In [ ]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Deep Learning Libraries
from tensorflow.keras import datasets, layers, models, losses, Model
from keras.callbacks import ReduceLROnPlateau, LearningRateScheduler
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import plot_model
from sklearn.metrics import confusion_matrix, accuracy_score, precision_recall_f
```

In [ ]:

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
ive.mount("/content/drive", force_remount=True).
```

In [ ]:

```
def accuracy_loss_plot(history):
    plt.Figure()
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='lower right')
    plt.yticks(np.arange(0, 1, step=0.1))
    plt.show()
    plt.Figure()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Val'], loc='lower right')
    plt.yticks(np.arange(0, 1, step=0.1))
    plt.show()
    plt.Figure()
    plt.plot(history.history['loss'])
    plt.plot(history.history['accuracy'])
    plt.title('Loss vs Accuracy')
    plt.ylabel('Loss/Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Loss', 'Accuracy'], loc='lower right')
    plt.yticks(np.arange(0, 1, step=0.1))
    plt.show()
```

In [ ]:
```
def plot_augmented_data(X_train, y_train):
# define number of rows & columns
  num_row = 2
  num_col = 8
  num= num_row*num_col

# plot after
  fig2, axes2 = plt.subplots(num_row, num_col, figsize=(1.5*num_col,2*num_row))
  for X, Y in datagen.flow(X_train,y_train,batch_size=num,shuffle=False):
      for i in range(0, num):
            ax = axes2[i//num_col, i%num_col]
            ax.imshow(X[i].reshape(28,28), cmap='gray_r')
            ax.set_title('Label: {}'.format(int(Y[i])))
      break
  plt.tight_layout()
  plt.show()
```

In [ ]:
```
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [ ]:
```
def replace_values(arr, original, sub):
  s = pd.Series(arr)
  target = s.replace(original, sub)
  target = target.to_numpy()
  return target
```

In [ ]:
```
def compare_accuracy_loss(history_list):
    n_models = len(history_list)
    fig, axs = plt.subplots(n_models, 3, figsize=(10, 16))

    for i in range(n_models):
        title = 'Model ' + str(i+1)
        axs[i, 0].plot(history_list[i].history['accuracy'])
        axs[i, 0].plot(history_list[i].history['val_accuracy'])
```

```
        axs[i, 0].legend(['Train', 'Val'], loc='lower right')
        axs[i, 0].set_title(title , y=1.0, pad=-60)
        axs[i, 0].set(ylabel='Accuracy')
        axs[i, 1].plot(history_list[i].history['loss'])
        axs[i, 1].plot(history_list[i].history['val_loss'])
        axs[i, 1].legend(['Train', 'Val'], loc='upper right')
        axs[i, 1].set_title(title, y=1.0, pad=-60)
        axs[i, 1].set(ylabel='Loss')
        axs[i, 2].plot(history_list[i].history['loss'])
        axs[i, 2].plot(history_list[i].history['accuracy'])
        axs[i, 2].legend(['Loss', 'Accuracy'], loc='upper right')
        axs[i, 2].set_title(title, y=1.0, pad=-60)
        axs[i, 2].set(ylabel='Loss')

    for i in range((n_models * 2) - 2, n_models * 2):
        axs.flat[i].set(xlabel='Epoch')
```

In [ ]:

```
def print_accuracy(model, y_train, y_val, y_test, mode = 0):
    global X_train, X_val, X_test
    global X_train_tl, X_val_tl, X_test_tl

    if mode == 0:
        y_pred_train = np.argmax(model.predict(X_train), axis=-1)
        y_pred_val = np.argmax(model.predict(X_val), axis=-1)
        y_pred_test = np.argmax(model.predict(X_test), axis=-1)
    else:
        y_pred_train = np.argmax(model.predict(X_train_tl), axis=-1)
        y_pred_val = np.argmax(model.predict(X_val_tl), axis=-1)
        y_pred_test = np.argmax(model.predict(X_test_tl), axis=-1)

    preds = [y_pred_train, y_pred_val, y_pred_test]
    trues = [y_train, y_val, y_test]

    tag = ['Train', 'Val', 'Test']

    for i in range(3):
        accuracy = accuracy_score(trues[i], preds[i])
        metric = list(precision_recall_fscore_support(trues[i], preds[i], averag
        print(tag[i], 'Set Accuracy: \t', round(accuracy * 100, 2))
        print(tag[i], 'Set Precision: \t', round(metric[0], 2))
        print(tag[i], 'Set Recall: \t', round(metric[1], 2))
        print(tag[i], 'Set F score: \t', round(metric[2], 2))
        print()
```

In [ ]:

```
data = np.load('/content/drive/MyDrive/mnist/fashion_mnist_dataset_train.npy', a
```

In [ ]:

```
data
```

Out[ ]:

```
{'features': array([[[0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         ...,
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.]],

        [[0., 0., 0., ..., 0., 0., 0.],
```

```
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]]]),
 'target': array([5., 2., 1., ..., 3., 1., 4.])}
```

In [ ]:

```
data['target'].shape
```

Out[ ]:

```
(60000,)
```

In [ ]:

```
target = replace_values(data['target'], [1, 2, 3, 4, 5], [0, 1, 2, 3, 4])
```

In [ ]:

```
target.shape
```

Out[ ]:

```
(60000,)
```

In [ ]:

```
from sklearn.model_selection import train_test_split
# Splitting the data into train, test, and validation sets
X_train, X_test, y_train, y_test = train_test_split(data['features'], target, te
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=0.5, r
```

In [ ]:

```
X_train.shape
```

Out[ ]: (48000, 28, 28)

In [ ]:
```
X_val.shape
```

Out[ ]: (6000, 28, 28)

In [ ]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train[i], cmap=plt.cm.binary)
    plt.xlabel(y_train[i])
plt.show()
```

In [ ]:
```
#plot the label distribution
import pandas as pd
```

```
df_label = pd.DataFrame(data['target'])
```

In [ ]:
```
df_label.head()
```

Out[ ]:

	0
0	5.0
1	2.0
2	1.0
3	2.0
4	1.0

In [ ]:
```
df_label.value_counts()
```

Out[ ]:
```
2.0    12019
3.0    12011
4.0    11992
5.0    11989
1.0    11989
dtype: int64
```

In [ ]:
```
X_train = X_train.reshape((-1, 28, 28, 1))
X_val = X_val.reshape((-1, 28, 28, 1))
X_test = X_test.reshape((-1, 28, 28, 1))
```

Brief Model

In [ ]:
```
model_brief=models.Sequential()
model_brief.add(layers.Conv2D(32, (3,3) , padding='same',activation='relu', inpu
model_brief.add(layers.BatchNormalization())
model_brief.add(layers.MaxPooling2D(pool_size=(2,2)))

model_brief.add(layers.Flatten())
model_brief.add(layers.Dense(128, activation='relu'))
model_brief.add(layers.Dense(5, activation='softmax'))
```

In [ ]:
```
model_brief.summary()
```

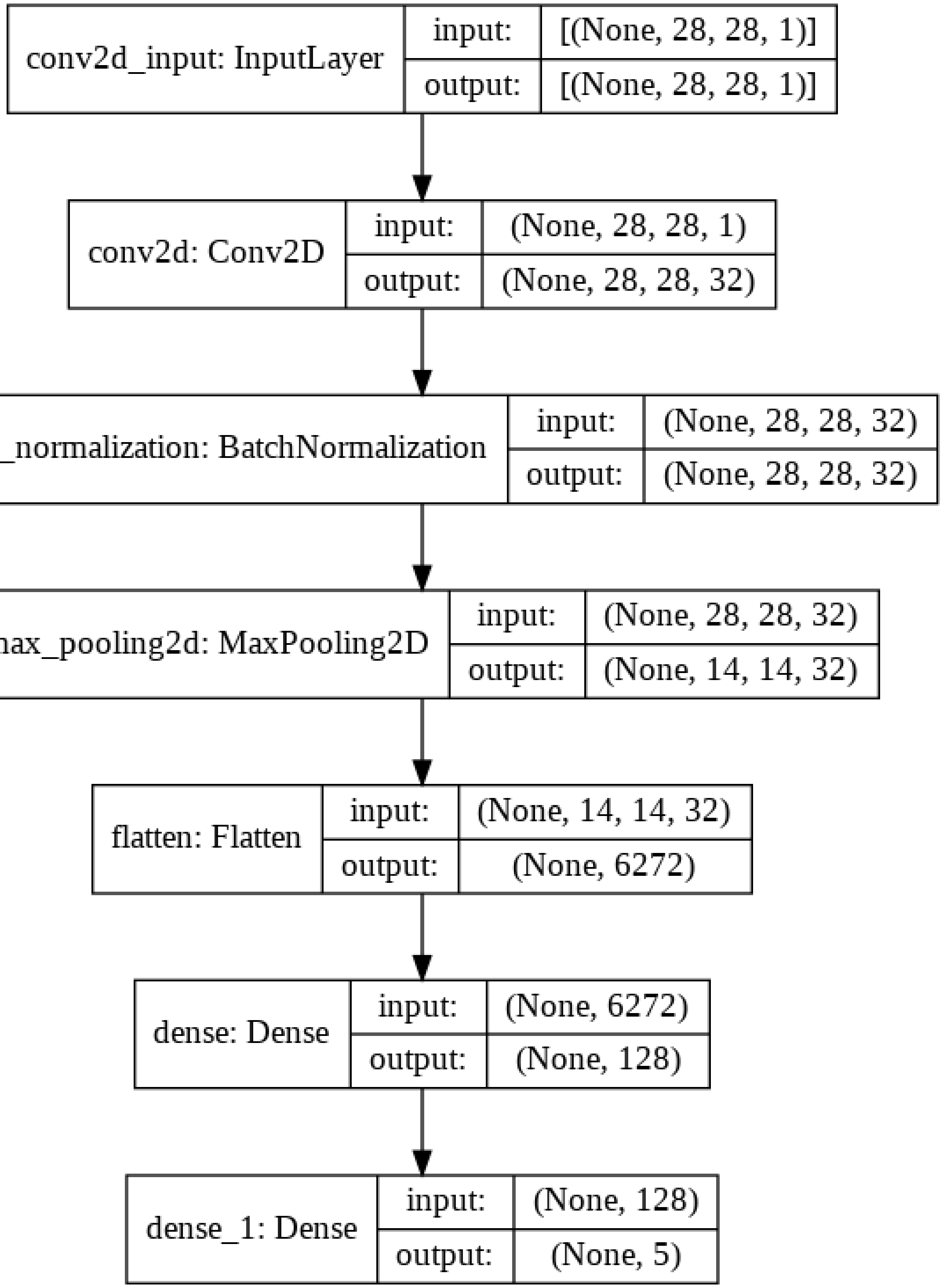
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 128)               802944
```

```
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 804,037
Trainable params: 803,973
Non-trainable params: 64
_________________________________________________________________
```

In [ ]:

```
plot_model(model_brief, show_shapes=True, rankdir="TD")
```

Out[ ]:



Description of brief Model

This model is a concise model consisting of one convolutional layer as described by [1]. The convolutional layer has 32 filters of size 3 x 3. This is followed by Batch Normalization.

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. We normalize each layer's inputs by using the mean and variance of the values in the current batch.

This is followed by a max pooling layer which reduces the spatial dimensions by downsampling the feature mask from 28 x 28 to 14 x 14.

The output of the dense layer is flattened and passed to a dense or fully connected layer. The final output layer contains 5 nodes for each of the five classes.

In [ ]:

```
model_brief.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Tru
              metrics=['accuracy'])
```

In [ ]:

```
%%time
with tf.device('/device:GPU:0'):
  history = model_brief.fit(X_train, y_train, epochs=100,
                  validation_data=(X_val, y_val), batch_size=128)
```

```
Epoch 1/100
375/375 [==============================] - 4s 5ms/step - loss: 0.6951 - accurac
y: 0.7429 - val_loss: 0.5559 - val_accuracy: 0.8400
Epoch 2/100
375/375 [==============================] - 1s 3ms/step - loss: 0.2919 - accurac
y: 0.8837 - val_loss: 0.2748 - val_accuracy: 0.8910
Epoch 3/100
375/375 [==============================] - 1s 3ms/step - loss: 0.2315 - accurac
y: 0.9089 - val_loss: 0.3020 - val_accuracy: 0.8800
Epoch 4/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1933 - accurac
y: 0.9236 - val_loss: 0.2442 - val_accuracy: 0.9055
Epoch 5/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1648 - accurac
y: 0.9353 - val_loss: 0.2931 - val_accuracy: 0.8893
Epoch 6/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1395 - accurac
y: 0.9456 - val_loss: 0.3399 - val_accuracy: 0.8800
Epoch 7/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1297 - accurac
y: 0.9498 - val_loss: 0.2718 - val_accuracy: 0.9078
Epoch 8/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1050 - accurac
y: 0.9602 - val_loss: 0.3030 - val_accuracy: 0.9077
Epoch 9/100
375/375 [==============================] - 1s 3ms/step - loss: 0.1034 - accurac
y: 0.9591 - val_loss: 0.4330 - val_accuracy: 0.8755
Epoch 10/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0881 - accurac
y: 0.9674 - val_loss: 0.3146 - val_accuracy: 0.9095
Epoch 11/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0835 - accurac
y: 0.9696 - val_loss: 0.3370 - val_accuracy: 0.9077
Epoch 12/100
```

```
375/375 [==============================] - 1s 3ms/step - loss: 0.0718 - accurac
y: 0.9731 - val_loss: 0.3492 - val_accuracy: 0.9042
Epoch 13/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0671 - accurac
y: 0.9738 - val_loss: 0.3971 - val_accuracy: 0.8953
Epoch 14/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0632 - accurac
y: 0.9764 - val_loss: 0.4260 - val_accuracy: 0.8938
Epoch 15/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0565 - accurac
y: 0.9781 - val_loss: 0.4132 - val_accuracy: 0.9005
Epoch 16/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0534 - accurac
y: 0.9799 - val_loss: 0.4280 - val_accuracy: 0.8998
Epoch 17/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0511 - accurac
y: 0.9808 - val_loss: 0.4442 - val_accuracy: 0.9043
Epoch 18/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0439 - accurac
y: 0.9840 - val_loss: 0.4865 - val_accuracy: 0.8978
Epoch 19/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0345 - accurac
y: 0.9878 - val_loss: 0.4475 - val_accuracy: 0.9033
Epoch 20/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0430 - accurac
y: 0.9843 - val_loss: 0.5378 - val_accuracy: 0.8903
Epoch 21/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0389 - accurac
y: 0.9856 - val_loss: 0.5681 - val_accuracy: 0.8848
Epoch 22/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0332 - accurac
y: 0.9874 - val_loss: 0.5080 - val_accuracy: 0.9005
Epoch 23/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0319 - accurac
y: 0.9893 - val_loss: 0.5880 - val_accuracy: 0.8927
Epoch 24/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0274 - accurac
y: 0.9900 - val_loss: 0.5385 - val_accuracy: 0.8995
Epoch 25/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0297 - accurac
y: 0.9893 - val_loss: 0.5643 - val_accuracy: 0.8955
Epoch 26/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0336 - accurac
y: 0.9883 - val_loss: 0.5836 - val_accuracy: 0.8970
Epoch 27/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0242 - accurac
y: 0.9914 - val_loss: 0.5432 - val_accuracy: 0.9025
Epoch 28/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0222 - accurac
y: 0.9922 - val_loss: 0.5436 - val_accuracy: 0.9073
Epoch 29/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0231 - accurac
y: 0.9922 - val_loss: 0.5947 - val_accuracy: 0.8998
Epoch 30/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0278 - accurac
y: 0.9903 - val_loss: 0.6031 - val_accuracy: 0.9005
Epoch 31/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0217 - accurac
y: 0.9925 - val_loss: 0.5788 - val_accuracy: 0.9078
Epoch 32/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0190 - accurac
y: 0.9941 - val_loss: 0.6228 - val_accuracy: 0.8992
Epoch 33/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0221 - accurac
y: 0.9924 - val_loss: 0.6339 - val_accuracy: 0.9020
```

```
Epoch 34/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0265 - accurac
y: 0.9909 - val_loss: 0.6189 - val_accuracy: 0.9048
Epoch 35/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0245 - accurac
y: 0.9910 - val_loss: 0.6668 - val_accuracy: 0.8952
Epoch 36/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0215 - accurac
y: 0.9933 - val_loss: 0.6565 - val_accuracy: 0.9033
Epoch 37/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0211 - accurac
y: 0.9932 - val_loss: 0.6643 - val_accuracy: 0.9010
Epoch 38/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0179 - accurac
y: 0.9937 - val_loss: 0.7681 - val_accuracy: 0.8837
Epoch 39/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0215 - accurac
y: 0.9927 - val_loss: 0.6822 - val_accuracy: 0.9022
Epoch 40/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0192 - accurac
y: 0.9939 - val_loss: 0.7871 - val_accuracy: 0.8948
Epoch 41/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0167 - accurac
y: 0.9939 - val_loss: 0.7008 - val_accuracy: 0.9038
Epoch 42/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0148 - accurac
y: 0.9949 - val_loss: 0.8642 - val_accuracy: 0.8818
Epoch 43/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0110 - accurac
y: 0.9960 - val_loss: 1.0415 - val_accuracy: 0.8695
Epoch 44/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0185 - accurac
y: 0.9940 - val_loss: 0.7764 - val_accuracy: 0.8912
Epoch 45/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0247 - accurac
y: 0.9920 - val_loss: 0.7605 - val_accuracy: 0.8980
Epoch 46/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0204 - accurac
y: 0.9935 - val_loss: 0.7360 - val_accuracy: 0.9027
Epoch 47/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0107 - accurac
y: 0.9962 - val_loss: 0.8104 - val_accuracy: 0.8950
Epoch 48/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0146 - accurac
y: 0.9954 - val_loss: 0.8514 - val_accuracy: 0.8940
Epoch 49/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0137 - accurac
y: 0.9959 - val_loss: 0.8174 - val_accuracy: 0.8978
Epoch 50/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0266 - accurac
y: 0.9917 - val_loss: 0.8373 - val_accuracy: 0.9012
Epoch 51/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0204 - accurac
y: 0.9930 - val_loss: 0.9129 - val_accuracy: 0.8918
Epoch 52/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0178 - accurac
y: 0.9943 - val_loss: 0.7983 - val_accuracy: 0.9000
Epoch 53/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0102 - accurac
y: 0.9968 - val_loss: 0.7581 - val_accuracy: 0.9047
Epoch 54/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0126 - accurac
y: 0.9957 - val_loss: 0.8290 - val_accuracy: 0.8975
Epoch 55/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0183 - accurac
```

