**Chapter 4**

**Problem 4­­: Surface Defects**

All materials have some crystalline structures. But this structure is not in the same orientation in all over the materials. Each region have it’s own orientation until they osculate each other, the line of coincidence is one of the kinds of surface defects. Surface defects are really important in mechanical engineering, because in spite of tininess they can grow till failure during machining or other mechanical processes.

Inspection of defects and removing before resuming process could be helpful. Machining processes usually works by highspeed cutting tools and during these processes some cutting materials vented so inspecting these defects by human is dangerous and difficult. A machine with camera can detect these defects and call the operator to avoid further damages.

**About Dataset**

This dataset contains 10 thousand images included 2 classes, Stars and Galaxies. Each image in dataset contains 512 pixels as height, 512 pixels as width and 3 color channels and pixel values are between 0 and 255. There are 8000 images set aside as training data and 2000 as validation and half of each dataset belonged to each class respectively.

**Introduction**

Unlike typical computer programs, Machine Learning techniques will literally learn from data. Machine Learning algorithms can actually find insights and data even if they are specifically instructed on what to look for in that data, and that's what separates a Machine Learning algorithm from a typical computer program. You're just giving the Machine Learning algorithm a set of rules to follow. Instead of actually telling it what to look for, it will find the insights on its own.

**Why do we use Machine Learning to solve mechanical problems?**

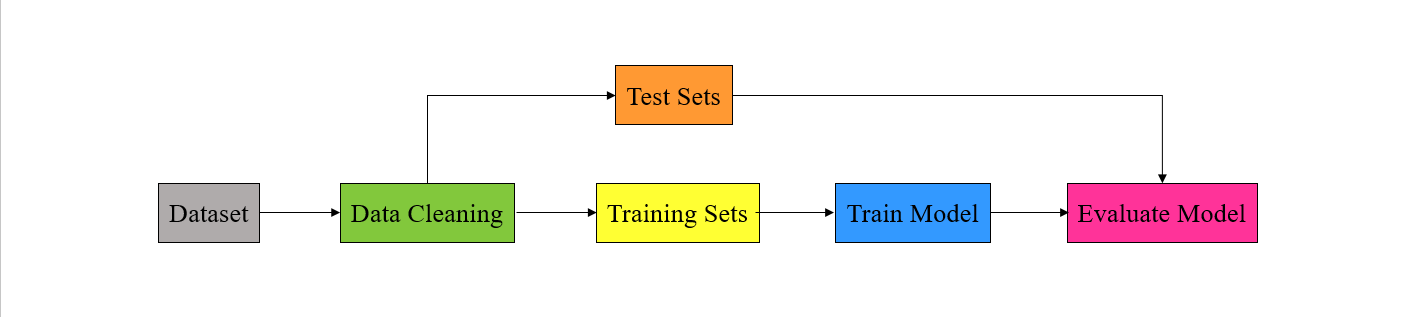
Machine learning is a method for predicting behavior or classifying data sets that, unlike common methods in mechanics, instead of being based on an intuitive model, uses a mathematical model and arbitrary functions to describe and predict the behavior of systems. In other words, machine learning is a search in the space of algorithms and parameters in such a way that it infers a model from the data (data-driven model) and based on that, predicts or categorizes the studied system. For example, using neural networks as one of the methods of traditional machines, I can perform a set of inputs based on an arbitrary number of intermediate hidden layers to the output image results. In the input and output data that are entered quantities, the middle layers do not necessarily have meanings and other adverbial expressions on them. For this reason, I can choose the number of intermediate layers and the number of nodes in each layer at will, and this approach is completely acceptable in the input to the output image. In particular, the relationship between the data is so complex that the models created with a limited number of adjustable settings express this relationship with sufficient accuracy, the efficiency of the methods using machines can be very important.

**Classification Problems**

High-level machine learning consists of supervised and unsupervised learning. Supervised learning means that we label historical data and use it to inform our model. We call that label or something that we want to predict as a target. So, in supervised learning, we have a specific goal (target) for that past information, and in unsupervised learning, we don't have a specific goal. In supervised learning, we have classification and regression. Classification problems are problems where our goal is a category (that is, we want to see what category it belongs to. It's usually True or False, but it can be multiple categories). Regression problems are those where our target is a numerical value.

