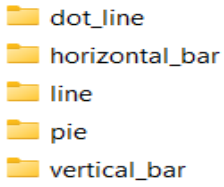
Report by Gaurav Jha

Problem Statement:
You have to develop a CNN-based classification architecture for classifying a given chart image to one of five chart classes, namely "**Line**","**Dot Line**","**Horizontal Bar**", and "**Vertical Bar**", and "**Pie**" chart.



Dataset:
Total images : 1050 images
Training data : 800 images with 5 classes
Validation data : 200 images with 5 classes
Testing dat : 50 images

Task 1 : Use the train and Val images for training and validation in an appropriate ratio (e.g., 80% for training and 20 % for validating). The CSV file contains corresponding labels for the images.

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(Train_Imgs, Train_Lbls, shuffle = True, test_size = 0.2, random_state = 42)
print('Shape of X_train: {}, y_train: {} '.format(X_train.shape, y_train.shape))
print('Shape of X_val: {}, y_val: {} '.format(X_val.shape, y_val.shape))
```

```
Shape of X_train: (800, 128, 128, 3), y_train: (800,)
Shape of X_val: (200, 128, 128, 3), y_val: (200,)
```

Task 2 : Implement a two-layer Convolutional Neural Network, and calculate accuracy, loss and plot the obtained loss. Briefly write your observation and submit your code so that we can evaluate your implementation at our end.
Model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 32)        9248
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 21, 21, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 14112)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               1806464
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 645
=================================================================
Total params: 1,817,253
Trainable params: 1,817,253
Non-trainable params: 0
_________________________________________________________________
```

Conv2D(kernel_size, (filters, filters), input_shape=(img_w, img_h, 3)) with activation function for each layer as a Rectified Linear Unit (ReLU): Activation('relu')

MaxPooling2D layer to reduce the spatial size of the incoming features; 2D input space: MaxPooling2D(pool_size=(max_pool, max_pool))

1. Flatten the input: transform the multidimensional vector into a single dimensional vector: Flatten()
2. Add dropout layer which randomly sets a certain fraction of its input to 0 and helps to reduce overfitting: Dropout(0.5)
3. Add fully connected layer with 128 nodes and activation function relu: Dense(128), Activation('relu')
4. Provide last fully connected layer which specifies the number of classes of charts: Softmax activation function outputs a vector that represents the probability distributions of a list of potential outcomes: (Dense(5, kernel_regularizer=tf.keras.regularizers.l2(0.001),activation ='softmax'))

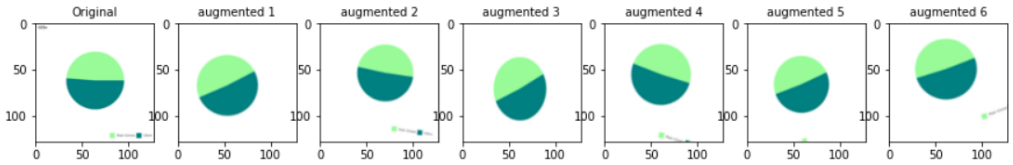
The model is compiled using adam optimizer which is a generalization of stochastic gradient descent (SGD) algo. Provided loss function is sparse_categorical_crossentropy as we are doing multiclass classification.

As far as there are not so many samples for every class add a train data generator using class ImageDataGenerator with augmentation parameters. Using the ImageDataGenerator class from Keras library, create additional images of each chart class in the memory.

```
n = randint(0,len(X_train))
samples = np.expand_dims(X_train[n], 0)
it = train_datagen.flow(samples, batch_size=32)
cols = 7

fig, ax = plt.subplots(nrows=1, ncols=cols, figsize=(15, 10))
ax[0].imshow(X_train[n], cmap='gray')
ax[0].set_title('Original', fontsize=10)

for i in range(1,cols):
    batch = it.next()    # generate batch of images
    image = batch[0].astype('uint32') # convert to unsigned int for viewing
    ax[i].set_title('augmented {}'.format(i), fontsize=10)
    ax[i].imshow(image, cmap='gray')
```

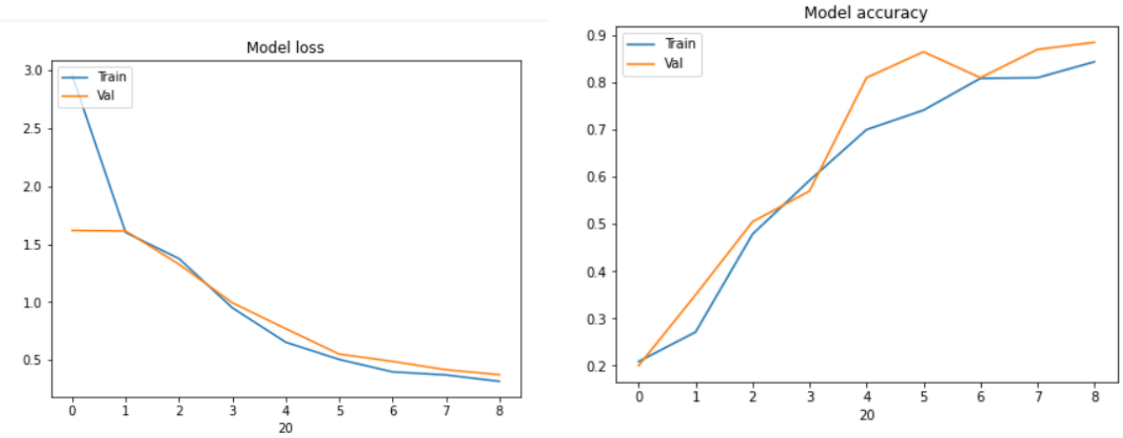


To fit the model on training and validation set we use:

Epoch:9

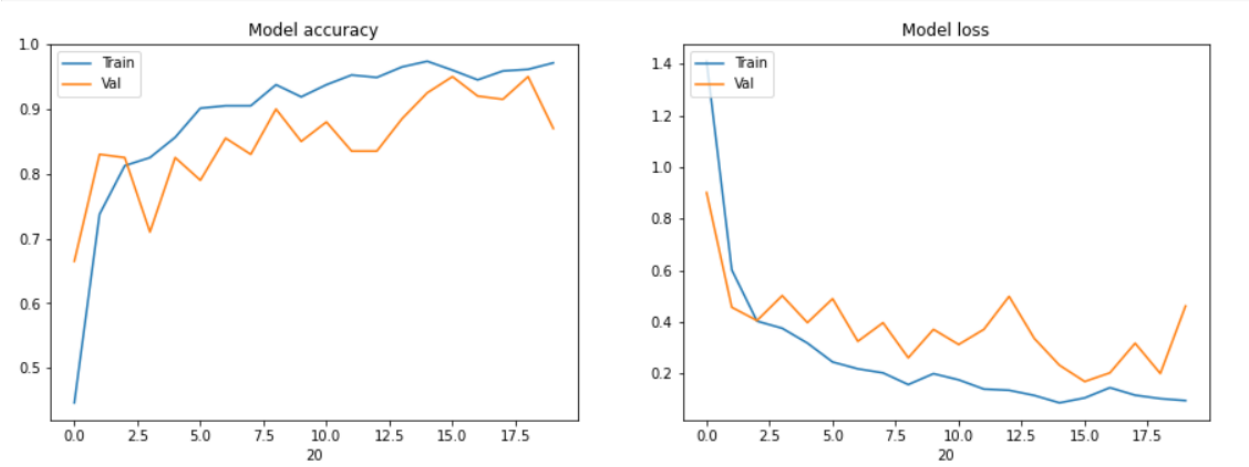
Batch_number = 32

The accuracy and loss plot:



As we can see validation accuracy is at **93%** and we achieved an almost perfect fit with well-balanced training and validation metrics.

The main problem I faced is overfitting:



As we can see we can see that the training loss is way below the validation loss and the training accuracy is above the validation accuracy, especially after the 20th epoch. It means that our model is **overfitting**.

I solved the overfitting issue:

1. By introducing regularization l2 in a fully connected layer is to solve the issue of overfitting.
2. Inserting dropout layer with p=0.5
3. By decreasing the the number of epochs(good epoch is around 9 to 11)

Task 3: Finetune a pretrained network (e.g., AlexNet) for this task and report the results.

I have fine tuned the VGG16 model for this particular task.

> VGG16 is a proven proficient algorithm for image classification (1000 classes of images). Keras framework already contain this model. We will import this model and fine-tune it to classify the images of dogs and cats (only 2 classes instead of 1000 classes).

Firstly I did the Feature Extraction using pretrained networks and we have assigned *include_top = False* because we are using convolution layer for features extraction and wants to train fully connected layer for our images classification(since it is not the part of Imagenet dataset)

Since we have 10 classes that's why we are using 10 in output dense layer with *softmax* function with loss as *categorical_crossentropy* and optimizers as *Adam* Optimizer.

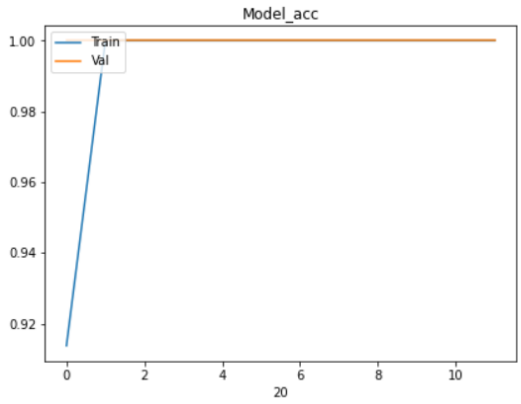
```
conv_base = VGG16(
                    weights = 'imagenet',
                    include_top = False,
                    input_shape = (128,128,3)
                  )
```

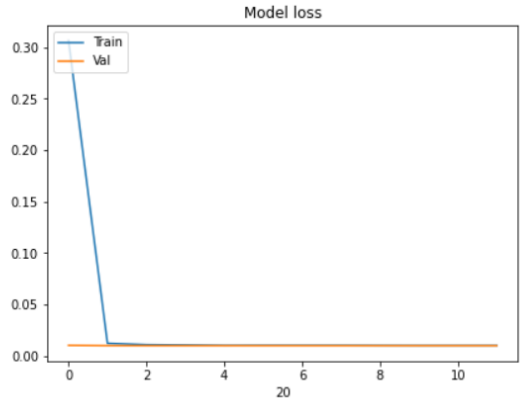
Since we have 5 classes that's why we are using 5 in output dense layer with ***softmax*** function with loss as ***categorical_crossentropy*** and optimizers as ***Adam*** Optimizer.

```
from keras import models, layers, optimizers
model = models.Sequential()
model.add(conv_base)
model.add(layers.Dropout(0.5))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dense(5,kernel_regularizer=tf.keras.regularizers.l2(0.001), activation='softmax'))
```

Result






```
val_acc = history.history['val_acc']
val_loss = history.history['val_loss']
print("Average Validation accuracy: ", np.mean(val_acc))
print("Average Validation loss: ", np.mean(val_loss))
```

```
Average Validation accuracy:  1.0
Average Validation loss:  0.009844273949662844
```