```
y: 0.9939 - val_loss: 0.8167 - val_accuracy: 0.8970
Epoch 56/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0052 - accurac
y: 0.9983 - val_loss: 0.8221 - val_accuracy: 0.9018
Epoch 57/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0114 - accurac
y: 0.9965 - val_loss: 1.0310 - val_accuracy: 0.8825
Epoch 58/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0139 - accurac
y: 0.9951 - val_loss: 0.8660 - val_accuracy: 0.8945
Epoch 59/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0201 - accurac
y: 0.9936 - val_loss: 0.8276 - val_accuracy: 0.8963
Epoch 60/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0164 - accurac
y: 0.9951 - val_loss: 0.8467 - val_accuracy: 0.8962
Epoch 61/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0159 - accurac
y: 0.9945 - val_loss: 1.0394 - val_accuracy: 0.8780
Epoch 62/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0125 - accurac
y: 0.9959 - val_loss: 1.1154 - val_accuracy: 0.8772
Epoch 63/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0070 - accurac
y: 0.9979 - val_loss: 0.8850 - val_accuracy: 0.9015
Epoch 64/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0108 - accurac
y: 0.9966 - val_loss: 0.8547 - val_accuracy: 0.8975
Epoch 65/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0082 - accurac
y: 0.9971 - val_loss: 1.0011 - val_accuracy: 0.8863
Epoch 66/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0219 - accurac
y: 0.9932 - val_loss: 0.8708 - val_accuracy: 0.9032
Epoch 67/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0058 - accurac
y: 0.9982 - val_loss: 1.1742 - val_accuracy: 0.8773
Epoch 68/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0158 - accurac
y: 0.9951 - val_loss: 0.9580 - val_accuracy: 0.8938
Epoch 69/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0172 - accurac
y: 0.9953 - val_loss: 0.9679 - val_accuracy: 0.8912
Epoch 70/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0155 - accurac
y: 0.9954 - val_loss: 0.9287 - val_accuracy: 0.9022
Epoch 71/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0132 - accurac
y: 0.9966 - val_loss: 0.9475 - val_accuracy: 0.8952
Epoch 72/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0061 - accurac
y: 0.9983 - val_loss: 0.8916 - val_accuracy: 0.8995
Epoch 73/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0184 - accurac
y: 0.9952 - val_loss: 1.0422 - val_accuracy: 0.8883
Epoch 74/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0079 - accurac
y: 0.9973 - val_loss: 0.9429 - val_accuracy: 0.8980
Epoch 75/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0065 - accurac
y: 0.9976 - val_loss: 1.0057 - val_accuracy: 0.8915
Epoch 76/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0224 - accurac
y: 0.9934 - val_loss: 1.1400 - val_accuracy: 0.8797
Epoch 77/100
```

```
375/375 [==============================] - 1s 3ms/step - loss: 0.0117 - accurac
y: 0.9963 - val_loss: 0.9020 - val_accuracy: 0.9008
Epoch 78/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0103 - accurac
y: 0.9969 - val_loss: 0.9799 - val_accuracy: 0.8938
Epoch 79/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0051 - accurac
y: 0.9985 - val_loss: 1.2243 - val_accuracy: 0.8825
Epoch 80/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0080 - accurac
y: 0.9972 - val_loss: 1.1213 - val_accuracy: 0.8840
Epoch 81/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0099 - accurac
y: 0.9966 - val_loss: 1.0490 - val_accuracy: 0.8895
Epoch 82/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0128 - accurac
y: 0.9961 - val_loss: 0.9561 - val_accuracy: 0.8988
Epoch 83/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0037 - accurac
y: 0.9989 - val_loss: 1.0845 - val_accuracy: 0.8967
Epoch 84/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0145 - accurac
y: 0.9958 - val_loss: 1.0285 - val_accuracy: 0.8908
Epoch 85/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0177 - accurac
y: 0.9945 - val_loss: 1.2448 - val_accuracy: 0.8788
Epoch 86/100
375/375 [==============================] - 1s 4ms/step - loss: 0.0130 - accurac
y: 0.9964 - val_loss: 1.0051 - val_accuracy: 0.8973
Epoch 87/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0027 - accurac
y: 0.9993 - val_loss: 0.9851 - val_accuracy: 0.8990
Epoch 88/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0076 - accurac
y: 0.9980 - val_loss: 1.0878 - val_accuracy: 0.8930
Epoch 89/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0098 - accurac
y: 0.9973 - val_loss: 1.0757 - val_accuracy: 0.8937
Epoch 90/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0097 - accurac
y: 0.9968 - val_loss: 1.0078 - val_accuracy: 0.8960
Epoch 91/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0082 - accurac
y: 0.9976 - val_loss: 1.1433 - val_accuracy: 0.8945
Epoch 92/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0169 - accurac
y: 0.9962 - val_loss: 0.9768 - val_accuracy: 0.9008
Epoch 93/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0090 - accurac
y: 0.9970 - val_loss: 1.0356 - val_accuracy: 0.8958
Epoch 94/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0066 - accurac
y: 0.9975 - val_loss: 1.0802 - val_accuracy: 0.8932
Epoch 95/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0102 - accurac
y: 0.9967 - val_loss: 1.0731 - val_accuracy: 0.8957
Epoch 96/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0065 - accurac
y: 0.9982 - val_loss: 1.0556 - val_accuracy: 0.8997
Epoch 97/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0054 - accurac
y: 0.9982 - val_loss: 1.3618 - val_accuracy: 0.8777
Epoch 98/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0036 - accurac
y: 0.9992 - val_loss: 1.1840 - val_accuracy: 0.8902
```

```
Epoch 99/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0073 - accurac
y: 0.9976 - val_loss: 1.2314 - val_accuracy: 0.8942
Epoch 100/100
375/375 [==============================] - 1s 3ms/step - loss: 0.0164 - accurac
y: 0.9947 - val_loss: 1.0618 - val_accuracy: 0.8932
CPU times: user 2min 9s, sys: 21.1 s, total: 2min 31s
Wall time: 2min 9s
```

In [ ]:

```
%%time
with tf.device('/device:GPU:0'):
  test_loss, test_acc = model_brief.evaluate(X_test,  y_test, verbose=2)
print("Test of accuracy of brief model", test_acc)
```

```
188/188 - 0s - loss: 1.0568 - accuracy: 0.8957
Test of accuracy of brief model 0.8956666588783264
CPU times: user 355 ms, sys: 43.4 ms, total: 398 ms
Wall time: 279 ms
```

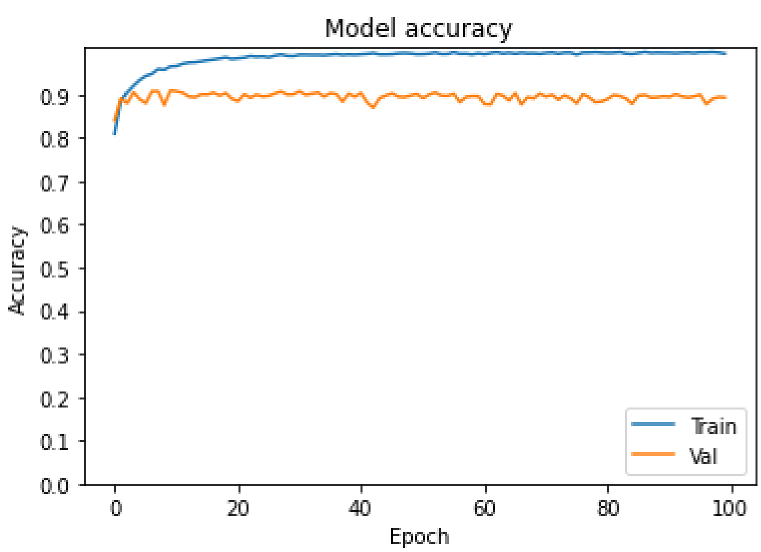
In [ ]:

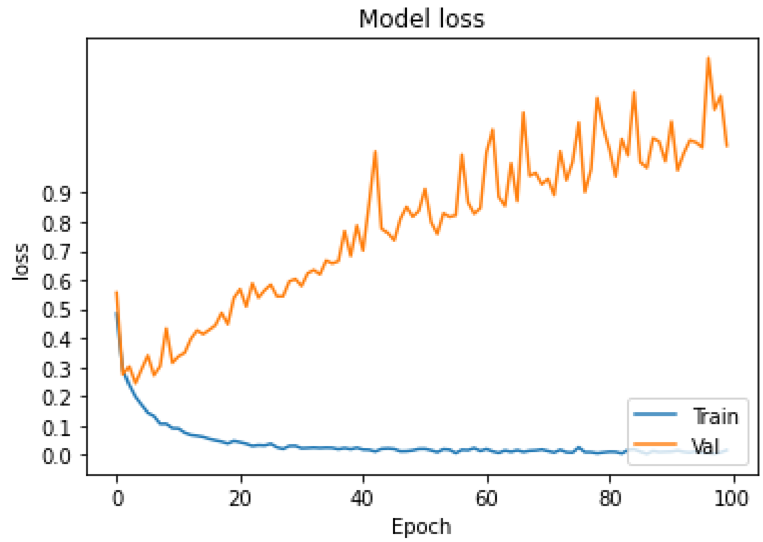
```
print_accuracy(model_brief, y_train, y_val, y_test)
```

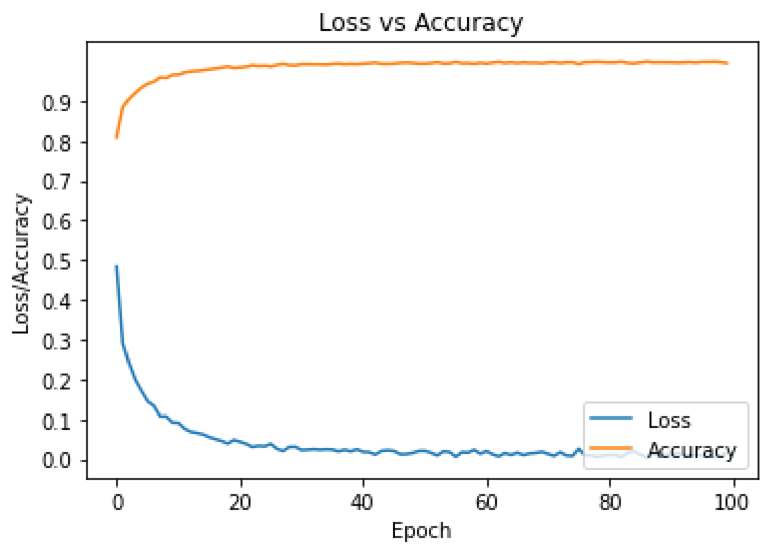
```
Train Set Accuracy:      99.81
Train Set Precision:     1.0
Train Set Recall:        1.0
Train Set F score:       1.0

Val Set Accuracy:        89.32
Val Set Precision:       0.89
Val Set Recall:          0.89
Val Set F score:         0.89

Test Set Accuracy:       89.57
Test Set Precision:      0.9
Test Set Recall:         0.9
Test Set F score:        0.9
```

In [ ]:

```
accuracy_loss_plot(history)
```







In [ ]:

```
history_list = []
history_list.append(history)
```

Explanation of Brief Model Plots

Accuracy plots

This model achieves an accuracy of up to 99.81% on the training data. However, it achieves only 89.37% and 89.57% accuracy on the validation and test sets respectively. From the accuracy graph above, there is a large gap between the training and validation accuracy per epoch.

Loss plots

There is also a relatively large gap between the losses per epoch in the training and validation sets. This represents possible overfitting.

Reason for overfitting

- This is most likely due to the lack of regularization

Deeper Model

In [ ]:

```
#output softmax layer should have 5 outputs
# Building a ConvNet
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_sha
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(layers.Dropout(0.25))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.25))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
```

In [ ]:

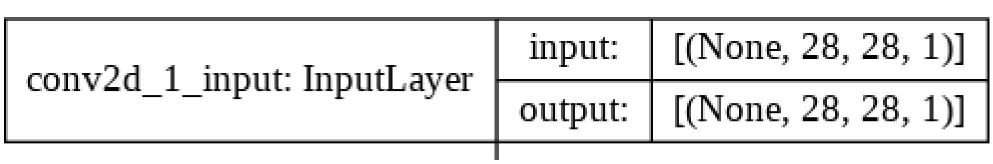
```
model.summary()
```

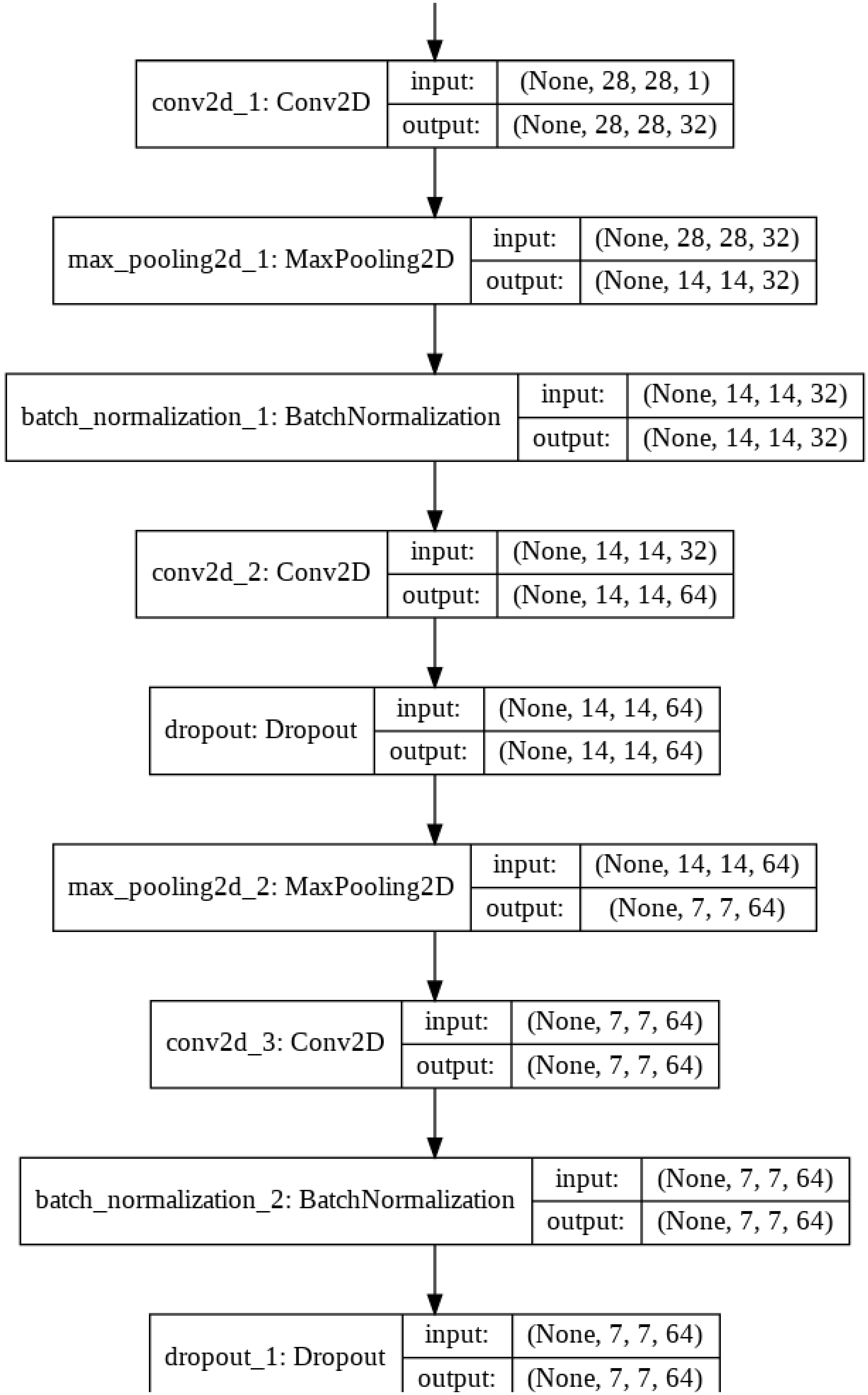
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 32)        128       
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        18496     
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 64)          36928     
_________________________________________________________________
batch_normalization_2 (Batch (None, 7, 7, 64)          256       
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 64)                200768    
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325       
=================================================================
Total params: 257,221
Trainable params: 257,029
Non-trainable params: 192
_________________________________________________________________
```

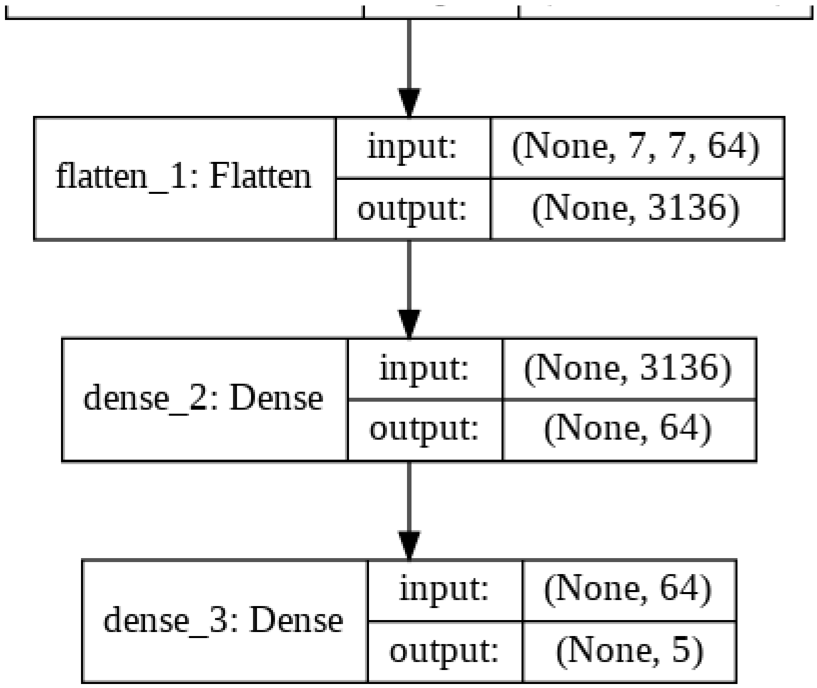
In [ ]:

```
plot_model(model, show_shapes=True, rankdir="TD")
```

Out[ ]:




conv2d_1: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 32)
max_pooling2d_1: MaxPooling2D
input: (None, 28, 28, 32)
output: (None, 14, 14, 32)
batch_normalization_1: BatchNormalization
input: (None, 14, 14, 32)
output: (None, 14, 14, 32)
conv2d_2: Conv2D
input: (None, 14, 14, 32)
output: (None, 14, 14, 64)
dropout: Dropout
input: (None, 14, 14, 64)
output: (None, 14, 14, 64)
max_pooling2d_2: MaxPooling2D
input: (None, 14, 14, 64)
output: (None, 7, 7, 64)
conv2d_3: Conv2D
input: (None, 7, 7, 64)
output: (None, 7, 7, 64)
batch_normalization_2: BatchNormalization
input: (None, 7, 7, 64)
output: (None, 7, 7, 64)
dropout_1: Dropout
input: (None, 7, 7, 64)
output: (None, 7, 7, 64)



Description of Deeper Model

This model is a deeper model consisting of three convolutional layers partially influenced by the model described in [x]. The convolutional layer has 32 filters of size 3 x 3. This is followed by a MaxPooling layer of size 2 x 2 and Batch Normalization. The Max pooling layer reduces the spatial dimensions by half.

The second layer has 64 filters, also of size 3 x 3. This is followed by a dropout layer for reqularization and a max pooling layer. Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.

During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer.

The final convolutional layer also contains 64 filters and is followed by Batch Normalization and Dropout.

The output of the dense layer is flattened and passed to a dense or fully connected layer. The final output layer contains 5 nodes for each of the five classes. The final layer utilises the softmax activation function since there are multiple output classes.