**What are we going to do in this Chapter?**

We have a dataset from the Kaggle website and then clean that data. After that, we split our data into two groups (train and test). Then we train our model on the training set and after that, we evaluate our model with the test set we have.

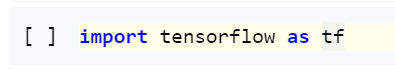


***Figure 3-3.***

**Tensorflow**

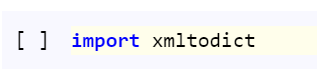
Tensorflow is an open-source library for machine learning and numerical computation based on tensors. Tensorflow can be developed even by who has a least knowledge of programming alse can be used in popular programming languages like Python, Javascript and C++.

The main application of Tensorflow is building neural networks like CNN and RNN, and because Tensorflow works based on graphs, it is much easier and faster to execute on multiple processors like GPUs.



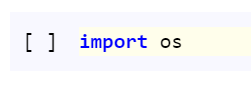
**xmltodict**

XML (Extensible Markup Language) is a type of format of saving data, its foundation is very likely to HTML and it is easy to extract and analyze. This type of files needed to be converted to python data structures for extracting data. Dictionary is suitable kind of data structure because it has key-value pair like XML file. To convert XML file to dictionary, we use **xmltodict** library.



**OS**

OS module in python contains some libraries and functions to access through underlying operating system to manipulate directories, create and change directories, access files in directories and etc. It’s very important when you are dealing with files like reading or writing files.

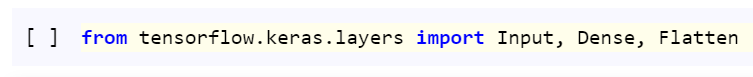


**Keras layers**

Keras models whether sequential or functional, need layers to be defined. Every layer has variant parameters must be defined. Layers that used in this notebook are:

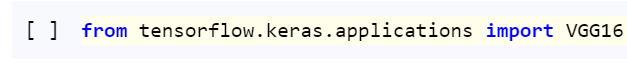
**Input:** this layer is input layer, as its name suggest, must be the first layer that receive input. There is only one parameter and it is input shape that indicates shape of input tensors except batch size.   
**Flatten:** this layer converts every multidimensional tensor to one dimensional tensor.

**Dense:** this layer indicates the simple neural network architecture with neurons fully connected to each other. Two parameters of this layer must be set, number of neurons and activation function must be defined.



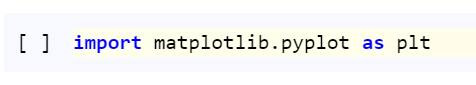
**VGG16**

Tensorflow provides some methods as classic models. These methods have classic model’s architecture and tuned weights, so it can be used for transfer learning. One of key models in image recognition is Inception V3. This model uses repetitive blocks include convolutions with different kernel size, max pooling, and concatenations named inception blocks.



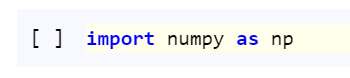
**Matplotlib**

Matplotlib is a library package for generating static and dynamic visualization in Python. Pyplot is a Matplotlib module that creates figure in a separated window and make changes to this figure to plot.



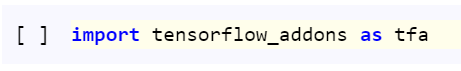
**Numpy**

Numpy is a library provides functions dealing with arrays. Numpy arrays have advantages that makes them usefull, have been processed faster than lists, less memory consumption and static memory place.



**Tensorflow Addons**

Tensorflow\_addons is a library that provides extra helper functions for tensorflow and keras models include additional layers, optimizers, losses, metrics and etc.



**Step 1. Preprocessing**

At first data in machine learning are not usable often, so needed to be cleaned like filling missed values, separating features and labels, normalizing or scaling and so on. These kinds of processes named preprocessing to prepare data for feed into the model.



