machine Learning

The objective of machine learning is not a model that does well on training data, but one that demonstrates it satisfies the business need and can be deployed on live data. Modelling in Machine Learning is an iterative process where we build one baseline model, check its accuracy and performance, then iteratively we tweak the parameters and hyperparameters to find the best model.



Overfitting vs Underfitting

One very important to remember while finding the best model for your problem statement is to keep plotting the accuracy and error graphs so that we can understand if our model is overfitting or underfitting. Overfitting is a condition where our model is performing really good on the training set but is unable to perform well in the test set, while underfitting is a condition where the model is unable to perform even on the training set.



How to determine which model is better?

This indeed is a very good question. Neural networks were designed to ‘learn’ on their own, find patterns, features and etc. by running iteratively over and over again. Each cycle is called an epoch, and after each epoch the model see’s how it performs and then tweaks its own parameters. Choosing a good metrics is just as important as choosing the right model, using a bad metric can lead to us choosing a overfitted model or a model which provides erroneous results.

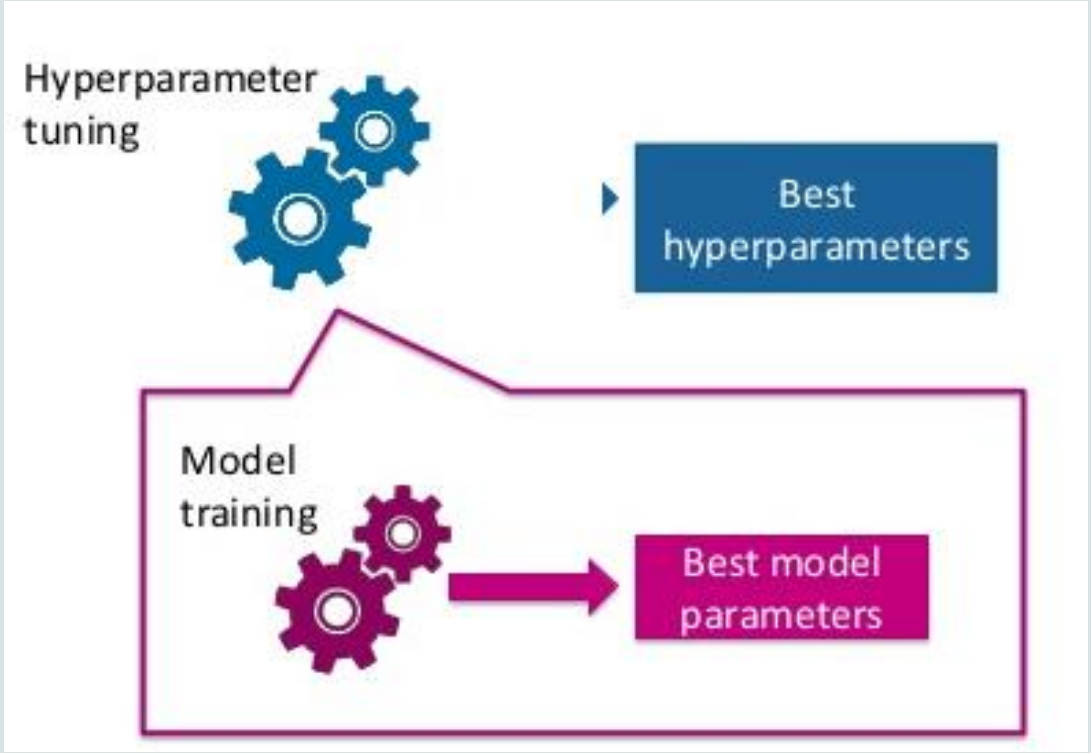
The metric used in the project

In our project in order to measure the performance of our model we use Cross Entropy Loss to find the performance of the model at each epoch. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.

Optimizing the model

How to optimize your model for best performance?

There are mainly two types of parameters, model parameters and the hyperparameters. model parameters are estimated from data automatically and model hyperparameters are set manually and are used in processes to help estimate model parameters. Hence, in order to achieve the best results and to reduce the number of iterations we need to run to achieve the perfect model, we must do Hyper Parameter tuning.



Optimizers

How you should change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use. Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible. The optimizer we use in the project is called Stochastic gradient descent. It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in Gradient Descent.

Learning Rate

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

The challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model.