In [ ]:
```
%%time
with tf.device('/device:GPU:0'):
  model.compile(optimizer='adam',
                loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Tru
                metrics=['accuracy'])
  history_deeper = model.fit(X_train, y_train, epochs=100,
                      validation_data=(X_val, y_val), batch_size=128)
```

```
Epoch 1/100
```

```
375/375 [==============================] - 2s 5ms/step - loss: 0.6385 - accurac
y: 0.7450 - val_loss: 1.1492 - val_accuracy: 0.4713
Epoch 2/100
375/375 [==============================] - 2s 5ms/step - loss: 0.3366 - accurac
y: 0.8644 - val_loss: 0.4337 - val_accuracy: 0.8197
Epoch 3/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2915 - accurac
y: 0.8814 - val_loss: 0.4222 - val_accuracy: 0.8238
Epoch 4/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2638 - accurac
y: 0.8926 - val_loss: 0.3331 - val_accuracy: 0.8712
Epoch 5/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2406 - accurac
y: 0.9033 - val_loss: 0.3855 - val_accuracy: 0.8453
Epoch 6/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2271 - accurac
y: 0.9100 - val_loss: 0.3003 - val_accuracy: 0.8770
Epoch 7/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2114 - accurac
y: 0.9151 - val_loss: 0.3842 - val_accuracy: 0.8472
Epoch 8/100
375/375 [==============================] - 2s 5ms/step - loss: 0.2007 - accurac
y: 0.9199 - val_loss: 0.2408 - val_accuracy: 0.9043
Epoch 9/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1921 - accurac
y: 0.9225 - val_loss: 0.2403 - val_accuracy: 0.9053
Epoch 10/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1915 - accurac
y: 0.9224 - val_loss: 0.5198 - val_accuracy: 0.8233
Epoch 11/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1761 - accurac
y: 0.9273 - val_loss: 0.4780 - val_accuracy: 0.8272
Epoch 12/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1698 - accurac
y: 0.9335 - val_loss: 0.2456 - val_accuracy: 0.9033
Epoch 13/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1572 - accurac
y: 0.9360 - val_loss: 0.2335 - val_accuracy: 0.9092
Epoch 14/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1553 - accurac
y: 0.9394 - val_loss: 0.2222 - val_accuracy: 0.9102
Epoch 15/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1439 - accurac
y: 0.9419 - val_loss: 0.3745 - val_accuracy: 0.8618
Epoch 16/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1488 - accurac
y: 0.9402 - val_loss: 0.2561 - val_accuracy: 0.9020
Epoch 17/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1478 - accurac
y: 0.9425 - val_loss: 0.2195 - val_accuracy: 0.9155
Epoch 18/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1307 - accurac
y: 0.9482 - val_loss: 0.2271 - val_accuracy: 0.9147
Epoch 19/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1218 - accurac
y: 0.9519 - val_loss: 0.2102 - val_accuracy: 0.9207
Epoch 20/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1254 - accurac
y: 0.9513 - val_loss: 0.3218 - val_accuracy: 0.8805
Epoch 21/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1309 - accurac
y: 0.9471 - val_loss: 0.2868 - val_accuracy: 0.8915
Epoch 22/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1196 - accurac
y: 0.9529 - val_loss: 0.2691 - val_accuracy: 0.8973
```

```
Epoch 23/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1102 - accurac
y: 0.9569 - val_loss: 0.2135 - val_accuracy: 0.9225
Epoch 24/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1169 - accurac
y: 0.9526 - val_loss: 0.3502 - val_accuracy: 0.8770
Epoch 25/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1099 - accurac
y: 0.9564 - val_loss: 0.2258 - val_accuracy: 0.9155
Epoch 26/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1066 - accurac
y: 0.9579 - val_loss: 0.2821 - val_accuracy: 0.8975
Epoch 27/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1035 - accurac
y: 0.9597 - val_loss: 0.2416 - val_accuracy: 0.9133
Epoch 28/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1024 - accurac
y: 0.9593 - val_loss: 0.2319 - val_accuracy: 0.9148
Epoch 29/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0924 - accurac
y: 0.9641 - val_loss: 0.2444 - val_accuracy: 0.9140
Epoch 30/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1000 - accurac
y: 0.9606 - val_loss: 0.2292 - val_accuracy: 0.9188
Epoch 31/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0912 - accurac
y: 0.9648 - val_loss: 0.2743 - val_accuracy: 0.9075
Epoch 32/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0854 - accurac
y: 0.9661 - val_loss: 0.2852 - val_accuracy: 0.9057
Epoch 33/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0911 - accurac
y: 0.9655 - val_loss: 0.4887 - val_accuracy: 0.8542
Epoch 34/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0891 - accurac
y: 0.9649 - val_loss: 0.3201 - val_accuracy: 0.8938
Epoch 35/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0881 - accurac
y: 0.9653 - val_loss: 0.2870 - val_accuracy: 0.9042
Epoch 36/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0813 - accurac
y: 0.9681 - val_loss: 0.2668 - val_accuracy: 0.9138
Epoch 37/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0842 - accurac
y: 0.9679 - val_loss: 0.2783 - val_accuracy: 0.9053
Epoch 38/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0752 - accurac
y: 0.9707 - val_loss: 0.3422 - val_accuracy: 0.8932
Epoch 39/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0774 - accurac
y: 0.9689 - val_loss: 0.3101 - val_accuracy: 0.9023
Epoch 40/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0800 - accurac
y: 0.9680 - val_loss: 0.3076 - val_accuracy: 0.9025
Epoch 41/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0748 - accurac
y: 0.9708 - val_loss: 0.2777 - val_accuracy: 0.9127
Epoch 42/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0700 - accurac
y: 0.9732 - val_loss: 0.2620 - val_accuracy: 0.9198
Epoch 43/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0757 - accurac
y: 0.9708 - val_loss: 0.3221 - val_accuracy: 0.8982
Epoch 44/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0712 - accurac
```

```
y: 0.9711 - val_loss: 0.3766 - val_accuracy: 0.8907
Epoch 45/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0673 - accurac
y: 0.9730 - val_loss: 0.2522 - val_accuracy: 0.9183
Epoch 46/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0655 - accurac
y: 0.9749 - val_loss: 0.2622 - val_accuracy: 0.9190
Epoch 47/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0686 - accurac
y: 0.9725 - val_loss: 0.3067 - val_accuracy: 0.9037
Epoch 48/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0646 - accurac
y: 0.9751 - val_loss: 0.4400 - val_accuracy: 0.8802
Epoch 49/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0667 - accurac
y: 0.9739 - val_loss: 0.4445 - val_accuracy: 0.8817
Epoch 50/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0634 - accurac
y: 0.9746 - val_loss: 0.2939 - val_accuracy: 0.9103
Epoch 51/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0594 - accurac
y: 0.9769 - val_loss: 0.3458 - val_accuracy: 0.9073
Epoch 52/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0635 - accurac
y: 0.9753 - val_loss: 0.2837 - val_accuracy: 0.9140
Epoch 53/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0594 - accurac
y: 0.9773 - val_loss: 0.5342 - val_accuracy: 0.8630
Epoch 54/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0649 - accurac
y: 0.9743 - val_loss: 0.2943 - val_accuracy: 0.9180
Epoch 55/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0570 - accurac
y: 0.9788 - val_loss: 0.2762 - val_accuracy: 0.9220
Epoch 56/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0640 - accurac
y: 0.9751 - val_loss: 0.3132 - val_accuracy: 0.9113
Epoch 57/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0587 - accurac
y: 0.9777 - val_loss: 0.2937 - val_accuracy: 0.9125
Epoch 58/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0545 - accurac
y: 0.9792 - val_loss: 0.3263 - val_accuracy: 0.9058
Epoch 59/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0613 - accurac
y: 0.9758 - val_loss: 0.3393 - val_accuracy: 0.9023
Epoch 60/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0575 - accurac
y: 0.9782 - val_loss: 0.2993 - val_accuracy: 0.9158
Epoch 61/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0522 - accurac
y: 0.9810 - val_loss: 0.3954 - val_accuracy: 0.8940
Epoch 62/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0539 - accurac
y: 0.9798 - val_loss: 0.3372 - val_accuracy: 0.9045
Epoch 63/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0552 - accurac
y: 0.9789 - val_loss: 0.3245 - val_accuracy: 0.9088
Epoch 64/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0538 - accurac
y: 0.9795 - val_loss: 0.2904 - val_accuracy: 0.9200
Epoch 65/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0548 - accurac
y: 0.9783 - val_loss: 0.5904 - val_accuracy: 0.8702
Epoch 66/100
```

```
375/375 [==============================] - 2s 5ms/step - loss: 0.0473 - accurac
y: 0.9824 - val_loss: 0.3776 - val_accuracy: 0.8992
Epoch 67/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0502 - accurac
y: 0.9814 - val_loss: 0.3840 - val_accuracy: 0.8945
Epoch 68/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0520 - accurac
y: 0.9792 - val_loss: 0.3624 - val_accuracy: 0.9008
Epoch 69/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0442 - accurac
y: 0.9833 - val_loss: 0.3971 - val_accuracy: 0.8923
Epoch 70/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0487 - accurac
y: 0.9813 - val_loss: 0.3140 - val_accuracy: 0.9150
Epoch 71/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0476 - accurac
y: 0.9821 - val_loss: 0.3453 - val_accuracy: 0.9107
Epoch 72/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0476 - accurac
y: 0.9820 - val_loss: 0.3474 - val_accuracy: 0.9057
Epoch 73/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0462 - accurac
y: 0.9834 - val_loss: 0.3303 - val_accuracy: 0.9110
Epoch 74/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0461 - accurac
y: 0.9825 - val_loss: 0.3433 - val_accuracy: 0.9128
Epoch 75/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0446 - accurac
y: 0.9842 - val_loss: 0.4413 - val_accuracy: 0.8982
Epoch 76/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0472 - accurac
y: 0.9824 - val_loss: 0.2993 - val_accuracy: 0.9225
Epoch 77/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0502 - accurac
y: 0.9804 - val_loss: 0.3098 - val_accuracy: 0.9173
Epoch 78/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0467 - accurac
y: 0.9826 - val_loss: 0.3046 - val_accuracy: 0.9202
Epoch 79/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0390 - accurac
y: 0.9854 - val_loss: 0.3547 - val_accuracy: 0.9062
Epoch 80/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0439 - accurac
y: 0.9826 - val_loss: 0.3314 - val_accuracy: 0.9175
Epoch 81/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0422 - accurac
y: 0.9838 - val_loss: 0.3085 - val_accuracy: 0.9188
Epoch 82/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0429 - accurac
y: 0.9825 - val_loss: 0.3204 - val_accuracy: 0.9150
Epoch 83/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0441 - accurac
y: 0.9837 - val_loss: 0.3140 - val_accuracy: 0.9143
Epoch 84/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0384 - accurac
y: 0.9849 - val_loss: 0.4141 - val_accuracy: 0.9002
Epoch 85/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0428 - accurac
y: 0.9840 - val_loss: 0.3773 - val_accuracy: 0.9057
Epoch 86/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0499 - accurac
y: 0.9815 - val_loss: 0.3140 - val_accuracy: 0.9185
Epoch 87/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0380 - accurac
y: 0.9857 - val_loss: 0.3454 - val_accuracy: 0.9097
```

```
Epoch 88/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0380 - accurac
y: 0.9864 - val_loss: 0.3738 - val_accuracy: 0.9028
Epoch 89/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0419 - accurac
y: 0.9848 - val_loss: 0.3188 - val_accuracy: 0.9223
Epoch 90/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0383 - accurac
y: 0.9854 - val_loss: 0.3551 - val_accuracy: 0.9117
Epoch 91/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0382 - accurac
y: 0.9861 - val_loss: 0.3623 - val_accuracy: 0.9093
Epoch 92/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0384 - accurac
y: 0.9851 - val_loss: 0.4293 - val_accuracy: 0.8970
Epoch 93/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0465 - accurac
y: 0.9828 - val_loss: 0.3228 - val_accuracy: 0.9202
Epoch 94/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0379 - accurac
y: 0.9863 - val_loss: 0.3726 - val_accuracy: 0.9092
Epoch 95/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0383 - accurac
y: 0.9853 - val_loss: 0.3826 - val_accuracy: 0.9075
Epoch 96/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0412 - accurac
y: 0.9857 - val_loss: 0.5077 - val_accuracy: 0.8847
Epoch 97/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0379 - accurac
y: 0.9864 - val_loss: 0.3358 - val_accuracy: 0.9163
Epoch 98/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0371 - accurac
y: 0.9868 - val_loss: 0.3197 - val_accuracy: 0.9202
Epoch 99/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0391 - accurac
y: 0.9858 - val_loss: 0.3410 - val_accuracy: 0.9178
Epoch 100/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0358 - accurac
y: 0.9870 - val_loss: 0.3383 - val_accuracy: 0.9195
CPU times: user 2min 53s, sys: 27.2 s, total: 3min 20s
Wall time: 2min 57s
```

In [ ]:

```
%%time
with tf.device('/device:GPU:0'):
  test_loss, test_acc = model_brief.evaluate(X_test,  y_test, verbose=2)
print("Test of accuracy of brief model", test_acc)
```

```
188/188 - 0s - loss: 1.0568 - accuracy: 0.8957
Test of accuracy of brief model 0.8956666588783264
CPU times: user 361 ms, sys: 32.9 ms, total: 394 ms
Wall time: 298 ms
```

In [ ]:

```
history_list.append(history_deeper)
```

In [ ]:

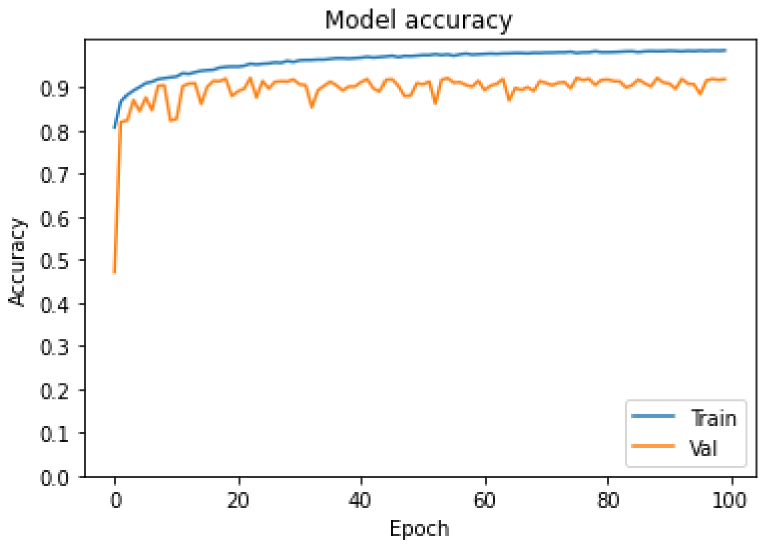
```
print_accuracy(model, y_train, y_val, y_test)
```

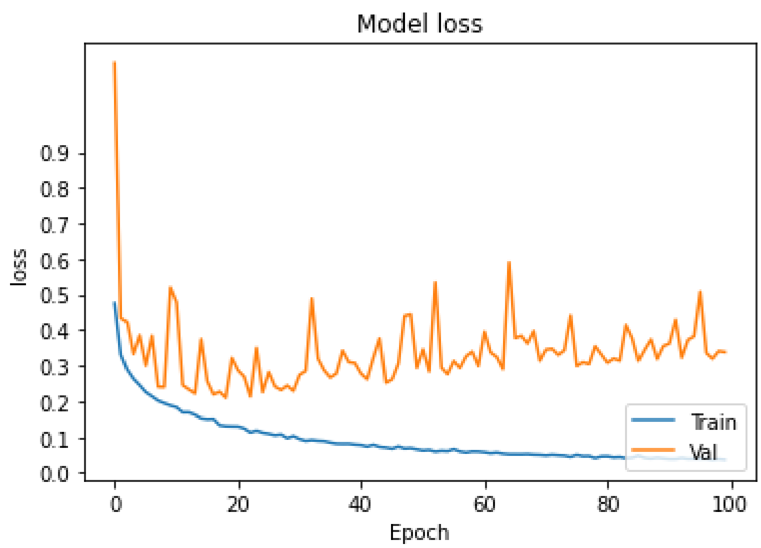
```
Train Set Accuracy:      99.53
Train Set Precision:     1.0
Train Set Recall:        1.0
Train Set F score:       1.0
```

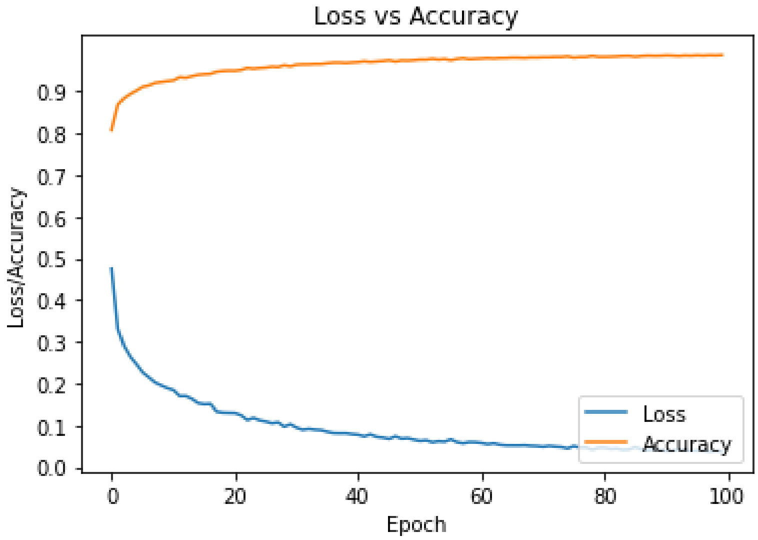
```
Val Set Accuracy:        91.95
Val Set Precision:       0.92
Val Set Recall:          0.92
Val Set F score:         0.92

Test Set Accuracy:       91.77
Test Set Precision:      0.92
Test Set Recall:         0.92
Test Set F score:        0.92
```

In [ ]:

```
accuracy_loss_plot(history_deeper)
```







Explanation of deeper Model plots

Accuracy plots

This model achieves an accuracy of up to 99.53% on the training data. However, it achieves 91.95% accuracy on the validation and 91.77% on the test set. From the accuracy graph above, there is a significant gap between the training and validation accuracy per epoch. This is much smaller than the gap in the brief model.

Loss plots

There is also a significant gap between the losses per epoch in the training and validation sets. This represents possible overfitting.

Observations

- This model has less overfitting than the previous model because of the addition of dropout layers.

Data Augmentation

This a technique used to increase the diversity of your training set by applying random (but realistic) transformations such as image rotation. Increasing the data in this way, could make the model better at generalizing to new data. In this way, data augmentation acts as a regularizer.

Types of Augmentation Used to update our Deeper model

1. Rotation:

   A rotation augmentation randomly rotates the image clockwise by a given number of degrees from 0 to 360. Our model supplies random rotations via the rotation_range argument, with rotations to the image between 0 and 8 degrees.

2. Zoom:

A zoom augmentation randomly zooms the image in and either adds new pixel values around the image or interpolates pixel values respectively.

3. Shear:

   Shear' means that the image will be distorted along an axis, mostly to create or rectify the perception angles. For example, if a image appears as a rectangle, applying would make it resemble a parallelogram. It's usually used to augment images so that computers can see how humans see things from different angles.

4. Flip:

   An image flip means reversing the rows or columns of pixels in the case of a vertical or horizontal flip respectively. In this model, we make use of a vertical flip.

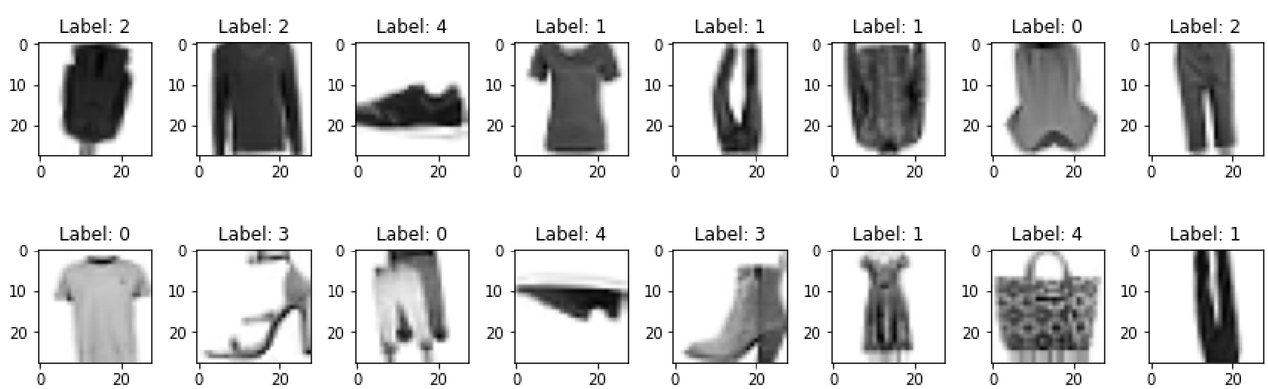
5. Width/Height Shift:

   A shift to an image means moving all pixels of the image in one direction, such as horizontally(width shift) or vertically(horizontal shift), while keeping the image dimensions the same.