**Normalization of Data**

Numeric values in machine learning might have vary in terms of range, for faster and more stable training, these values must have the same range like between 0 and 1. This kind of processing called normalizing because all values scaled between 0 and 1. It’s equation like below:

There are pixel values as features in this problem. These pixel values are 8 bytes values so there are in range between 0 and 255, as the equation above suggests, pixel values must be divided by 255 to have range between 0 and 1 but before returning values, result of division must be casted as float to avoid further tensorflow errors (cast in tensorflow means changing value types).

As it’s shown in the table below, there is some random data between 0 and 10, to normalize this data, subtract each number from minimum (0) and divide by data range (10 – 0).

|  |  |  |
| --- | --- | --- |
| *X* |  |  |
| 4 | (4 - 0)/(10 - 0) | 0.4 |
| 10 | (10 - 0)/(10 - 0) | 1 |
| 0 | (0 - 0)/(10 - 0) | 0 |
| 5 | (5 - 0)/(10 - 0) | 0.5 |
| 7 | (7 - 0)/(10 - 0) | 0.7 |

**Why Do We Need to Normalize Data in Python?**

One of the most important topics in the field of Machine Learning and Data Mining, especially in the Data Preparation section, is the topic of Re-scaling of data, which is usually done by There are two methods of Standardization and Normalization. The meaning of normalization is to transform the data into the domain [0 and 1]. Each of the data recorded in the dataset will change to a range between zero and one. This makes the data fall under a shorter domain and the model is trained better.

**Step 2. Load Image**

This Function takes image name and folder to read image file and save it as eager tensor. At first create image path from image directory, subsequent folder and name then read image path by tensorflow package and save it as a file. Convert file to eager by decoding jpg file into eager tensor that has 3 channel.

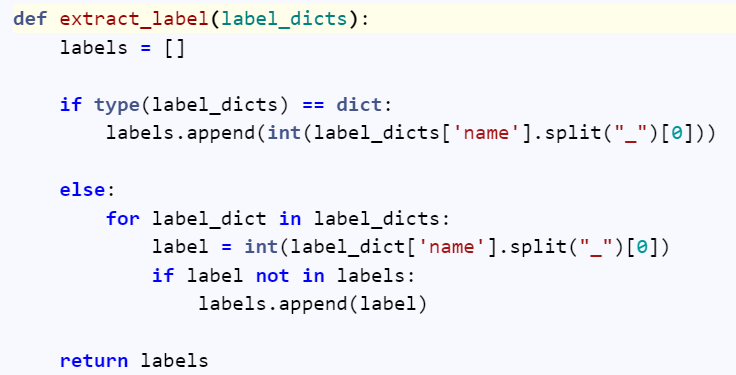
To feed this image to model, image tensor size must have specific dimensions as we are going to use VGG16 it is recommended to resize image to 224 by 224 as width and height of image.

To normalize values, process function, that already have defined, must be mapped through every pixel value to scale between 0 and 1.



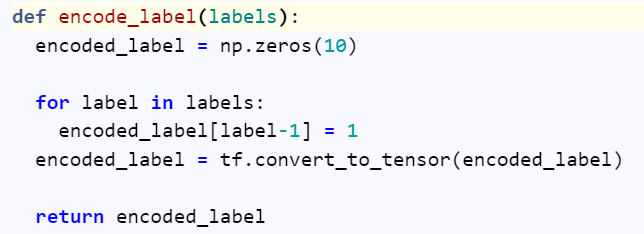
**Step 3. Extract Labels**

This Function takes a dictionary as input and checks whether this dictionary has only one label or not, then append label(s) to the list of labels as string type.



**Step 4. Encode Data**

This function takes list of string labels as input and create label vector of ones and zeros as map each string label to a number from 0 to 9 and which labels appear in the labels list, insert one in subsequent position in the vector and insert zero elsewhere.



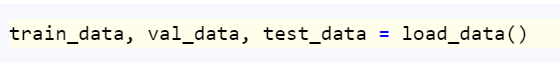
**Step 5. Load Data**

This function takes xml files directory and validation and test split ratio as input. It gets all xml files in directory and store it in list. After reading subsequent xml files and convert it to dictionary, extract image folder, name and label. This information is sufficient for extract and encode label and load image by previously defined functions. Dataset has been created from extracted images and labels. After shuffling, splitting and batching dataset, three main datasets as training, validation and test returned as function output.