In [ ]:
```
datagen = ImageDataGenerator(
        rotation_range = 8,  # randomly rotate images in the range (degrees, 0 t
        zoom_range = 0.1, # Randomly zoom image
        shear_range = 0.3,# shear angle in counter-clockwise direction in degree
        width_shift_range=0.08,  # randomly shift images horizontally (fraction
        height_shift_range=0.08,  # randomly shift images vertically (fraction o
        vertical_flip=True)  # randomly flip images
```

In [ ]:
```
datagen.fit(X_train)
```

In [ ]:
```
%%time
plot_augmented_data(X_train, y_train)
```




```
CPU times: user 1.8 s, sys: 139 ms, total: 1.94 s
Wall time: 1.78 s
```

This plot above shows the effect of the data augmentation techniques applied above.

In [ ]:
```
%%time
```

```
batch_size = 128
epochs = 100
reduce_lr = LearningRateScheduler(lambda x: 1e-3 * 0.9 ** x)
with tf.device('/device:GPU:0'):
  # Fit the Model
  history = model.fit(datagen.flow(X_train, y_train, batch_size = batch_size), e
                              validation_data = (X_val, y_val), verbose=2,
                              steps_per_epoch=X_train.shape[0] // batch_size,
                              callbacks = [reduce_lr])
```

```
Epoch 1/100
375/375 - 12s - loss: 0.6391 - accuracy: 0.7620 - val_loss: 0.3285 - val_accurac
y: 0.8707
Epoch 2/100
375/375 - 11s - loss: 0.4706 - accuracy: 0.8075 - val_loss: 0.3014 - val_accurac
y: 0.8835
Epoch 3/100
375/375 - 11s - loss: 0.4408 - accuracy: 0.8197 - val_loss: 0.3003 - val_accurac
y: 0.8823
Epoch 4/100
375/375 - 11s - loss: 0.4143 - accuracy: 0.8302 - val_loss: 0.3006 - val_accurac
y: 0.8773
Epoch 5/100
375/375 - 12s - loss: 0.4043 - accuracy: 0.8350 - val_loss: 0.3073 - val_accurac
y: 0.8728
Epoch 6/100
375/375 - 11s - loss: 0.3922 - accuracy: 0.8377 - val_loss: 0.2821 - val_accurac
y: 0.8887
Epoch 7/100
375/375 - 12s - loss: 0.3834 - accuracy: 0.8425 - val_loss: 0.2750 - val_accurac
y: 0.8942
Epoch 8/100
375/375 - 11s - loss: 0.3755 - accuracy: 0.8466 - val_loss: 0.2943 - val_accurac
y: 0.8772
Epoch 9/100
375/375 - 12s - loss: 0.3663 - accuracy: 0.8498 - val_loss: 0.3077 - val_accurac
y: 0.8712
Epoch 10/100
375/375 - 12s - loss: 0.3628 - accuracy: 0.8509 - val_loss: 0.3017 - val_accurac
y: 0.8770
Epoch 11/100
375/375 - 11s - loss: 0.3577 - accuracy: 0.8523 - val_loss: 0.4349 - val_accurac
y: 0.8250
Epoch 12/100
375/375 - 11s - loss: 0.3565 - accuracy: 0.8518 - val_loss: 0.2989 - val_accurac
y: 0.8757
Epoch 13/100
375/375 - 12s - loss: 0.3513 - accuracy: 0.8564 - val_loss: 0.2819 - val_accurac
y: 0.8848
Epoch 14/100
375/375 - 11s - loss: 0.3506 - accuracy: 0.8574 - val_loss: 0.3387 - val_accurac
y: 0.8608
Epoch 15/100
375/375 - 11s - loss: 0.3456 - accuracy: 0.8590 - val_loss: 0.3124 - val_accurac
y: 0.8713
Epoch 16/100
375/375 - 12s - loss: 0.3430 - accuracy: 0.8602 - val_loss: 0.3257 - val_accurac
y: 0.8652
Epoch 17/100
375/375 - 11s - loss: 0.3409 - accuracy: 0.8603 - val_loss: 0.3051 - val_accurac
y: 0.8732
Epoch 18/100
375/375 - 11s - loss: 0.3387 - accuracy: 0.8620 - val_loss: 0.3661 - val_accurac
y: 0.8497
```

```
Epoch 19/100
375/375 - 12s - loss: 0.3391 - accuracy: 0.8612 - val_loss: 0.3303 - val_accurac
y: 0.8635
Epoch 20/100
375/375 - 11s - loss: 0.3355 - accuracy: 0.8639 - val_loss: 0.3223 - val_accurac
y: 0.8660
Epoch 21/100
375/375 - 11s - loss: 0.3329 - accuracy: 0.8642 - val_loss: 0.3195 - val_accurac
y: 0.8683
Epoch 22/100
375/375 - 12s - loss: 0.3295 - accuracy: 0.8658 - val_loss: 0.3306 - val_accurac
y: 0.8617
Epoch 23/100
375/375 - 11s - loss: 0.3327 - accuracy: 0.8638 - val_loss: 0.3181 - val_accurac
y: 0.8698
Epoch 24/100
375/375 - 11s - loss: 0.3337 - accuracy: 0.8636 - val_loss: 0.3452 - val_accurac
y: 0.8570
Epoch 25/100
375/375 - 11s - loss: 0.3321 - accuracy: 0.8625 - val_loss: 0.3414 - val_accurac
y: 0.8578
Epoch 26/100
375/375 - 11s - loss: 0.3292 - accuracy: 0.8658 - val_loss: 0.3286 - val_accurac
y: 0.8635
Epoch 27/100
375/375 - 11s - loss: 0.3324 - accuracy: 0.8643 - val_loss: 0.3107 - val_accurac
y: 0.8720
Epoch 28/100
375/375 - 11s - loss: 0.3248 - accuracy: 0.8676 - val_loss: 0.3242 - val_accurac
y: 0.8655
Epoch 29/100
375/375 - 11s - loss: 0.3281 - accuracy: 0.8666 - val_loss: 0.3244 - val_accurac
y: 0.8655
Epoch 30/100
375/375 - 11s - loss: 0.3273 - accuracy: 0.8673 - val_loss: 0.3314 - val_accurac
y: 0.8642
Epoch 31/100
375/375 - 12s - loss: 0.3247 - accuracy: 0.8670 - val_loss: 0.3282 - val_accurac
y: 0.8648
Epoch 32/100
375/375 - 11s - loss: 0.3277 - accuracy: 0.8656 - val_loss: 0.3221 - val_accurac
y: 0.8667
Epoch 33/100
375/375 - 12s - loss: 0.3282 - accuracy: 0.8652 - val_loss: 0.3183 - val_accurac
y: 0.8690
Epoch 34/100
375/375 - 11s - loss: 0.3231 - accuracy: 0.8685 - val_loss: 0.3153 - val_accurac
y: 0.8697
Epoch 35/100
375/375 - 11s - loss: 0.3271 - accuracy: 0.8649 - val_loss: 0.3332 - val_accurac
y: 0.8612
Epoch 36/100
375/375 - 12s - loss: 0.3302 - accuracy: 0.8645 - val_loss: 0.3346 - val_accurac
y: 0.8600
Epoch 37/100
375/375 - 11s - loss: 0.3260 - accuracy: 0.8659 - val_loss: 0.3304 - val_accurac
y: 0.8643
Epoch 38/100
375/375 - 11s - loss: 0.3253 - accuracy: 0.8684 - val_loss: 0.3279 - val_accurac
y: 0.8642
Epoch 39/100
375/375 - 12s - loss: 0.3202 - accuracy: 0.8689 - val_loss: 0.3248 - val_accurac
y: 0.8652
Epoch 40/100
375/375 - 11s - loss: 0.3255 - accuracy: 0.8655 - val_loss: 0.3261 - val_accurac
```

```
y: 0.8637
Epoch 41/100
375/375 - 11s - loss: 0.3263 - accuracy: 0.8668 - val_loss: 0.3295 - val_accurac
y: 0.8625
Epoch 42/100
375/375 - 11s - loss: 0.3264 - accuracy: 0.8672 - val_loss: 0.3219 - val_accurac
y: 0.8665
Epoch 43/100
375/375 - 11s - loss: 0.3246 - accuracy: 0.8680 - val_loss: 0.3270 - val_accurac
y: 0.8648
Epoch 44/100
375/375 - 11s - loss: 0.3242 - accuracy: 0.8678 - val_loss: 0.3258 - val_accurac
y: 0.8653
Epoch 45/100
375/375 - 12s - loss: 0.3254 - accuracy: 0.8669 - val_loss: 0.3305 - val_accurac
y: 0.8638
Epoch 46/100
375/375 - 11s - loss: 0.3236 - accuracy: 0.8670 - val_loss: 0.3248 - val_accurac
y: 0.8655
Epoch 47/100
375/375 - 11s - loss: 0.3229 - accuracy: 0.8683 - val_loss: 0.3250 - val_accurac
y: 0.8657
Epoch 48/100
375/375 - 11s - loss: 0.3224 - accuracy: 0.8685 - val_loss: 0.3260 - val_accurac
y: 0.8655
Epoch 49/100
375/375 - 11s - loss: 0.3251 - accuracy: 0.8671 - val_loss: 0.3270 - val_accurac
y: 0.8652
Epoch 50/100
375/375 - 11s - loss: 0.3226 - accuracy: 0.8698 - val_loss: 0.3277 - val_accurac
y: 0.8647
Epoch 51/100
375/375 - 11s - loss: 0.3228 - accuracy: 0.8687 - val_loss: 0.3243 - val_accurac
y: 0.8655
Epoch 52/100
375/375 - 11s - loss: 0.3231 - accuracy: 0.8676 - val_loss: 0.3260 - val_accurac
y: 0.8643
Epoch 53/100
375/375 - 11s - loss: 0.3238 - accuracy: 0.8681 - val_loss: 0.3267 - val_accurac
y: 0.8647
Epoch 54/100
375/375 - 11s - loss: 0.3208 - accuracy: 0.8697 - val_loss: 0.3263 - val_accurac
y: 0.8650
Epoch 55/100
375/375 - 11s - loss: 0.3273 - accuracy: 0.8651 - val_loss: 0.3263 - val_accurac
y: 0.8662
Epoch 56/100
375/375 - 11s - loss: 0.3204 - accuracy: 0.8691 - val_loss: 0.3273 - val_accurac
y: 0.8655
Epoch 57/100
375/375 - 11s - loss: 0.3266 - accuracy: 0.8673 - val_loss: 0.3283 - val_accurac
y: 0.8643
Epoch 58/100
375/375 - 12s - loss: 0.3239 - accuracy: 0.8681 - val_loss: 0.3280 - val_accurac
y: 0.8645
Epoch 59/100
375/375 - 11s - loss: 0.3232 - accuracy: 0.8682 - val_loss: 0.3269 - val_accurac
y: 0.8652
Epoch 60/100
375/375 - 11s - loss: 0.3229 - accuracy: 0.8679 - val_loss: 0.3285 - val_accurac
y: 0.8642
Epoch 61/100
375/375 - 11s - loss: 0.3250 - accuracy: 0.8662 - val_loss: 0.3281 - val_accurac
y: 0.8640
Epoch 62/100
```

```
375/375 - 11s - loss: 0.3269 - accuracy: 0.8662 - val_loss: 0.3284 - val_accurac
y: 0.8642
Epoch 63/100
375/375 - 11s - loss: 0.3230 - accuracy: 0.8678 - val_loss: 0.3276 - val_accurac
y: 0.8650
Epoch 64/100
375/375 - 11s - loss: 0.3206 - accuracy: 0.8686 - val_loss: 0.3273 - val_accurac
y: 0.8652
Epoch 65/100
375/375 - 11s - loss: 0.3247 - accuracy: 0.8660 - val_loss: 0.3291 - val_accurac
y: 0.8640
Epoch 66/100
375/375 - 11s - loss: 0.3221 - accuracy: 0.8689 - val_loss: 0.3284 - val_accurac
y: 0.8643
Epoch 67/100
375/375 - 11s - loss: 0.3212 - accuracy: 0.8696 - val_loss: 0.3277 - val_accurac
y: 0.8647
Epoch 68/100
375/375 - 11s - loss: 0.3250 - accuracy: 0.8673 - val_loss: 0.3257 - val_accurac
y: 0.8653
Epoch 69/100
375/375 - 11s - loss: 0.3246 - accuracy: 0.8676 - val_loss: 0.3266 - val_accurac
y: 0.8650
Epoch 70/100
375/375 - 11s - loss: 0.3255 - accuracy: 0.8671 - val_loss: 0.3266 - val_accurac
y: 0.8653
Epoch 71/100
375/375 - 11s - loss: 0.3203 - accuracy: 0.8699 - val_loss: 0.3279 - val_accurac
y: 0.8648
Epoch 72/100
375/375 - 11s - loss: 0.3221 - accuracy: 0.8679 - val_loss: 0.3268 - val_accurac
y: 0.8650
Epoch 73/100
375/375 - 11s - loss: 0.3242 - accuracy: 0.8680 - val_loss: 0.3259 - val_accurac
y: 0.8657
Epoch 74/100
375/375 - 11s - loss: 0.3236 - accuracy: 0.8679 - val_loss: 0.3266 - val_accurac
y: 0.8653
Epoch 75/100
375/375 - 11s - loss: 0.3252 - accuracy: 0.8664 - val_loss: 0.3249 - val_accurac
y: 0.8660
Epoch 76/100
375/375 - 11s - loss: 0.3197 - accuracy: 0.8687 - val_loss: 0.3276 - val_accurac
y: 0.8652
Epoch 77/100
375/375 - 11s - loss: 0.3265 - accuracy: 0.8649 - val_loss: 0.3287 - val_accurac
y: 0.8635
Epoch 78/100
375/375 - 11s - loss: 0.3227 - accuracy: 0.8669 - val_loss: 0.3259 - val_accurac
y: 0.8660
Epoch 79/100
375/375 - 11s - loss: 0.3278 - accuracy: 0.8679 - val_loss: 0.3270 - val_accurac
y: 0.8647
Epoch 80/100
375/375 - 11s - loss: 0.3240 - accuracy: 0.8671 - val_loss: 0.3273 - val_accurac
y: 0.8652
Epoch 81/100
375/375 - 12s - loss: 0.3246 - accuracy: 0.8685 - val_loss: 0.3281 - val_accurac
y: 0.8650
Epoch 82/100
375/375 - 11s - loss: 0.3193 - accuracy: 0.8712 - val_loss: 0.3257 - val_accurac
y: 0.8658
Epoch 83/100
375/375 - 11s - loss: 0.3249 - accuracy: 0.8671 - val_loss: 0.3273 - val_accurac
y: 0.8648
```

```
Epoch 84/100
375/375 - 11s - loss: 0.3259 - accuracy: 0.8670 - val_loss: 0.3272 - val_accurac
y: 0.8650
Epoch 85/100
375/375 - 11s - loss: 0.3241 - accuracy: 0.8671 - val_loss: 0.3268 - val_accurac
y: 0.8652
Epoch 86/100
375/375 - 12s - loss: 0.3189 - accuracy: 0.8704 - val_loss: 0.3251 - val_accurac
y: 0.8662
Epoch 87/100
375/375 - 11s - loss: 0.3239 - accuracy: 0.8694 - val_loss: 0.3268 - val_accurac
y: 0.8653
Epoch 88/100
375/375 - 11s - loss: 0.3187 - accuracy: 0.8708 - val_loss: 0.3276 - val_accurac
y: 0.8650
Epoch 89/100
375/375 - 11s - loss: 0.3231 - accuracy: 0.8669 - val_loss: 0.3284 - val_accurac
y: 0.8643
Epoch 90/100
375/375 - 11s - loss: 0.3239 - accuracy: 0.8675 - val_loss: 0.3294 - val_accurac
y: 0.8637
Epoch 91/100
375/375 - 12s - loss: 0.3216 - accuracy: 0.8665 - val_loss: 0.3253 - val_accurac
y: 0.8663
Epoch 92/100
375/375 - 11s - loss: 0.3226 - accuracy: 0.8685 - val_loss: 0.3268 - val_accurac
y: 0.8652
Epoch 93/100
375/375 - 11s - loss: 0.3188 - accuracy: 0.8691 - val_loss: 0.3247 - val_accurac
y: 0.8663
Epoch 94/100
375/375 - 11s - loss: 0.3241 - accuracy: 0.8686 - val_loss: 0.3264 - val_accurac
y: 0.8652
Epoch 95/100
375/375 - 11s - loss: 0.3222 - accuracy: 0.8683 - val_loss: 0.3271 - val_accurac
y: 0.8655
Epoch 96/100
375/375 - 11s - loss: 0.3245 - accuracy: 0.8683 - val_loss: 0.3281 - val_accurac
y: 0.8642
Epoch 97/100
375/375 - 11s - loss: 0.3246 - accuracy: 0.8669 - val_loss: 0.3267 - val_accurac
y: 0.8653
Epoch 98/100
375/375 - 11s - loss: 0.3264 - accuracy: 0.8674 - val_loss: 0.3280 - val_accurac
y: 0.8647
Epoch 99/100
375/375 - 11s - loss: 0.3247 - accuracy: 0.8664 - val_loss: 0.3273 - val_accurac
y: 0.8647
Epoch 100/100
375/375 - 11s - loss: 0.3254 - accuracy: 0.8682 - val_loss: 0.3261 - val_accurac
y: 0.8657
CPU times: user 21min 11s, sys: 13.9 s, total: 21min 25s
Wall time: 18min 56s
```

In [ ]:

```
%%time
with tf.device('/device:GPU:0'):
  test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)
print(test_acc)
```

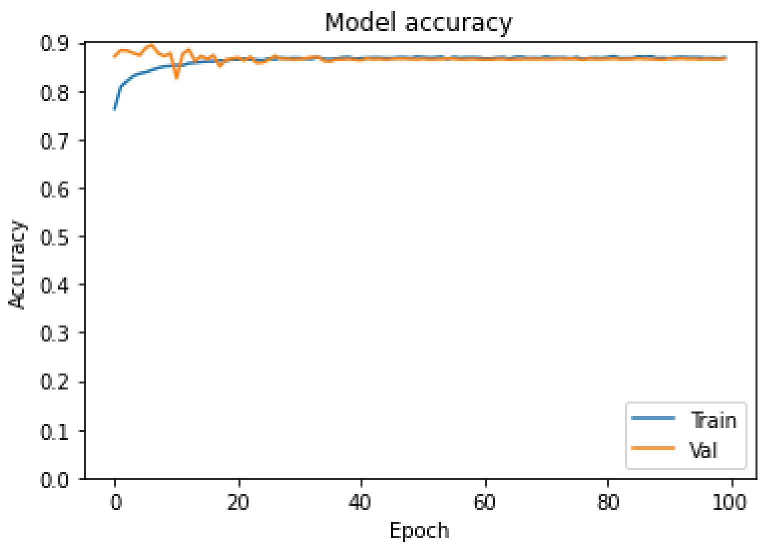
```
188/188 - 0s - loss: 0.3174 - accuracy: 0.8720
0.871999979019165
CPU times: user 428 ms, sys: 25.1 ms, total: 453 ms
Wall time: 346 ms
```