**Step 6. Split Training and Testing Datasets**

All models in machine learning needs data to be trained on, changing parameters to fit on available data. This means the more data lead to better results. The model compute training data loss and then by computing gradient descent, tries to decrease loss value. But is decreasing training data loss mean better results on real data? Not necessarily. Real data often include the data that model has not previously seen before, so for evaluating results, loss value on training data is not enough and some data like validation needed to be collected.

****

**Batch**

Computing loss on whole data is much expensive, so computing loss on a small part of data and calculating gradient descent based on that is much more computationally efficient. This part of data to calculate loss once a time called Batch and numbers of data in each batch called Batch size. Batch size is the first element of shape in tensorflow and save as **None** object because it is independent from model architecture.

**Step 7. Training**

**Custom Loss**

Images in this dataset may contain more than one defect, so this is multi label problem. Categorical Cross Entropy loss function compute loss based on logarithmic difference between ground truth label and prediction which comes out from probability distribution function like Soft-max. In this case, summation of all probabilities must be equal to one and changing in probability of one class can affect the probability of other classes.

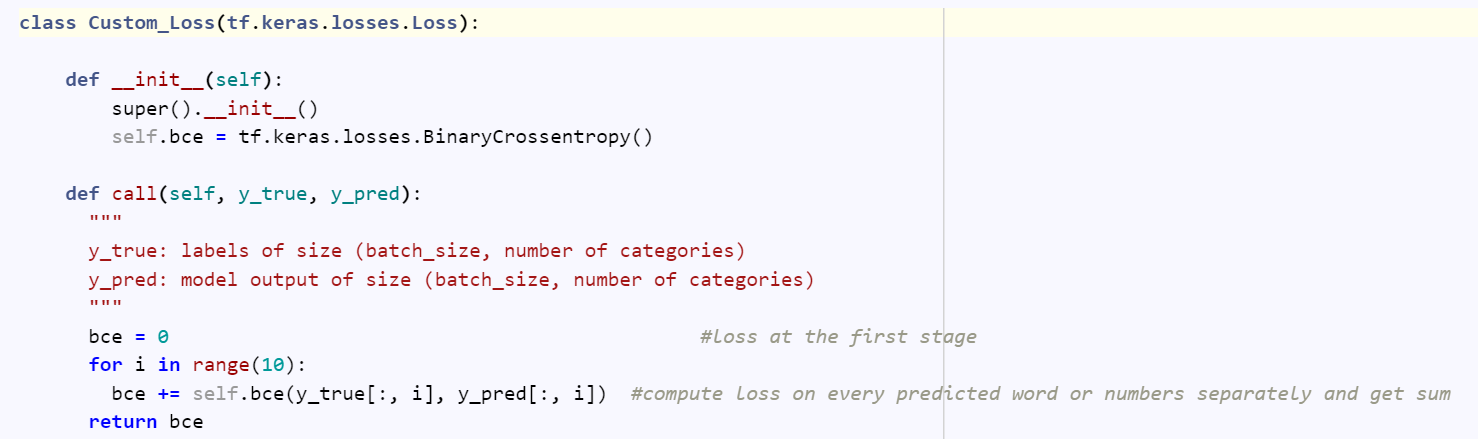
This problem discussing about multi label multi class classification that the summation of probabilities over different classes can extend from 1 and each class have separate probability distribution so we need to use Sigmoid as last layer activation function and compute Binary Cross Entropy on each class separately and sum over all classes.

**Binary Cross Entropy**

Binary cross entropy is a loss function that compute logarithmic loss of data as mentioned below.

As it is clear in binary classification, there is only 2 classes 0 and 1. this loss function compute large loss every time there is difference between true label (y) and prediction (p(y)):

If y is equal to 0, first term of loss become 0 and second part depends on prediction. If prediction is near 0, logarithmic term become near 0 and total loss would be near 0, but if prediction is near 1, logarithmic term become near –infinity and total loss goes to infinity and vice versa if y is equal to 1. at last Binary cross entropy takes averages across all losses.