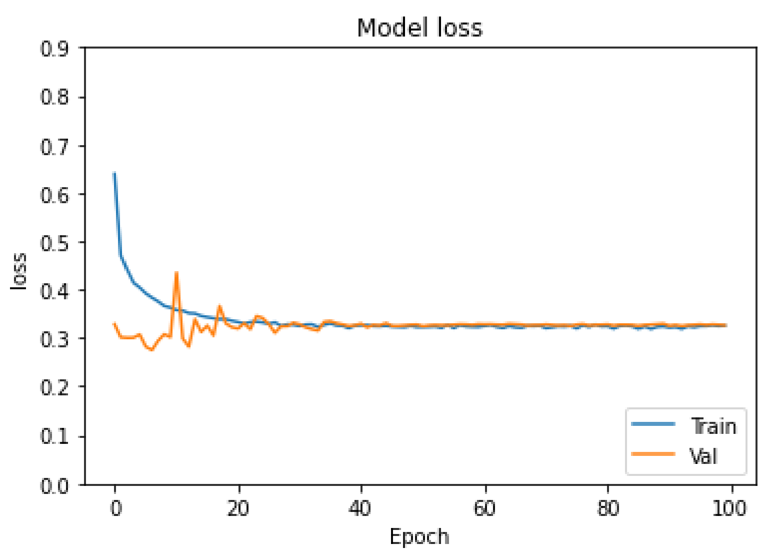
In [ ]:
```
history_list.append(history)
```

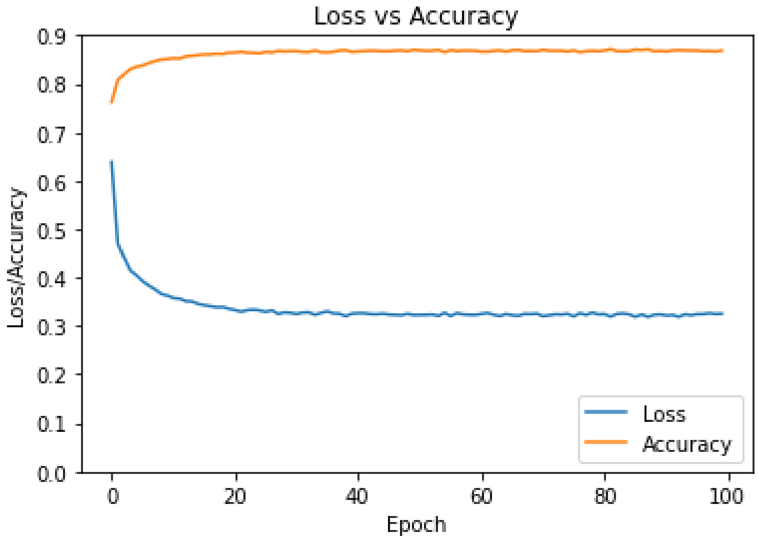
In [ ]:
```
print_accuracy(model, y_train, y_val, y_test)
```

```
Train Set Accuracy:      89.49
Train Set Precision:     0.9
Train Set Recall:        0.89
Train Set F score:       0.89

Val Set Accuracy:        86.57
Val Set Precision:       0.87
Val Set Recall:          0.87
Val Set F score:         0.87

Test Set Accuracy:       87.2
Test Set Precision:      0.87
Test Set Recall:         0.87
Test Set F score:        0.87
```

In [ ]:
```
accuracy_loss_plot(history)
```







Discussion on Data Augmentation Perfomance

From the accuracy and loss plots above, the gap between the training and validation sets is quite small. Augmenting the data had a good regularization effect on the data. The training accuracy is 88.35%% and the validation and test accuracy are 86.38% and 86.43% respectively.

Even though there is little overfitting after applying these augmentations, the accuracy is worse than it was before Data Augmentation is applied. This may be because, if there are types of augmentation that are not relevant to the test set, certain types of data augmentation may not be effective. For example, for an object recognition model that will mimic how humans see in real life, these data augmentation techniques above may be relevant in that context.

In our context however, the images in the test set are not likely to be zoomed, sheared or flipped. In the next section, we will look at a different data augmentation technique that is more relevant in our context.

Improving Accuracy of Deeper Model

Elastic distortion

Elastic distortion is another method of data augmentation, as opposed to affine distortion which is the method Keras uses. Elastic distortion does a good job of mimicking variations in human hand writing. A method for applying elastic distortion to the MNIST data set is described by Simard, Steinkraus, and Plattin [2]. We applied this to our Fashion MNIST model, because it is similar to MNIST.

The method outline:

- Create random displacement fields for height and width, with values randomly sampled from $\mathrm{unif}(-1,1)$. A displacement field defines a direction and magnitude to move a pixel.
- Smooth the fields with a gaussian filter. Since $\mu=0$ for $\mathrm{unif}(-1,1)$, most values will be close to 0 after the gaussian filter is applied. Thus most of the changes made by the fields will be

small (assuming the gaussian filter's sigma value is large enough).

- Multiply the fields by a scaling factor to control intensity of the deformations.
- Use interpolation to apply the displacement fields to the image.

Cosine Annealing

Cosine annealing [3] [4] is a relatively new learning rate annealing technique that does a more thorough job of exploring the model's solution space by using warm restarts to break out of local minimums. As the learning rate decreases, the model gets more precise but may also get stuck in a particular state. Warm restarts raise the learning rate to get the model unstuck. I found that, as long as the model doesn't overfit on the training set too much, continual warm restarts can potentially discover better and better models.

Adamax

The Adam optimizer uses an exponentially decaying weighted average estimate of the variance of the gradient in its formulation. The variance is equivalent to the second moment or L2 norm of the gradient. The L_n = norm is defined as:

$L_1 = g$

$L_2 = \sqrt{g^2}$

$L_3 = \sqrt[3]{g^3}$

$L_n = \sqrt[n]{g^n}$

Adamax $L_\infty = \sqrt[\infty]{g^\infty}$

The infinite norm is numerically stable because it has asymptotically convergent behavior (assuming g ∊ [0,1]). **AdaMax is a generalisation of Adam from the L_2 norm to the L_∞ norm.**

AdaMax is more robust to gradient update noise than Adam is, and has better numerical stability. [5]

In [ ]:
```
from scipy.ndimage.filters import gaussian_filter
from scipy.ndimage.interpolation import map_coordinates

def elastic_transform(image, alpha_range, sigma, random_state=None):
    """Elastic deformation of images as described in [Simard2003]_.
    .. [Simard2003] Simard, Steinkraus and Platt, "Best Practices for
       Convolutional Neural Networks applied to Visual Document Analysis", in
       Proc. of the International Conference on Document Analysis and
       Recognition, 2003.

   # Arguments
       image: Numpy array with shape (height, width, channels).
       alpha_range: Float for fixed value or [lower, upper] for random value fro
           Controls intensity of deformation.
       sigma: Float, sigma of gaussian filter that smooths the displacement fiel
       random_state: `numpy.random.RandomState` object for generating displaceme
```

```
    """

    if random_state is None:
        random_state = np.random.RandomState(None)

    if np.isscalar(alpha_range):
        alpha = alpha_range
    else:
        alpha = np.random.uniform(low=alpha_range[0], high=alpha_range[1])
        shape = image.shape
    dx = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma) * alpha
    dy = gaussian_filter((random_state.rand(*shape) * 2 - 1), sigma) * alpha

    x, y, z = np.meshgrid(np.arange(shape[0]), np.arange(shape[1]), np.arange(sh
    indices = np.reshape(x+dx, (-1, 1)), np.reshape(y+dy, (-1, 1)), np.reshape(z

    return map_coordinates(image, indices, order=1, mode='reflect').reshape(shap
```

In [ ]:

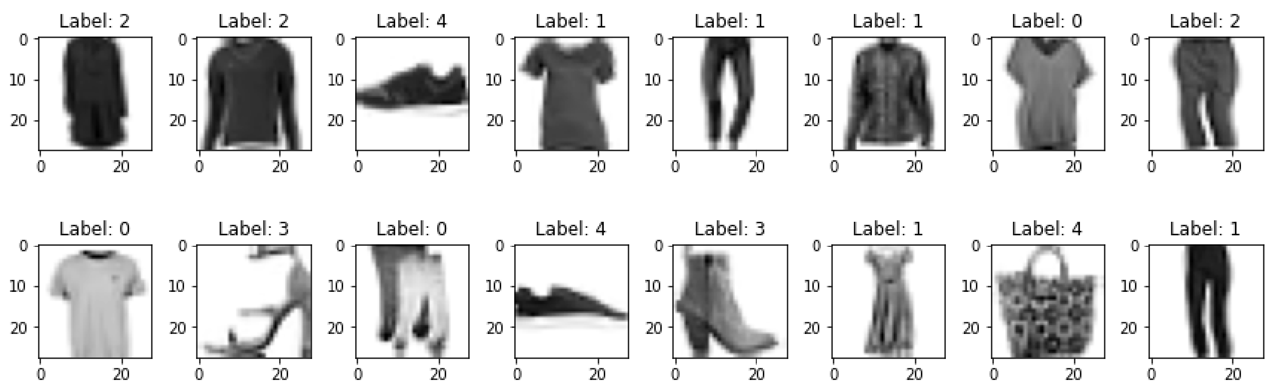
```
from keras import backend as K
class CosineAnneal(tf.keras.callbacks.Callback):
    """"Cosine annealing with warm restarts.

    As described in section 3 of "SGDR: Stochastic Gradient Descent with Warm Re

    # Arguments
        max_lr: Maximum value of learning rate range.
        min_lr: Minimum value of learning rate range.
        T: Number of epochs between warm restarts.
        T_mul: At warm restarts, multiply `T` by this amount.
    """
    def __init__(self, max_lr, min_lr, T, T_mul=1):
        self.max_lr = max_lr
        self.min_lr = min_lr
        self.T = T
        self.T_cur = 0
        self.T_mul = T_mul
        self.step = 0

    def on_batch_begin(self, batch, logs=None):
        if self.T <= self.T_cur:
            self.T *= self.T_mul
            self.T_cur = 0
            self.step = 0
        lr = self.min_lr + 0.5 * (self.max_lr - self.min_lr) * (1 + np.cos(self.
        K.set_value(self.model.optimizer.lr, lr)
        # use self.step to avoid floating point arithmetic errors at warm restar
        self.step += 1
        self.T_cur = self.step / self.params['steps']

    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        logs['lr'] = K.get_value(self.model.optimizer.lr)
```

In [ ]:

```
#using Adamax optimizer
from keras import optimizers
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=Tru
              optimizer=optimizers.Adamax(lr=0.006, beta_1=0.49, beta_2=0.999),
              metrics=['accuracy'])
```

In [ ]:
```
batch_size = 128
epochs = 100

# setup callbacks
annealer = CosineAnneal(max_lr=0.006, min_lr=0.001, T=10, T_mul=1)


# define data augmentations
datagen = ImageDataGenerator(
    height_shift_range=2,
    horizontal_flip=True,
    preprocessing_function=lambda x: elastic_transform(x, alpha_range=[10, 12],
)
```

In [ ]:
```
%%time
#plot of new augmentation technique
plot_augmented_data(X_train, y_train)
```




```
CPU times: user 1.57 s, sys: 133 ms, total: 1.7 s
Wall time: 1.57 s
```

Discussion of Data Augmentation (ii)

Elastic distortion utilises a Gaussian filter for smoothing, smoothing can result in a slight blur which is relevant to our dataset. Some images in the non-augmented data have more blur than others, making them difficult to distinguish. Adding different degrees of blur at random, using elastic distortion, helps our model to generalize better on images. If the alpha parameter is too high, it would result in extreme blurring and deformation of the original images.

In [ ]:
```
%%time
# train model
with tf.device('/device:GPU:0'):
  history_final = model.fit_generator(
    datagen.flow(X_train, y_train, batch_size=batch_size, shuffle=True),
    epochs=epochs,
    steps_per_epoch=(len(y_train) - 1) // batch_size + 1,
    validation_data=(X_val, y_val),
    callbacks=[annealer])
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.p
y:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in
```

```
a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/100
375/375 [==============================] - 48s 126ms/step - loss: 0.3330 - accur
acy: 0.8685 - val_loss: 0.2608 - val_accuracy: 0.8975
Epoch 2/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2689 - accur
acy: 0.8928 - val_loss: 0.2581 - val_accuracy: 0.8967
Epoch 3/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2557 - accur
acy: 0.8977 - val_loss: 0.2784 - val_accuracy: 0.8895
Epoch 4/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2459 - accur
acy: 0.9015 - val_loss: 0.2204 - val_accuracy: 0.9130
Epoch 5/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2343 - accur
acy: 0.9052 - val_loss: 0.2810 - val_accuracy: 0.8880
Epoch 6/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2335 - accur
acy: 0.9054 - val_loss: 0.2347 - val_accuracy: 0.9078
Epoch 7/100
375/375 [==============================] - 47s 127ms/step - loss: 0.2222 - accur
acy: 0.9094 - val_loss: 0.2410 - val_accuracy: 0.9012
Epoch 8/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2155 - accur
acy: 0.9146 - val_loss: 0.2353 - val_accuracy: 0.9037
Epoch 9/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2083 - accur
acy: 0.9165 - val_loss: 0.2147 - val_accuracy: 0.9165
Epoch 10/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2134 - accur
acy: 0.9143 - val_loss: 0.2198 - val_accuracy: 0.9140
Epoch 11/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2309 - accur
acy: 0.9064 - val_loss: 0.2355 - val_accuracy: 0.9103
Epoch 12/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2326 - accur
acy: 0.9066 - val_loss: 0.2968 - val_accuracy: 0.8803
Epoch 13/100
375/375 [==============================] - 46s 124ms/step - loss: 0.2266 - accur
acy: 0.9079 - val_loss: 0.2687 - val_accuracy: 0.8895
Epoch 14/100
375/375 [==============================] - 46s 123ms/step - loss: 0.2327 - accur
acy: 0.9053 - val_loss: 0.2322 - val_accuracy: 0.9062
Epoch 15/100
375/375 [==============================] - 46s 123ms/step - loss: 0.2206 - accur
acy: 0.9105 - val_loss: 0.2425 - val_accuracy: 0.9063
Epoch 16/100
375/375 [==============================] - 46s 122ms/step - loss: 0.2149 - accur
acy: 0.9134 - val_loss: 0.2716 - val_accuracy: 0.8887
Epoch 17/100
375/375 [==============================] - 46s 122ms/step - loss: 0.2087 - accur
acy: 0.9161 - val_loss: 0.2249 - val_accuracy: 0.9140
Epoch 18/100
375/375 [==============================] - 46s 122ms/step - loss: 0.2024 - accur
acy: 0.9185 - val_loss: 0.2305 - val_accuracy: 0.9110
Epoch 19/100
375/375 [==============================] - 46s 123ms/step - loss: 0.1981 - accur
acy: 0.9205 - val_loss: 0.2140 - val_accuracy: 0.9180
Epoch 20/100
375/375 [==============================] - 47s 124ms/step - loss: 0.1948 - accur
acy: 0.9201 - val_loss: 0.2143 - val_accuracy: 0.9182
Epoch 21/100
375/375 [==============================] - 47s 125ms/step - loss: 0.2206 - accur
acy: 0.9133 - val_loss: 0.2467 - val_accuracy: 0.9057
```

```
Epoch 22/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2209 - accur
acy: 0.9112 - val_loss: 0.2752 - val_accuracy: 0.8925
Epoch 23/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2213 - accur
acy: 0.9113 - val_loss: 0.2813 - val_accuracy: 0.8923
Epoch 24/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2157 - accur
acy: 0.9127 - val_loss: 0.2295 - val_accuracy: 0.9103
Epoch 25/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2076 - accur
acy: 0.9155 - val_loss: 0.2568 - val_accuracy: 0.9005
Epoch 26/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2003 - accur
acy: 0.9192 - val_loss: 0.2514 - val_accuracy: 0.9022
Epoch 27/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2006 - accur
acy: 0.9202 - val_loss: 0.2255 - val_accuracy: 0.9087
Epoch 28/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1928 - accur
acy: 0.9218 - val_loss: 0.2146 - val_accuracy: 0.9158
Epoch 29/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1883 - accur
acy: 0.9264 - val_loss: 0.2113 - val_accuracy: 0.9178
Epoch 30/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1815 - accur
acy: 0.9264 - val_loss: 0.2106 - val_accuracy: 0.9170
Epoch 31/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2113 - accur
acy: 0.9165 - val_loss: 0.2829 - val_accuracy: 0.8902
Epoch 32/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2124 - accur
acy: 0.9131 - val_loss: 0.2354 - val_accuracy: 0.9080
Epoch 33/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2149 - accur
acy: 0.9136 - val_loss: 0.2280 - val_accuracy: 0.9150
Epoch 34/100
375/375 [==============================] - 47s 125ms/step - loss: 0.2007 - accur
acy: 0.9187 - val_loss: 0.2348 - val_accuracy: 0.9107
Epoch 35/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2023 - accur
acy: 0.9166 - val_loss: 0.3038 - val_accuracy: 0.8793
Epoch 36/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1983 - accur
acy: 0.9210 - val_loss: 0.2215 - val_accuracy: 0.9122
Epoch 37/100
375/375 [==============================] - 47s 125ms/step - loss: 0.1871 - accur
acy: 0.9241 - val_loss: 0.2407 - val_accuracy: 0.9055
Epoch 38/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1863 - accur
acy: 0.9249 - val_loss: 0.2361 - val_accuracy: 0.9050
Epoch 39/100
375/375 [==============================] - 47s 125ms/step - loss: 0.1784 - accur
acy: 0.9274 - val_loss: 0.2268 - val_accuracy: 0.9115
Epoch 40/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1825 - accur
acy: 0.9253 - val_loss: 0.2186 - val_accuracy: 0.9148
Epoch 41/100
375/375 [==============================] - 47s 125ms/step - loss: 0.2065 - accur
acy: 0.9164 - val_loss: 0.2564 - val_accuracy: 0.8998
Epoch 42/100
375/375 [==============================] - 48s 127ms/step - loss: 0.2022 - accur
acy: 0.9181 - val_loss: 0.2680 - val_accuracy: 0.8908
Epoch 43/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1994 - accur
```

```
acy: 0.9211 - val_loss: 0.2415 - val_accuracy: 0.9057
Epoch 44/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1948 - accur
acy: 0.9210 - val_loss: 0.2458 - val_accuracy: 0.9063
Epoch 45/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1958 - accur
acy: 0.9210 - val_loss: 0.2332 - val_accuracy: 0.9055
Epoch 46/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1897 - accur
acy: 0.9247 - val_loss: 0.2347 - val_accuracy: 0.9070
Epoch 47/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1872 - accur
acy: 0.9241 - val_loss: 0.2466 - val_accuracy: 0.9037
Epoch 48/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1880 - accur
acy: 0.9250 - val_loss: 0.2134 - val_accuracy: 0.9152
Epoch 49/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1744 - accur
acy: 0.9290 - val_loss: 0.2081 - val_accuracy: 0.9167
Epoch 50/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1726 - accur
acy: 0.9309 - val_loss: 0.2041 - val_accuracy: 0.9227
Epoch 51/100
375/375 [==============================] - 47s 126ms/step - loss: 0.2036 - accur
acy: 0.9184 - val_loss: 0.2460 - val_accuracy: 0.9060
Epoch 52/100
375/375 [==============================] - 47s 125ms/step - loss: 0.1993 - accur
acy: 0.9185 - val_loss: 0.3133 - val_accuracy: 0.8802
Epoch 53/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1938 - accur
acy: 0.9230 - val_loss: 0.2936 - val_accuracy: 0.8838
Epoch 54/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1930 - accur
acy: 0.9200 - val_loss: 0.2205 - val_accuracy: 0.9150
Epoch 55/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1898 - accur
acy: 0.9226 - val_loss: 0.2180 - val_accuracy: 0.9147
Epoch 56/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1864 - accur
acy: 0.9239 - val_loss: 0.2446 - val_accuracy: 0.9083
Epoch 57/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1830 - accur
acy: 0.9255 - val_loss: 0.2153 - val_accuracy: 0.9148
Epoch 58/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1741 - accur
acy: 0.9303 - val_loss: 0.2085 - val_accuracy: 0.9200
Epoch 59/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1738 - accur
acy: 0.9293 - val_loss: 0.2297 - val_accuracy: 0.9105
Epoch 60/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1738 - accur
acy: 0.9297 - val_loss: 0.2152 - val_accuracy: 0.9158
Epoch 61/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1873 - accur
acy: 0.9230 - val_loss: 0.2439 - val_accuracy: 0.9050
Epoch 62/100
375/375 [==============================] - 47s 125ms/step - loss: 0.1988 - accur
acy: 0.9201 - val_loss: 0.2552 - val_accuracy: 0.9023
Epoch 63/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1904 - accur
acy: 0.9243 - val_loss: 0.2826 - val_accuracy: 0.8877
Epoch 64/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1942 - accur
acy: 0.9216 - val_loss: 0.2865 - val_accuracy: 0.8862
Epoch 65/100
```

```
375/375 [==============================] - 47s 126ms/step - loss: 0.1889 - accur
acy: 0.9243 - val_loss: 0.2208 - val_accuracy: 0.9118
Epoch 66/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1824 - accur
acy: 0.9270 - val_loss: 0.2171 - val_accuracy: 0.9150
Epoch 67/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1758 - accur
acy: 0.9285 - val_loss: 0.2455 - val_accuracy: 0.9047
Epoch 68/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1698 - accur
acy: 0.9305 - val_loss: 0.2130 - val_accuracy: 0.9152
Epoch 69/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1734 - accur
acy: 0.9307 - val_loss: 0.2212 - val_accuracy: 0.9147
Epoch 70/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1665 - accur
acy: 0.9327 - val_loss: 0.2126 - val_accuracy: 0.9183
Epoch 71/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1864 - accur
acy: 0.9254 - val_loss: 0.2201 - val_accuracy: 0.9157
Epoch 72/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1871 - accur
acy: 0.9226 - val_loss: 0.2430 - val_accuracy: 0.9047
Epoch 73/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1887 - accur
acy: 0.9252 - val_loss: 0.2323 - val_accuracy: 0.9108
Epoch 74/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1856 - accur
acy: 0.9237 - val_loss: 0.2221 - val_accuracy: 0.9112
Epoch 75/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1829 - accur
acy: 0.9259 - val_loss: 0.2150 - val_accuracy: 0.9183
Epoch 76/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1761 - accur
acy: 0.9286 - val_loss: 0.2141 - val_accuracy: 0.9160
Epoch 77/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1739 - accur
acy: 0.9303 - val_loss: 0.2123 - val_accuracy: 0.9182
Epoch 78/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1691 - accur
acy: 0.9311 - val_loss: 0.2129 - val_accuracy: 0.9200
Epoch 79/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1699 - accur
acy: 0.9323 - val_loss: 0.2112 - val_accuracy: 0.9215
Epoch 80/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1688 - accur
acy: 0.9319 - val_loss: 0.2088 - val_accuracy: 0.9225
Epoch 81/100
375/375 [==============================] - 48s 129ms/step - loss: 0.1861 - accur
acy: 0.9250 - val_loss: 0.2227 - val_accuracy: 0.9102
Epoch 82/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1901 - accur
acy: 0.9254 - val_loss: 0.2209 - val_accuracy: 0.9192
Epoch 83/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1819 - accur
acy: 0.9278 - val_loss: 0.2358 - val_accuracy: 0.9083
Epoch 84/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1835 - accur
acy: 0.9269 - val_loss: 0.2359 - val_accuracy: 0.9070
Epoch 85/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1796 - accur
acy: 0.9262 - val_loss: 0.2161 - val_accuracy: 0.9165
Epoch 86/100
375/375 [==============================] - 48s 129ms/step - loss: 0.1784 - accur
acy: 0.9282 - val_loss: 0.2567 - val_accuracy: 0.9000
```

```
Epoch 87/100
375/375 [==============================] - 48s 129ms/step - loss: 0.1633 - accur
acy: 0.9345 - val_loss: 0.2058 - val_accuracy: 0.9212
Epoch 88/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1672 - accur
acy: 0.9319 - val_loss: 0.2101 - val_accuracy: 0.9202
Epoch 89/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1609 - accur
acy: 0.9349 - val_loss: 0.2122 - val_accuracy: 0.9223
Epoch 90/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1579 - accur
acy: 0.9374 - val_loss: 0.2025 - val_accuracy: 0.9267
Epoch 91/100
375/375 [==============================] - 47s 126ms/step - loss: 0.1828 - accur
acy: 0.9267 - val_loss: 0.2320 - val_accuracy: 0.9108
Epoch 92/100
375/375 [==============================] - 48s 127ms/step - loss: 0.1830 - accur
acy: 0.9268 - val_loss: 0.2183 - val_accuracy: 0.9173
Epoch 93/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1844 - accur
acy: 0.9241 - val_loss: 0.2336 - val_accuracy: 0.9115
Epoch 94/100
375/375 [==============================] - 47s 127ms/step - loss: 0.1826 - accur
acy: 0.9258 - val_loss: 0.2322 - val_accuracy: 0.9093
Epoch 95/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1751 - accur
acy: 0.9294 - val_loss: 0.2157 - val_accuracy: 0.9160
Epoch 96/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1667 - accur
acy: 0.9338 - val_loss: 0.2176 - val_accuracy: 0.9162
Epoch 97/100
375/375 [==============================] - 48s 129ms/step - loss: 0.1613 - accur
acy: 0.9345 - val_loss: 0.2047 - val_accuracy: 0.9235
Epoch 98/100
375/375 [==============================] - 48s 129ms/step - loss: 0.1657 - accur
acy: 0.9328 - val_loss: 0.2050 - val_accuracy: 0.9205
Epoch 99/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1599 - accur
acy: 0.9351 - val_loss: 0.2000 - val_accuracy: 0.9223
Epoch 100/100
375/375 [==============================] - 48s 128ms/step - loss: 0.1591 - accur
acy: 0.9354 - val_loss: 0.2201 - val_accuracy: 0.9133
CPU times: user 1h 20min 43s, sys: 1min 7s, total: 1h 21min 51s
Wall time: 1h 19min 3s
```

In [ ]:

```
%%time
#test accuracy of model brief with augmented data
with tf.device('/device:GPU:0'):
  test_loss, test_acc = model.evaluate(X_test,  y_test, verbose=2)
print("Test Accuracy of Brief Model with Augmented Data", test_acc)
```

```
188/188 - 0s - loss: 0.2161 - accuracy: 0.9132
Test Accuracy of Brief Model with Augmented Data 0.9131666421890259
CPU times: user 398 ms, sys: 27.5 ms, total: 425 ms
Wall time: 336 ms
```

In [ ]:

```
history_list.append(history_final)
```

In [94]:

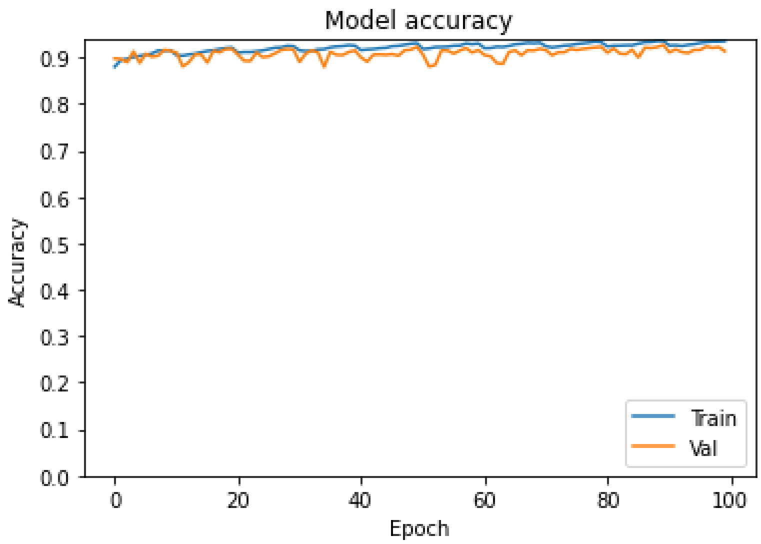
```
print_accuracy(model, y_train, y_val, y_test)
```

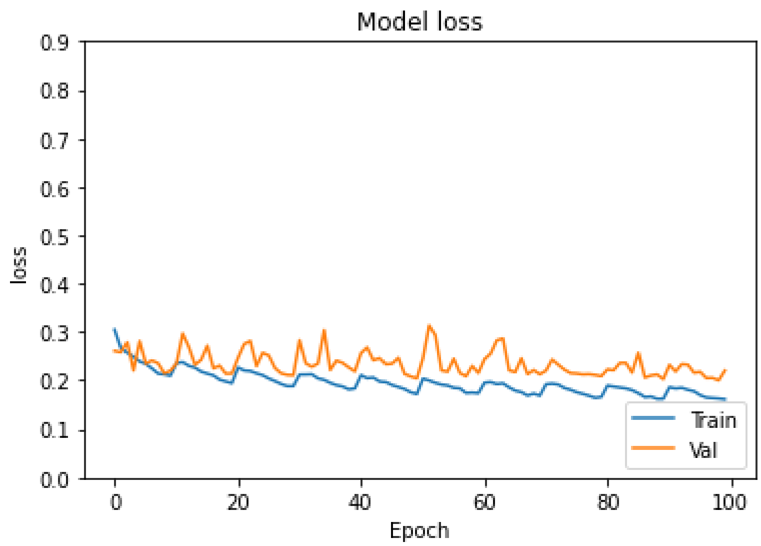
```
Train Set Accuracy:      95.17
```

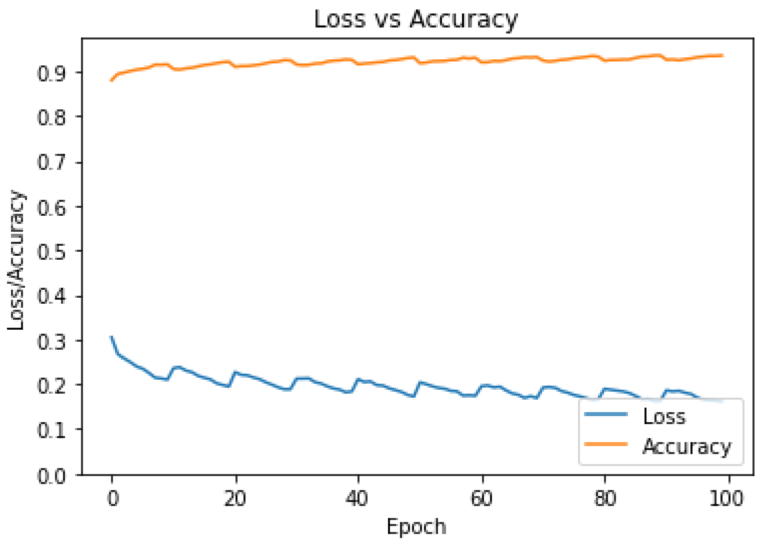
```
Train Set Precision:     0.95
Train Set Recall:        0.95
Train Set F score:       0.95

Val Set Accuracy:        91.33
Val Set Precision:       0.91
Val Set Recall:          0.91
Val Set F score:         0.91

Test Set Accuracy:       91.32
Test Set Precision:      0.91
Test Set Recall:         0.91
Test Set F score:        0.91
```

In [ ]:

```
accuracy_loss_plot(history_final)
```







Discussion of Accuracy Improvement techniques

Accuracy plots

This model achieves an accuracy of up to 96% on the training data. The validation and test sets achieve an accuracy of up to 92.67% and 92.45%. The accuracy values for the training and validation sets are close together.

Loss plots

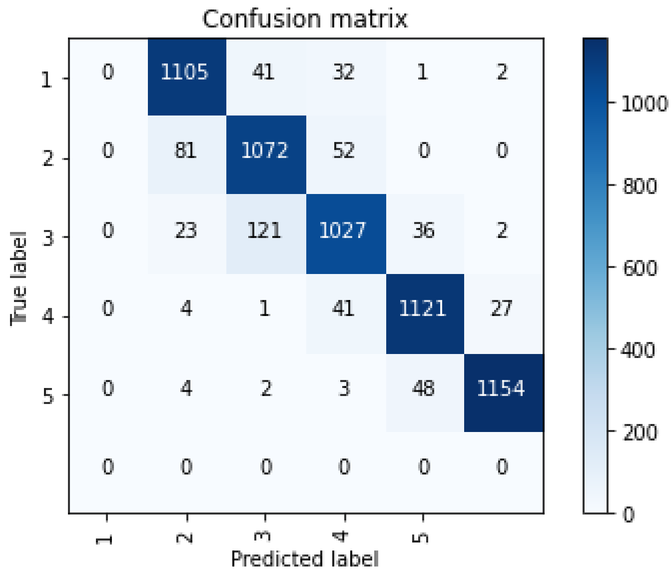
The loss plots for the training and validation sets are close together. This shows that there is no more overfitting.

In [ ]:
```
y_test[:20]
```

Out[ ]:
```
array([4., 0., 2., 4., 0., 3., 2., 2., 2., 0., 4., 2., 3., 2., 1., 1., 4.,
       4., 4., 4.])
```

In [ ]:
```
# Predict the values from the validation dataset
y_pred = model.predict(X_test)
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(y_pred,axis = 1)
#replace classes with original values
y_pred_classes = replace_values(y_pred_classes, [0, 1, 2, 3, 4], [1, 2, 3, 4, 5]
```

In [ ]:
```
#y_true = np.argmax(y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(y_test, y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx,
            classes = [1,2,3,4,5])
```



Hyperparameter Choices

Number of Epochs

Model	No of Epochs
Brief Model	100
Deeper Model	80
Deeper Model with Data Augmentation	100
Deeper Model with Accuracy Improvement	100
ResNet Model	100
VGGNet Model	100

80 epochs were chosen for the vanilla Deeper Model because beyond 80 epochs, it overfit rapidly.

Optimizers

Model	Optimizer
Brief Model	Adam
Deeper Model	Adam
Deeper Model with Data Augmentation	Adam
Deeper Model with Accuracy Improvement	Adamax
ResNet Model	Adam
VGGNet Model	Adam

Batch size

Model	Batch Size
Brief Model	128

Model	Batch Size
Deeper Model	128
Deeper Model with Data Augmentation	128
Deeper Model with Accuracy Improvement	128
ResNet Model	128
VGGNet Model	128

Loss function

The Loss function is the function used to evaluate a candidate solution (i.e. a set of weights). Our model uses Sparse Categorical Cross Entropy.

Transfer Learning

ResNet

ResNet is a network architecture that posseses residual blocks with skip connections, that enable the model to be extremely deep. These skip connections enabled the network to be up to 152 layers with no vanishing or exploding gradient problems during training.

In [ ]:
```
#Resnet
base_model = tf.keras.applications.ResNet152(weights = 'imagenet', include_top =
for layer in base_model.layers:
    layer.trainable = False
```

In [ ]:
```
x = layers.Flatten()(base_model.output)
x = layers.Dense(1000, activation='relu')(x)
predictions = layers.Dense(5, activation = 'softmax')(x)
```

In [ ]:
```
resnet_model = Model(inputs = base_model.input, outputs = predictions)
resnet_model.compile(optimizer='adam', loss=losses.sparse_categorical_crossentro
```

In [ ]:
```
#plot_model(resnet_model, show_shapes=True, rankdir="TD")
```

In [ ]:
```
from sklearn.model_selection import train_test_split
# Splitting the data into train, test, and validation sets
X_train_tl, X_test_tl, y_train, y_test = train_test_split(data['features'], targ
X_val_tl, X_test_tl, y_val, y_test = train_test_split(X_test_tl, y_test, test_si
```

In [ ]:
```
#pad the images to achieve 32 x 32
X_train_tl = tf.pad(X_train_tl, [[0, 0], [2,2], [2,2]])
X_val_tl = tf.pad(X_val_tl, [[0, 0], [2,2], [2,2]])
X_test_tl = tf.pad(X_test_tl, [[0, 0], [2,2], [2,2]])
#expand and repeat to create 3 channels
X_train_tl = tf.expand_dims(X_train_tl, axis=3, name=None)
```

```
X_val_tl = tf.expand_dims(X_val_tl, axis=3, name=None)
X_test_tl = tf.expand_dims(X_test_tl, axis=3, name=None)
```

In [ ]:

```
print(X_train_tl.shape, '\n')
print(y_train.shape, '\n')
print(X_val_tl.shape, '\n')
print(X_test_tl.shape, '\n')
```

```
(48000, 32, 32, 1)

(48000,)

(6000, 32, 32, 1)

(6000, 32, 32, 1)
```

In [ ]:

```
X_train_tl = tf.repeat(X_train_tl, 3, axis=3)
X_val_tl = tf.repeat(X_val_tl, 3, axis=3)
X_test_tl = tf.repeat(X_test_tl, 3, axis=3)
```

In [ ]:

```
print(X_train_tl.shape, '\n')
print(y_train.shape, '\n')
print(X_val_tl.shape, '\n')
print(X_test_tl.shape, '\n')
```

```
(48000, 32, 32, 3)

(48000,)

(6000, 32, 32, 3)

(6000, 32, 32, 3)
```