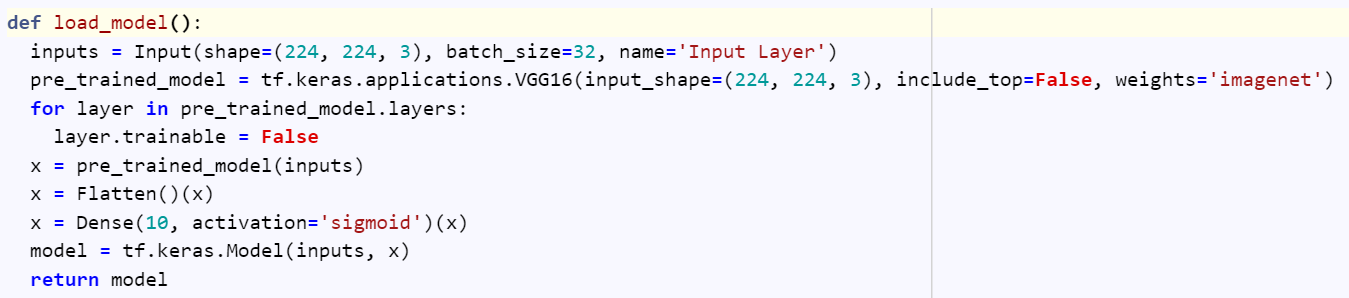


**Create Model**

This Function create neural network architecture. First layer of the network is Input layer then save VGG16 architecture with weights tuned on image\_net dataset and last layer that make classify images, must be ignored so include\_top parameter set to False. This vgg model must have input shape equal to dataset images so input\_shape parameter set to (224, 224, 3).

After vgg model, Flatten layer convert multidimensional data (vgg model output) to one dimensional. For the last layers, Dense layers has been used to classify images. Because of multi class classification problem, there is 10 (total number of classes) neurons in last layer needed and activation function must set to sigmoid (as previously mentioned).

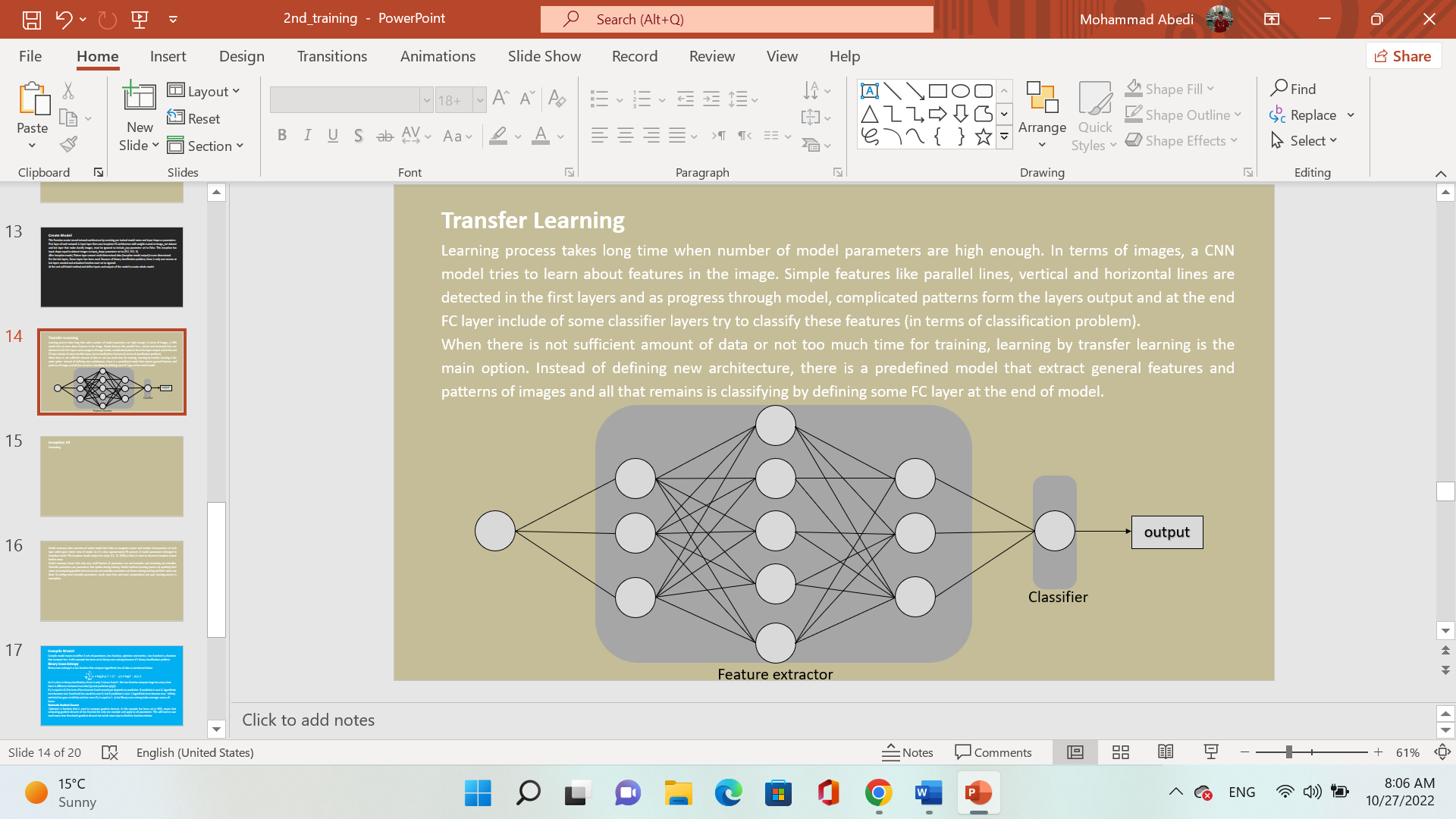
At the end call Model method and define inputs and outputs of the model to create model instance.