In [ ]:

```
%%time
with tf.device('/device:GPU:0'):
  history_resnet = resnet_model.fit(X_train_tl, y_train, batch_size=128, epochs=
```

```
Epoch 1/100
375/375 [==============================] - 26s 52ms/step - loss: 1.1156 - accura
cy: 0.5377 - val_loss: 0.8029 - val_accuracy: 0.6733
Epoch 2/100
375/375 [==============================] - 17s 45ms/step - loss: 0.8012 - accura
cy: 0.6768 - val_loss: 0.7671 - val_accuracy: 0.6795
Epoch 3/100
375/375 [==============================] - 17s 45ms/step - loss: 0.7469 - accura
cy: 0.6968 - val_loss: 0.7422 - val_accuracy: 0.6898
Epoch 4/100
375/375 [==============================] - 17s 45ms/step - loss: 0.7193 - accura
cy: 0.7036 - val_loss: 0.7064 - val_accuracy: 0.7040
Epoch 5/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6991 - accura
cy: 0.7108 - val_loss: 0.6842 - val_accuracy: 0.7148
Epoch 6/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6807 - accura
cy: 0.7165 - val_loss: 0.7160 - val_accuracy: 0.6980
Epoch 7/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6750 - accura
```

```
cy: 0.7195 - val_loss: 0.6759 - val_accuracy: 0.7072
Epoch 8/100
375/375 [==============================] - 17s 46ms/step - loss: 0.6567 - accura
cy: 0.7249 - val_loss: 0.6596 - val_accuracy: 0.7195
Epoch 9/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6453 - accura
cy: 0.7297 - val_loss: 0.6441 - val_accuracy: 0.7237
Epoch 10/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6336 - accura
cy: 0.7332 - val_loss: 0.6387 - val_accuracy: 0.7267
Epoch 11/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6321 - accura
cy: 0.7358 - val_loss: 0.6289 - val_accuracy: 0.7383
Epoch 12/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6215 - accura
cy: 0.7428 - val_loss: 0.6324 - val_accuracy: 0.7333
Epoch 13/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6183 - accura
cy: 0.7421 - val_loss: 0.6366 - val_accuracy: 0.7285
Epoch 14/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6104 - accura
cy: 0.7415 - val_loss: 0.6297 - val_accuracy: 0.7370
Epoch 15/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5981 - accura
cy: 0.7510 - val_loss: 0.6062 - val_accuracy: 0.7482
Epoch 16/100
375/375 [==============================] - 17s 45ms/step - loss: 0.6041 - accura
cy: 0.7489 - val_loss: 0.5882 - val_accuracy: 0.7522
Epoch 17/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5875 - accura
cy: 0.7557 - val_loss: 0.5968 - val_accuracy: 0.7452
Epoch 18/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5765 - accura
cy: 0.7607 - val_loss: 0.6062 - val_accuracy: 0.7423
Epoch 19/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5708 - accura
cy: 0.7623 - val_loss: 0.5870 - val_accuracy: 0.7557
Epoch 20/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5687 - accura
cy: 0.7634 - val_loss: 0.5749 - val_accuracy: 0.7655
Epoch 21/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5718 - accura
cy: 0.7587 - val_loss: 0.5919 - val_accuracy: 0.7473
Epoch 22/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5606 - accura
cy: 0.7650 - val_loss: 0.5608 - val_accuracy: 0.7680
Epoch 23/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5577 - accura
cy: 0.7653 - val_loss: 0.5565 - val_accuracy: 0.7682
Epoch 24/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5491 - accura
cy: 0.7692 - val_loss: 0.5512 - val_accuracy: 0.7685
Epoch 25/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5563 - accura
cy: 0.7664 - val_loss: 0.5590 - val_accuracy: 0.7687
Epoch 26/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5509 - accura
cy: 0.7701 - val_loss: 0.5733 - val_accuracy: 0.7535
Epoch 27/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5452 - accura
cy: 0.7742 - val_loss: 0.5670 - val_accuracy: 0.7607
Epoch 28/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5361 - accura
cy: 0.7768 - val_loss: 0.5497 - val_accuracy: 0.7715
Epoch 29/100
```

```
375/375 [==============================] - 17s 46ms/step - loss: 0.5398 - accura
cy: 0.7772 - val_loss: 0.5527 - val_accuracy: 0.7712
Epoch 30/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5360 - accura
cy: 0.7776 - val_loss: 0.5393 - val_accuracy: 0.7740
Epoch 31/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5316 - accura
cy: 0.7797 - val_loss: 0.5526 - val_accuracy: 0.7705
Epoch 32/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5298 - accura
cy: 0.7821 - val_loss: 0.5407 - val_accuracy: 0.7707
Epoch 33/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5326 - accura
cy: 0.7785 - val_loss: 0.5382 - val_accuracy: 0.7685
Epoch 34/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5166 - accura
cy: 0.7858 - val_loss: 0.5290 - val_accuracy: 0.7758
Epoch 35/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5215 - accura
cy: 0.7852 - val_loss: 0.5313 - val_accuracy: 0.7810
Epoch 36/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5229 - accura
cy: 0.7829 - val_loss: 0.5224 - val_accuracy: 0.7895
Epoch 37/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5210 - accura
cy: 0.7844 - val_loss: 0.5477 - val_accuracy: 0.7693
Epoch 38/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5091 - accura
cy: 0.7889 - val_loss: 0.5530 - val_accuracy: 0.7712
Epoch 39/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5185 - accura
cy: 0.7835 - val_loss: 0.5461 - val_accuracy: 0.7738
Epoch 40/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5123 - accura
cy: 0.7860 - val_loss: 0.5277 - val_accuracy: 0.7845
Epoch 41/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5069 - accura
cy: 0.7908 - val_loss: 0.5180 - val_accuracy: 0.7828
Epoch 42/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5038 - accura
cy: 0.7900 - val_loss: 0.5390 - val_accuracy: 0.7750
Epoch 43/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5044 - accura
cy: 0.7894 - val_loss: 0.5221 - val_accuracy: 0.7858
Epoch 44/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5117 - accura
cy: 0.7899 - val_loss: 0.5270 - val_accuracy: 0.7783
Epoch 45/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5037 - accura
cy: 0.7915 - val_loss: 0.5093 - val_accuracy: 0.7835
Epoch 46/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4980 - accura
cy: 0.7950 - val_loss: 0.5120 - val_accuracy: 0.7833
Epoch 47/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5004 - accura
cy: 0.7983 - val_loss: 0.5439 - val_accuracy: 0.7708
Epoch 48/100
375/375 [==============================] - 17s 46ms/step - loss: 0.5008 - accura
cy: 0.7906 - val_loss: 0.5179 - val_accuracy: 0.7862
Epoch 49/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4913 - accura
cy: 0.7968 - val_loss: 0.5259 - val_accuracy: 0.7772
Epoch 50/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4958 - accura
cy: 0.7953 - val_loss: 0.5070 - val_accuracy: 0.7852
```

```
Epoch 51/100
375/375 [==============================] - 17s 45ms/step - loss: 0.5018 - accura
cy: 0.7930 - val_loss: 0.5196 - val_accuracy: 0.7842
Epoch 52/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4958 - accura
cy: 0.7934 - val_loss: 0.5106 - val_accuracy: 0.7855
Epoch 53/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4859 - accura
cy: 0.7989 - val_loss: 0.5374 - val_accuracy: 0.7735
Epoch 54/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4910 - accura
cy: 0.7944 - val_loss: 0.5170 - val_accuracy: 0.7798
Epoch 55/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4917 - accura
cy: 0.7958 - val_loss: 0.5051 - val_accuracy: 0.7907
Epoch 56/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4811 - accura
cy: 0.8004 - val_loss: 0.5042 - val_accuracy: 0.7843
Epoch 57/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4849 - accura
cy: 0.8010 - val_loss: 0.5140 - val_accuracy: 0.7850
Epoch 58/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4905 - accura
cy: 0.7974 - val_loss: 0.5087 - val_accuracy: 0.7873
Epoch 59/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4868 - accura
cy: 0.7979 - val_loss: 0.4952 - val_accuracy: 0.7913
Epoch 60/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4740 - accura
cy: 0.8051 - val_loss: 0.5201 - val_accuracy: 0.7840
Epoch 61/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4894 - accura
cy: 0.7963 - val_loss: 0.4871 - val_accuracy: 0.7932
Epoch 62/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4734 - accura
cy: 0.8048 - val_loss: 0.4839 - val_accuracy: 0.7967
Epoch 63/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4735 - accura
cy: 0.8036 - val_loss: 0.5083 - val_accuracy: 0.7875
Epoch 64/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4779 - accura
cy: 0.8039 - val_loss: 0.4958 - val_accuracy: 0.7942
Epoch 65/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4697 - accura
cy: 0.8065 - val_loss: 0.4853 - val_accuracy: 0.8012
Epoch 66/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4771 - accura
cy: 0.8015 - val_loss: 0.4919 - val_accuracy: 0.7967
Epoch 67/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4731 - accura
cy: 0.8021 - val_loss: 0.4874 - val_accuracy: 0.7965
Epoch 68/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4743 - accura
cy: 0.8025 - val_loss: 0.4962 - val_accuracy: 0.7972
Epoch 69/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4754 - accura
cy: 0.8045 - val_loss: 0.4996 - val_accuracy: 0.7875
Epoch 70/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4716 - accura
cy: 0.8031 - val_loss: 0.4886 - val_accuracy: 0.7933
Epoch 71/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4703 - accura
cy: 0.8059 - val_loss: 0.5052 - val_accuracy: 0.7908
Epoch 72/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4707 - accura
```

```
cy: 0.8050 - val_loss: 0.4892 - val_accuracy: 0.7970
Epoch 73/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4756 - accura
cy: 0.8019 - val_loss: 0.4839 - val_accuracy: 0.7972
Epoch 74/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4681 - accura
cy: 0.8069 - val_loss: 0.4826 - val_accuracy: 0.8003
Epoch 75/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4751 - accura
cy: 0.8032 - val_loss: 0.4803 - val_accuracy: 0.8010
Epoch 76/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4629 - accura
cy: 0.8069 - val_loss: 0.5460 - val_accuracy: 0.7762
Epoch 77/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4654 - accura
cy: 0.8079 - val_loss: 0.4995 - val_accuracy: 0.7932
Epoch 78/100
375/375 [==============================] - 17s 45ms/step - loss: 0.4648 - accura
cy: 0.8068 - val_loss: 0.4854 - val_accuracy: 0.7995
Epoch 79/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4710 - accura
cy: 0.8006 - val_loss: 0.4876 - val_accuracy: 0.8000
Epoch 80/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4654 - accura
cy: 0.8070 - val_loss: 0.4875 - val_accuracy: 0.7987
Epoch 81/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4593 - accura
cy: 0.8092 - val_loss: 0.4835 - val_accuracy: 0.8015
Epoch 82/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4597 - accura
cy: 0.8094 - val_loss: 0.4839 - val_accuracy: 0.7988
Epoch 83/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4661 - accura
cy: 0.8060 - val_loss: 0.4746 - val_accuracy: 0.8015
Epoch 84/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4586 - accura
cy: 0.8077 - val_loss: 0.4843 - val_accuracy: 0.7997
Epoch 85/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4553 - accura
cy: 0.8107 - val_loss: 0.4909 - val_accuracy: 0.7968
Epoch 86/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4557 - accura
cy: 0.8114 - val_loss: 0.4719 - val_accuracy: 0.8010
Epoch 87/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4516 - accura
cy: 0.8131 - val_loss: 0.4864 - val_accuracy: 0.7982
Epoch 88/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4537 - accura
cy: 0.8127 - val_loss: 0.4848 - val_accuracy: 0.8007
Epoch 89/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4470 - accura
cy: 0.8151 - val_loss: 0.4939 - val_accuracy: 0.7958
Epoch 90/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4572 - accura
cy: 0.8096 - val_loss: 0.4755 - val_accuracy: 0.8008
Epoch 91/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4462 - accura
cy: 0.8158 - val_loss: 0.4781 - val_accuracy: 0.7992
Epoch 92/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4527 - accura
cy: 0.8139 - val_loss: 0.4754 - val_accuracy: 0.8068
Epoch 93/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4462 - accura
cy: 0.8146 - val_loss: 0.4672 - val_accuracy: 0.8085
Epoch 94/100
```

```
375/375 [==============================] - 17s 46ms/step - loss: 0.4525 - accura
cy: 0.8118 - val_loss: 0.4690 - val_accuracy: 0.8047
Epoch 95/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4579 - accura
cy: 0.8130 - val_loss: 0.4704 - val_accuracy: 0.8023
Epoch 96/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4472 - accura
cy: 0.8137 - val_loss: 0.4784 - val_accuracy: 0.8078
Epoch 97/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4479 - accura
cy: 0.8154 - val_loss: 0.4698 - val_accuracy: 0.8082
Epoch 98/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4425 - accura
cy: 0.8186 - val_loss: 0.4776 - val_accuracy: 0.8000
Epoch 99/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4472 - accura
cy: 0.8156 - val_loss: 0.4717 - val_accuracy: 0.8063
Epoch 100/100
375/375 [==============================] - 17s 46ms/step - loss: 0.4482 - accura
cy: 0.8134 - val_loss: 0.4710 - val_accuracy: 0.8028
CPU times: user 22min 22s, sys: 5min 44s, total: 28min 6s
Wall time: 28min 37s
```

In [95]:

```
%%time
resnet_model.evaluate(X_test_tl, y_test)
```

```
188/188 [==============================] - 5s 27ms/step - loss: 0.4626 - accurac
y: 0.8113
CPU times: user 4.34 s, sys: 134 ms, total: 4.48 s
Wall time: 5.16 s
```

Out[95]:

```
[0.4625893831253052, 0.8113333582878113]
```

In [96]:

```
history_list.append(history_resnet)
```

In [97]:

```
print_accuracy(resnet_model, y_train, y_val, y_test, mode=1)
```

```
Train Set Accuracy:      82.23
Train Set Precision:     0.82
Train Set Recall:        0.82
Train Set F score:       0.82

Val Set Accuracy:        80.28
Val Set Precision:       0.8
Val Set Recall:          0.8
Val Set F score:         0.8

Test Set Accuracy:       81.13
Test Set Precision:      0.81
Test Set Recall:         0.81
Test Set F score:        0.81
```

Transfer Learning with ResNet Performance

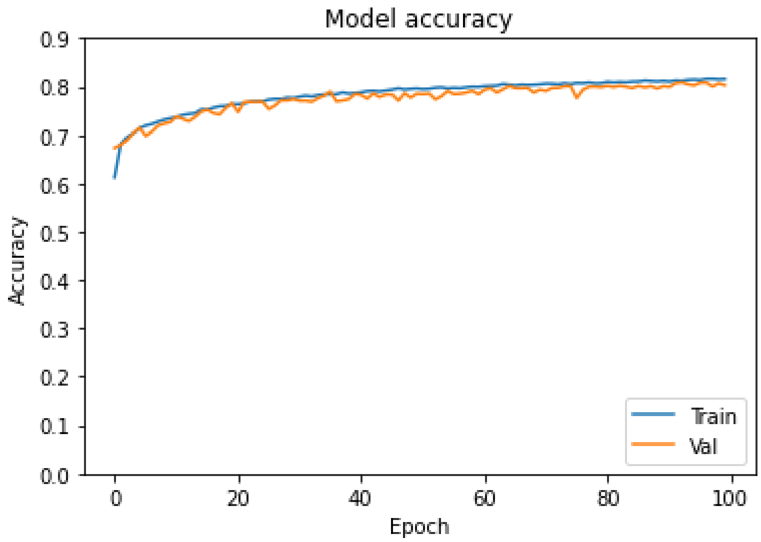
Training Accuracy - 82%

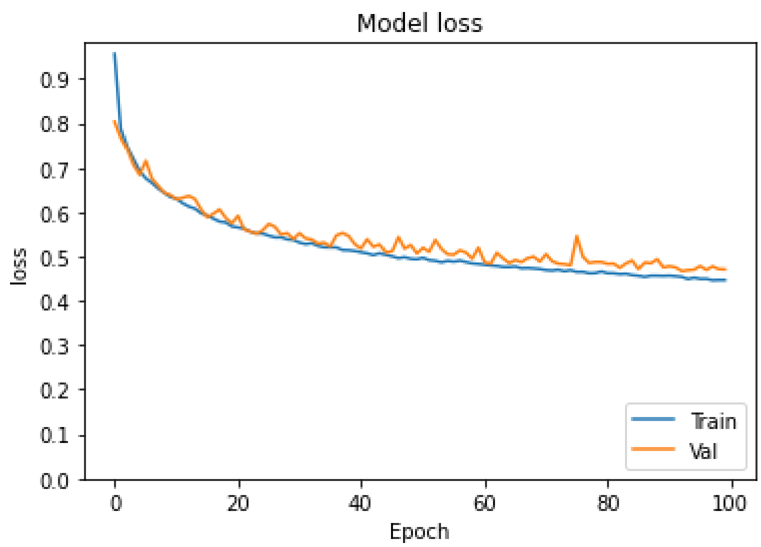
Validation Accuracy - 80%

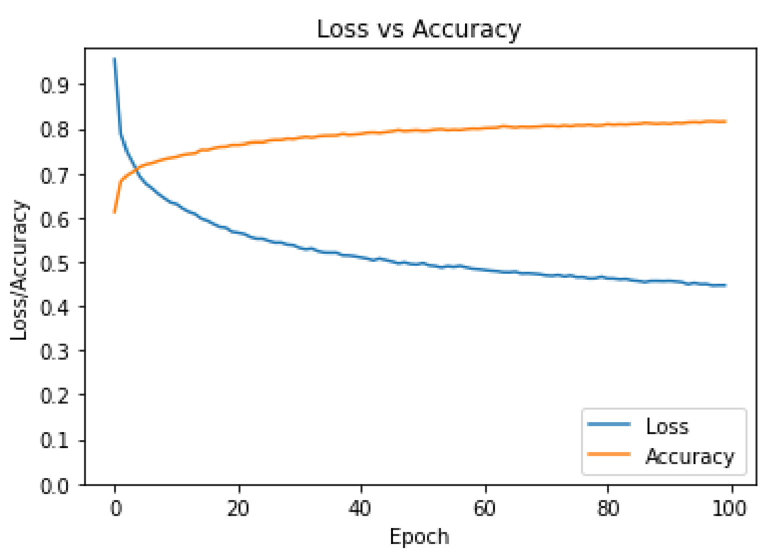
Test Accuracy - 81%

This produces the worst accuracy out of all the models

In [98]:
```
accuracy_loss_plot(history_resnet)
```







VGGNet

The input to VGG based convNet is a 224*224 RGB image. The training images are passed through a stack of convolution layers. There are total of 13 convolutional layers and 3 fully connected layers in VGG16 architecture.

VGG-16 was one of the best performing architecture in ILSVRC challenge 2014.It was the runner up in classification task with top-5 classification error of 7.32% (only behind GoogLeNet with classification error 6.66%). It was also the winner of localization task with 25.32% localization error.

In [ ]:
```
base_model = tf.keras.applications.VGG16(weights = 'imagenet', include_top = Fal
for layer in base_model.layers:
  layer.trainable = False
base_model.summary()
```

In [ ]:
```
x = layers.Flatten()(base_model.output)
x = layers.Dense(4096, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(4096, activation='relu')(x)
x = layers.Dropout(0.5)(x)
predictions = layers.Dense(5, activation = 'softmax')(x)
head_model = Model(inputs = base_model.input, outputs = predictions)
head_model.compile(optimizer='adam', loss=losses.sparse_categorical_crossentropy
```