**Transfer Learning**

Learning process takes long time when number of model parameters are high enough. In terms of images, a CNN model tries to learn about features in the image. Simple features like parallel lines, vertical and horizontal lines are detected in the first layers and as progress through model, complicated patterns form the layers output and at the end FC layer include of some classifier layers try to classify these features (in terms of classification problem).

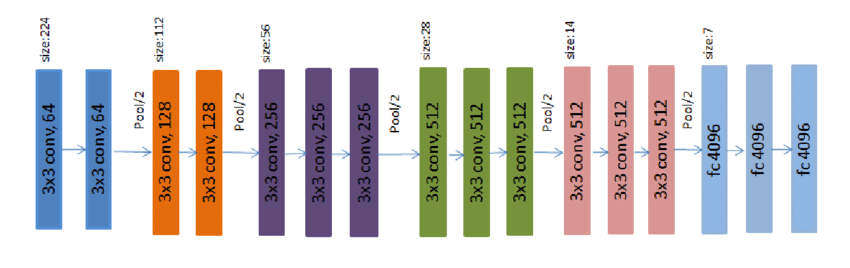
When there is not sufficient amount of data or not too much time for training, learning by transfer learning is the main option. Instead of defining new architecture, there is a predefined model that extract general features and patterns of images and all that remains is classifying by defining some FC layer at the end of model.



**VGG16**

The idea behind VGG16 is very simple but the results are amazing. This model has 5 blocks that include some convolution and max pooling layers. First blocks find simple features and contain less channels, as progressing through model more channels are defined to detect complex features. There are max pooling layers reducing dimension by factor of 2 and last 3 layers of 16 layers are fully connected layers trying to classify images.

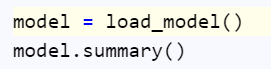
Standard input dimension for VGG16 is 224 by 224 by 3 and available pretrained weights are trained on imagenet dataset with 1000 classes.



**Summary**

Model summary takes overview of whole model that helps to recognize output and number of parameters of each layer which gives better view of model. As it is clear approximately 98 percent of model parameters belonged to VGG model. The VGG model output has shape (7, 7, 512).

Model summary shows that only very small fraction of parameters are non-trainable and remaining are trainable. Trainable parameters are parameters that update during training. Model continue learning process by updating their values by computing gradient descent on but non-trainable parameters are frozen during training and their values are fixed. So setting more trainable parameters, needs more time and more computations but your learning process is incomplete.



**Compile Model**

Compile model means to define 3 sets of parameters, loss function, optimizer and metrics. Loss function is a function that compute loss. In this example has been set to binary cross entropy because it’s binary classification problem.

**Adam**

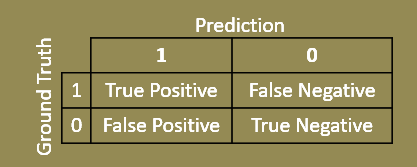
Optimizer is function that is used to compute gradient descent. In this example has been set to Adam. SGD that has been already explained, is very noisy and doesn’t descend well on curves to decrease noise in steps moving average is implemented by defining a new parameter called Momentum. SGD with Momentum has better performance on curves and need less steps to converge. Adam has 2 momentum variables, 1st order and 2nd order Momentum include their decay rates and an epsilon value that prevents from division by zero. Adam is very efficient and useful in terms of converging to local minima.

**Accuracy**

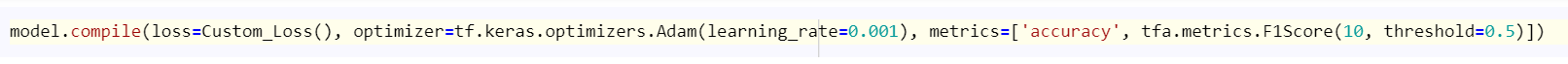
Metrics has been set to accuracy means that how much model predict accurately. The accuracy formula showed in below and as it suggests to evaluate our model by computing number of times that model predict right over by number of times that model predicts.

**F1 Score**

Accuracy determines the difference between prediction and ground truth, but sometimes more detailed metrics needed to evaluate model performance. In terms of classification, the model prediction and ground truth label is vector of zeroes and ones. The difference between prediction and ground truth is divided to these 4 options below:



Number of times that the model predicts a positive label true over number of positive labels called Recall and number of times that the model predict a positive label true over number of positive prediction called Precision. These two metrics are important to evaluate model performance but for better intuition combining these two results in F1 Score:

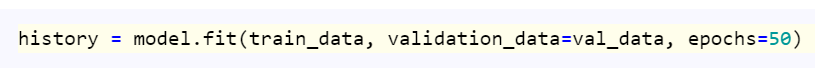
****

**Fit Model**

All previous step has been done to prepare model for training, and at this step all things that’s remained is fitting model over data. Training model needs to determine training and validation data and number of epochs, epochs mean number of times that model train on whole data.

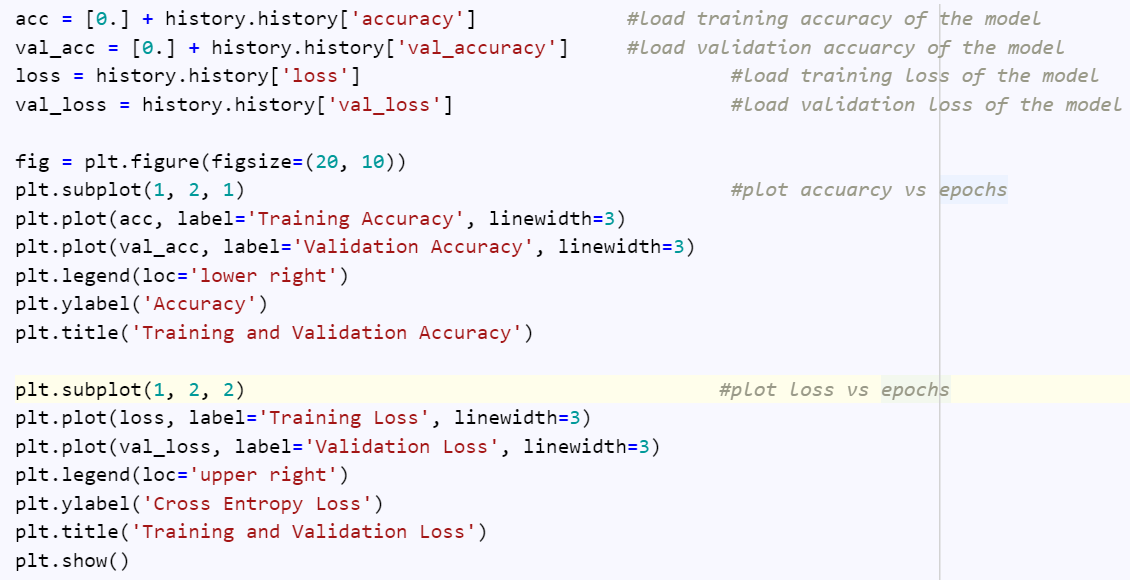
**Epoch**

Epoch means training process is passed through all training data, so number of epochs means number of times that all data have passed through training process. When dealing with batched dataset, the number of times that training algorithm has performed is not important. number of epochs shows how many times model compute loss on whole data and it is better parameter to control model.



**Analysis**

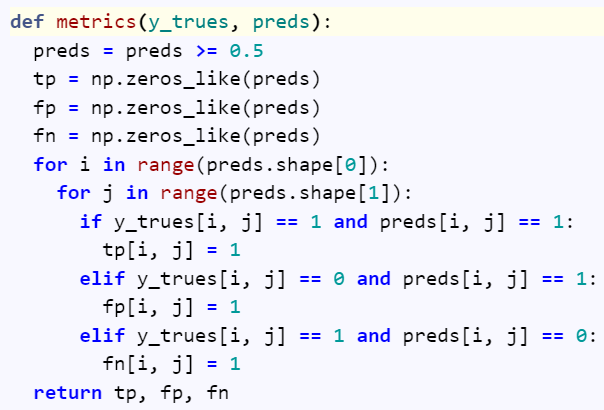
Loss and accuracy are two parameters that can be inspected through training as number of epochs. As it is clear from loss and accuracy plot, approximately 85% accuracy for validation and training would be acceptable.

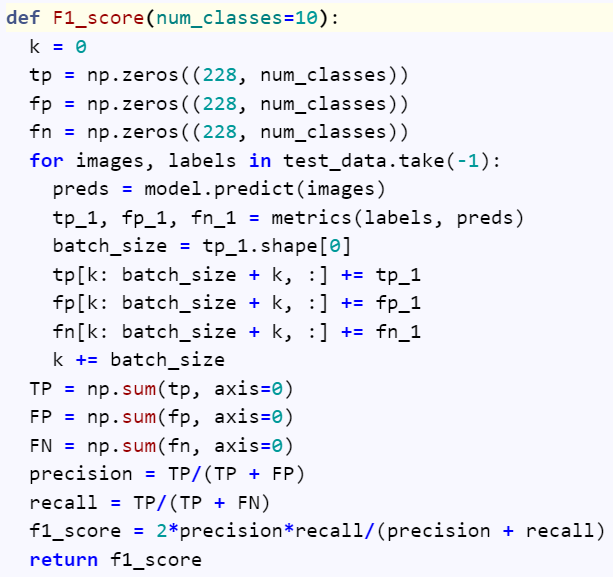
****

**Metric**

This function takes array of predictions and ground truth labels as input. Converts prediction probabilities to zeroes and ones, takes a look at each label and subsequent prediction if both are equal to 1, set the subsequent element in true positive array to 1 and compute false positive and false negative array respectively to return these three arrays as function output.

For computing f1 score on each class of test dataset, array of true labels and predictions must be extract on every batch taking these two array to metric function previously defined and compute summation of function outputs on column axis to reach the number of true positive, false positive and false negative predictions on each class. In the last part precision, recall and f1 score computed by the formula mentioned above after test data has been completely discovered.





**References**

[1] [Research Gate](https://www.researchgate.net/figure/portrays-the-VGG16-model-for-ImageNet-40-It-has-13-convolutional-layers-and-three_fig2_331562880)