In [ ]:
```
plot_model(head_model, show_shapes=True, rankdir="TD")
```

In [ ]:
```
head_model.summary()
```

In [ ]:
```
%%time
with tf.device('/device:GPU:0'):
  history = head_model.fit(X_train_tl, y_train, batch_size=128, epochs=100, vali
```

```
Epoch 1/100
375/375 [==============================] - 8s 18ms/step - loss: 0.9240 - accurac
y: 0.6591 - val_loss: 0.4892 - val_accuracy: 0.7918
Epoch 2/100
375/375 [==============================] - 6s 17ms/step - loss: 0.5371 - accurac
y: 0.7795 - val_loss: 0.4494 - val_accuracy: 0.8158
Epoch 3/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4933 - accurac
y: 0.7987 - val_loss: 0.4518 - val_accuracy: 0.8128
Epoch 4/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4719 - accurac
y: 0.8074 - val_loss: 0.3977 - val_accuracy: 0.8385
Epoch 5/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4497 - accurac
y: 0.8200 - val_loss: 0.4231 - val_accuracy: 0.8255
Epoch 6/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4471 - accurac
y: 0.8207 - val_loss: 0.3999 - val_accuracy: 0.8283
Epoch 7/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4310 - accurac
y: 0.8274 - val_loss: 0.3840 - val_accuracy: 0.8458
Epoch 8/100
```

```
375/375 [==============================] - 6s 17ms/step - loss: 0.4200 - accurac
y: 0.8294 - val_loss: 0.3831 - val_accuracy: 0.8447
Epoch 9/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4087 - accurac
y: 0.8349 - val_loss: 0.3804 - val_accuracy: 0.8482
Epoch 10/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4114 - accurac
y: 0.8336 - val_loss: 0.3780 - val_accuracy: 0.8468
Epoch 11/100
375/375 [==============================] - 6s 17ms/step - loss: 0.4065 - accurac
y: 0.8356 - val_loss: 0.3932 - val_accuracy: 0.8402
Epoch 12/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3943 - accurac
y: 0.8405 - val_loss: 0.3691 - val_accuracy: 0.8465
Epoch 13/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3902 - accurac
y: 0.8408 - val_loss: 0.3768 - val_accuracy: 0.8435
Epoch 14/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3864 - accurac
y: 0.8423 - val_loss: 0.3669 - val_accuracy: 0.8577
Epoch 15/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3817 - accurac
y: 0.8487 - val_loss: 0.3663 - val_accuracy: 0.8572
Epoch 16/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3874 - accurac
y: 0.8451 - val_loss: 0.3599 - val_accuracy: 0.8598
Epoch 17/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3807 - accurac
y: 0.8462 - val_loss: 0.3576 - val_accuracy: 0.8607
Epoch 18/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3670 - accurac
y: 0.8550 - val_loss: 0.3742 - val_accuracy: 0.8472
Epoch 19/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3726 - accurac
y: 0.8484 - val_loss: 0.3647 - val_accuracy: 0.8505
Epoch 20/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3631 - accurac
y: 0.8544 - val_loss: 0.3645 - val_accuracy: 0.8525
Epoch 21/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3629 - accurac
y: 0.8552 - val_loss: 0.3646 - val_accuracy: 0.8560
Epoch 22/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3513 - accurac
y: 0.8571 - val_loss: 0.3562 - val_accuracy: 0.8570
Epoch 23/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3519 - accurac
y: 0.8586 - val_loss: 0.3666 - val_accuracy: 0.8535
Epoch 24/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3454 - accurac
y: 0.8602 - val_loss: 0.3512 - val_accuracy: 0.8580
Epoch 25/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3412 - accurac
y: 0.8625 - val_loss: 0.3632 - val_accuracy: 0.8547
Epoch 26/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3513 - accurac
y: 0.8572 - val_loss: 0.3557 - val_accuracy: 0.8625
Epoch 27/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3437 - accurac
y: 0.8611 - val_loss: 0.3551 - val_accuracy: 0.8568
Epoch 28/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3331 - accurac
y: 0.8644 - val_loss: 0.3515 - val_accuracy: 0.8607
Epoch 29/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3385 - accurac
y: 0.8637 - val_loss: 0.3439 - val_accuracy: 0.8650
```

```
Epoch 30/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3435 - accurac
y: 0.8632 - val_loss: 0.3458 - val_accuracy: 0.8630
Epoch 31/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3337 - accurac
y: 0.8636 - val_loss: 0.3550 - val_accuracy: 0.8572
Epoch 32/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3290 - accurac
y: 0.8684 - val_loss: 0.3520 - val_accuracy: 0.8597
Epoch 33/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3222 - accurac
y: 0.8703 - val_loss: 0.3546 - val_accuracy: 0.8640
Epoch 34/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3191 - accurac
y: 0.8719 - val_loss: 0.3469 - val_accuracy: 0.8602
Epoch 35/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3199 - accurac
y: 0.8735 - val_loss: 0.3437 - val_accuracy: 0.8668
Epoch 36/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3212 - accurac
y: 0.8717 - val_loss: 0.3571 - val_accuracy: 0.8640
Epoch 37/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3200 - accurac
y: 0.8730 - val_loss: 0.3461 - val_accuracy: 0.8670
Epoch 38/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3202 - accurac
y: 0.8737 - val_loss: 0.3504 - val_accuracy: 0.8638
Epoch 39/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3161 - accurac
y: 0.8728 - val_loss: 0.3515 - val_accuracy: 0.8627
Epoch 40/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3076 - accurac
y: 0.8771 - val_loss: 0.3471 - val_accuracy: 0.8667
Epoch 41/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3090 - accurac
y: 0.8756 - val_loss: 0.3502 - val_accuracy: 0.8613
Epoch 42/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3071 - accurac
y: 0.8766 - val_loss: 0.3433 - val_accuracy: 0.8693
Epoch 43/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3124 - accurac
y: 0.8722 - val_loss: 0.3520 - val_accuracy: 0.8622
Epoch 44/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3038 - accurac
y: 0.8782 - val_loss: 0.3527 - val_accuracy: 0.8642
Epoch 45/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2990 - accurac
y: 0.8789 - val_loss: 0.3584 - val_accuracy: 0.8572
Epoch 46/100
375/375 [==============================] - 6s 17ms/step - loss: 0.3008 - accurac
y: 0.8797 - val_loss: 0.3461 - val_accuracy: 0.8710
Epoch 47/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2987 - accurac
y: 0.8818 - val_loss: 0.3539 - val_accuracy: 0.8607
Epoch 48/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2970 - accurac
y: 0.8793 - val_loss: 0.3517 - val_accuracy: 0.8638
Epoch 49/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2948 - accurac
y: 0.8821 - val_loss: 0.3538 - val_accuracy: 0.8623
Epoch 50/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2915 - accurac
y: 0.8852 - val_loss: 0.3651 - val_accuracy: 0.8583
Epoch 51/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2883 - accurac
```

```
y: 0.8858 - val_loss: 0.3514 - val_accuracy: 0.8702
Epoch 52/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2960 - accurac
y: 0.8818 - val_loss: 0.3537 - val_accuracy: 0.8630
Epoch 53/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2911 - accurac
y: 0.8806 - val_loss: 0.3550 - val_accuracy: 0.8590
Epoch 54/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2850 - accurac
y: 0.8875 - val_loss: 0.3585 - val_accuracy: 0.8635
Epoch 55/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2856 - accurac
y: 0.8856 - val_loss: 0.3591 - val_accuracy: 0.8627
Epoch 56/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2831 - accurac
y: 0.8870 - val_loss: 0.3570 - val_accuracy: 0.8625
Epoch 57/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2827 - accurac
y: 0.8868 - val_loss: 0.3517 - val_accuracy: 0.8685
Epoch 58/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2845 - accurac
y: 0.8848 - val_loss: 0.3607 - val_accuracy: 0.8653
Epoch 59/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2787 - accurac
y: 0.8887 - val_loss: 0.3471 - val_accuracy: 0.8663
Epoch 60/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2802 - accurac
y: 0.8875 - val_loss: 0.3586 - val_accuracy: 0.8708
Epoch 61/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2829 - accurac
y: 0.8882 - val_loss: 0.3476 - val_accuracy: 0.8690
Epoch 62/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2775 - accurac
y: 0.8881 - val_loss: 0.3547 - val_accuracy: 0.8692
Epoch 63/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2758 - accurac
y: 0.8892 - val_loss: 0.3543 - val_accuracy: 0.8675
Epoch 64/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2749 - accurac
y: 0.8894 - val_loss: 0.3673 - val_accuracy: 0.8622
Epoch 65/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2759 - accurac
y: 0.8894 - val_loss: 0.3606 - val_accuracy: 0.8658
Epoch 66/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2678 - accurac
y: 0.8926 - val_loss: 0.3556 - val_accuracy: 0.8717
Epoch 67/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2739 - accurac
y: 0.8902 - val_loss: 0.3583 - val_accuracy: 0.8647
Epoch 68/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2723 - accurac
y: 0.8937 - val_loss: 0.3539 - val_accuracy: 0.8713
Epoch 69/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2717 - accurac
y: 0.8930 - val_loss: 0.3550 - val_accuracy: 0.8667
Epoch 70/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2716 - accurac
y: 0.8929 - val_loss: 0.3599 - val_accuracy: 0.8680
Epoch 71/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2714 - accurac
y: 0.8911 - val_loss: 0.3528 - val_accuracy: 0.8685
Epoch 72/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2744 - accurac
y: 0.8924 - val_loss: 0.3579 - val_accuracy: 0.8700
Epoch 73/100
```

```
375/375 [==============================] - 6s 17ms/step - loss: 0.2670 - accurac
y: 0.8944 - val_loss: 0.3647 - val_accuracy: 0.8658
Epoch 74/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2731 - accurac
y: 0.8921 - val_loss: 0.3594 - val_accuracy: 0.8675
Epoch 75/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2669 - accurac
y: 0.8952 - val_loss: 0.3620 - val_accuracy: 0.8668
Epoch 76/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2610 - accurac
y: 0.8960 - val_loss: 0.3560 - val_accuracy: 0.8717
Epoch 77/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2646 - accurac
y: 0.8950 - val_loss: 0.3587 - val_accuracy: 0.8648
Epoch 78/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2582 - accurac
y: 0.8965 - val_loss: 0.3599 - val_accuracy: 0.8663
Epoch 79/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2625 - accurac
y: 0.8943 - val_loss: 0.3693 - val_accuracy: 0.8660
Epoch 80/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2604 - accurac
y: 0.8964 - val_loss: 0.3701 - val_accuracy: 0.8632
Epoch 81/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2581 - accurac
y: 0.8963 - val_loss: 0.3649 - val_accuracy: 0.8633
Epoch 82/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2550 - accurac
y: 0.8975 - val_loss: 0.3685 - val_accuracy: 0.8628
Epoch 83/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2557 - accurac
y: 0.8991 - val_loss: 0.3680 - val_accuracy: 0.8610
Epoch 84/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2637 - accurac
y: 0.8957 - val_loss: 0.3682 - val_accuracy: 0.8667
Epoch 85/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2519 - accurac
y: 0.9008 - val_loss: 0.3552 - val_accuracy: 0.8723
Epoch 86/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2617 - accurac
y: 0.8954 - val_loss: 0.3647 - val_accuracy: 0.8697
Epoch 87/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2494 - accurac
y: 0.9011 - val_loss: 0.3608 - val_accuracy: 0.8705
Epoch 88/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2608 - accurac
y: 0.8977 - val_loss: 0.3603 - val_accuracy: 0.8712
Epoch 89/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2510 - accurac
y: 0.9015 - val_loss: 0.3681 - val_accuracy: 0.8632
Epoch 90/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2540 - accurac
y: 0.8976 - val_loss: 0.3651 - val_accuracy: 0.8668
Epoch 91/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2463 - accurac
y: 0.9010 - val_loss: 0.3764 - val_accuracy: 0.8632
Epoch 92/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2503 - accurac
y: 0.9008 - val_loss: 0.3695 - val_accuracy: 0.8665
Epoch 93/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2512 - accurac
y: 0.9018 - val_loss: 0.3708 - val_accuracy: 0.8682
Epoch 94/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2417 - accurac
y: 0.9043 - val_loss: 0.3588 - val_accuracy: 0.8690
```

```
Epoch 95/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2548 - accurac
y: 0.8966 - val_loss: 0.3753 - val_accuracy: 0.8622
Epoch 96/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2356 - accurac
y: 0.9072 - val_loss: 0.3762 - val_accuracy: 0.8665
Epoch 97/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2367 - accurac
y: 0.9047 - val_loss: 0.3691 - val_accuracy: 0.8708
Epoch 98/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2436 - accurac
y: 0.9043 - val_loss: 0.3897 - val_accuracy: 0.8590
Epoch 99/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2460 - accurac
y: 0.9030 - val_loss: 0.3696 - val_accuracy: 0.8725
Epoch 100/100
375/375 [==============================] - 6s 17ms/step - loss: 0.2464 - accurac
y: 0.9023 - val_loss: 0.3654 - val_accuracy: 0.8678
CPU times: user 7min 24s, sys: 3min 6s, total: 10min 30s
Wall time: 10min 33s
```

In [ ]:

```
%%time
head_model.evaluate(X_test_tl, y_test)
```

```
188/188 [==============================] - 2s 8ms/step - loss: 0.3900 - accurac
y: 0.8542
CPU times: user 1.03 s, sys: 197 ms, total: 1.23 s
Wall time: 1.77 s
```

Out[ ]:

```
[0.39000391960144043, 0.8541666865348816]
```

In [ ]:

```
history_list.append(history)
```

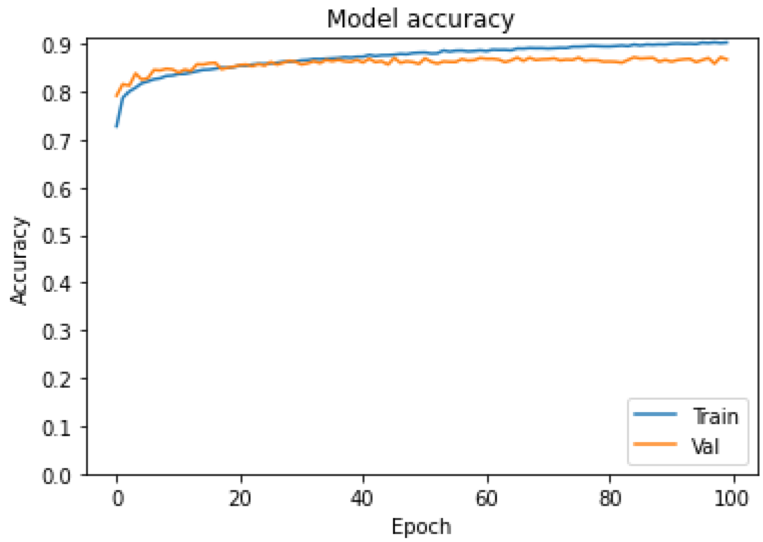
In [ ]:

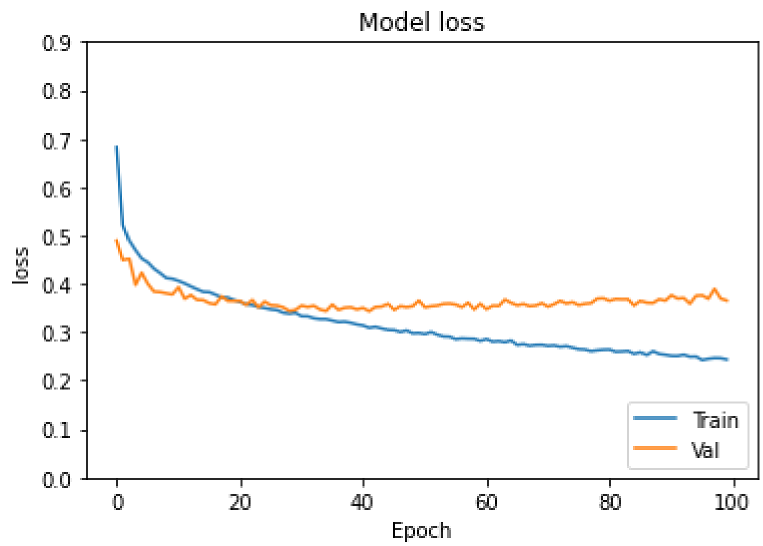
```
print_accuracy(head_model, y_train, y_val, y_test, mode=1)
```

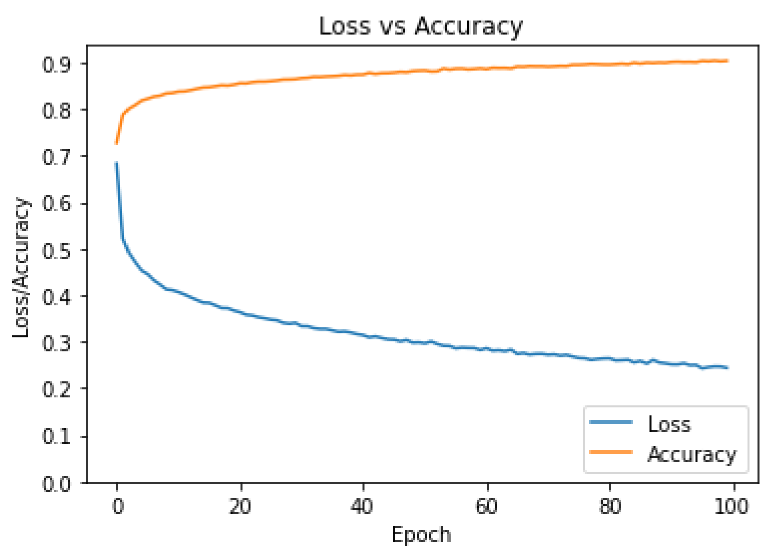
```
Train Set Accuracy:      92.92
Train Set Precision:     0.93
Train Set Recall:        0.93
Train Set F score:       0.93

Val Set Accuracy:        86.78
Val Set Precision:       0.87
Val Set Recall:          0.87
Val Set F score:         0.87

Test Set Accuracy:       85.42
Test Set Precision:      0.86
Test Set Recall:         0.85
Test Set F score:        0.85
```

In [ ]:

```
accuracy_loss_plot(history)
```







Transfer Learning with VGGNet Performance

Training Accuracy - 90%

Validation Accuracy - 86%

Test Accuracy - 85%

There is overfitting in with this model. This is probably because the VGGNet architecture is too deep for this task.

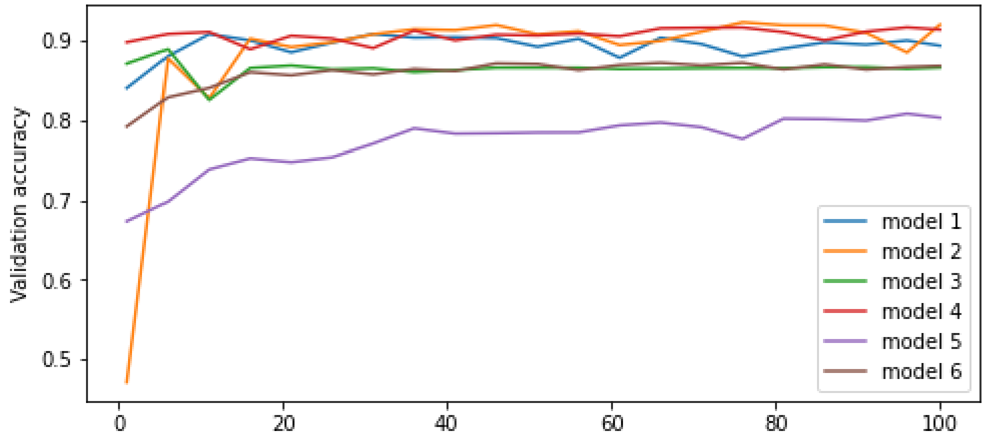
Comparison of Different Models

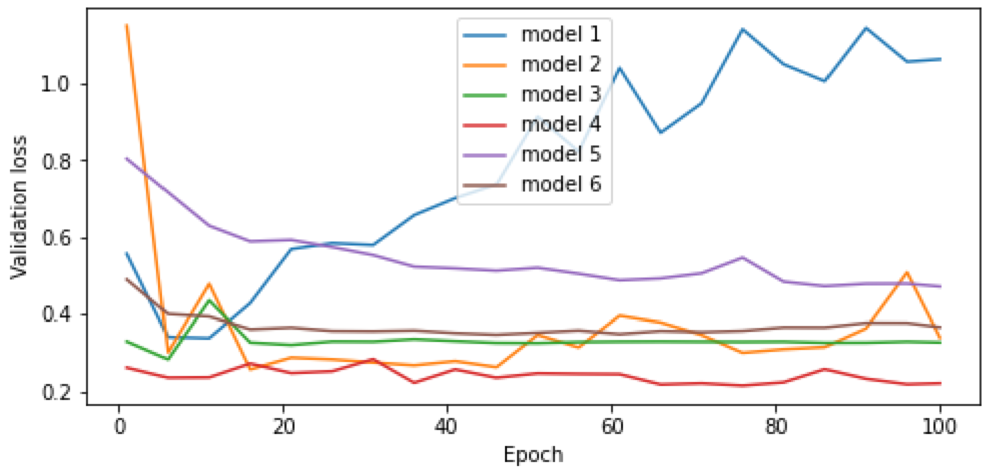
In [ ]:

```
fig, (ax1, ax2) = plt.subplots(2, figsize=(8, 8))
count = 1
epochs = list(range(1,100, 5))
epochs.append(100)
for history in history_list:
    label = 'model ' + str(count)
    val_accuracy = [history.history['val_accuracy'][i] for i in range(len(histor
    val_loss = [history.history['val_loss'][i] for i in range(len(history.histor
    val_accuracy.append(history.history['val_accuracy'][-1])
    val_loss.append(history.history['val_loss'][-1])
    ax1.plot(epochs, val_accuracy, label= label)
    ax2.plot(epochs, val_loss, label=label)
    count += 1

ax1.set_ylabel('Validation accuracy')
ax2.set_ylabel('Validation loss')
ax2.set_xlabel('Epoch')
ax1.legend()
ax2.legend()
```

Out[ ]:

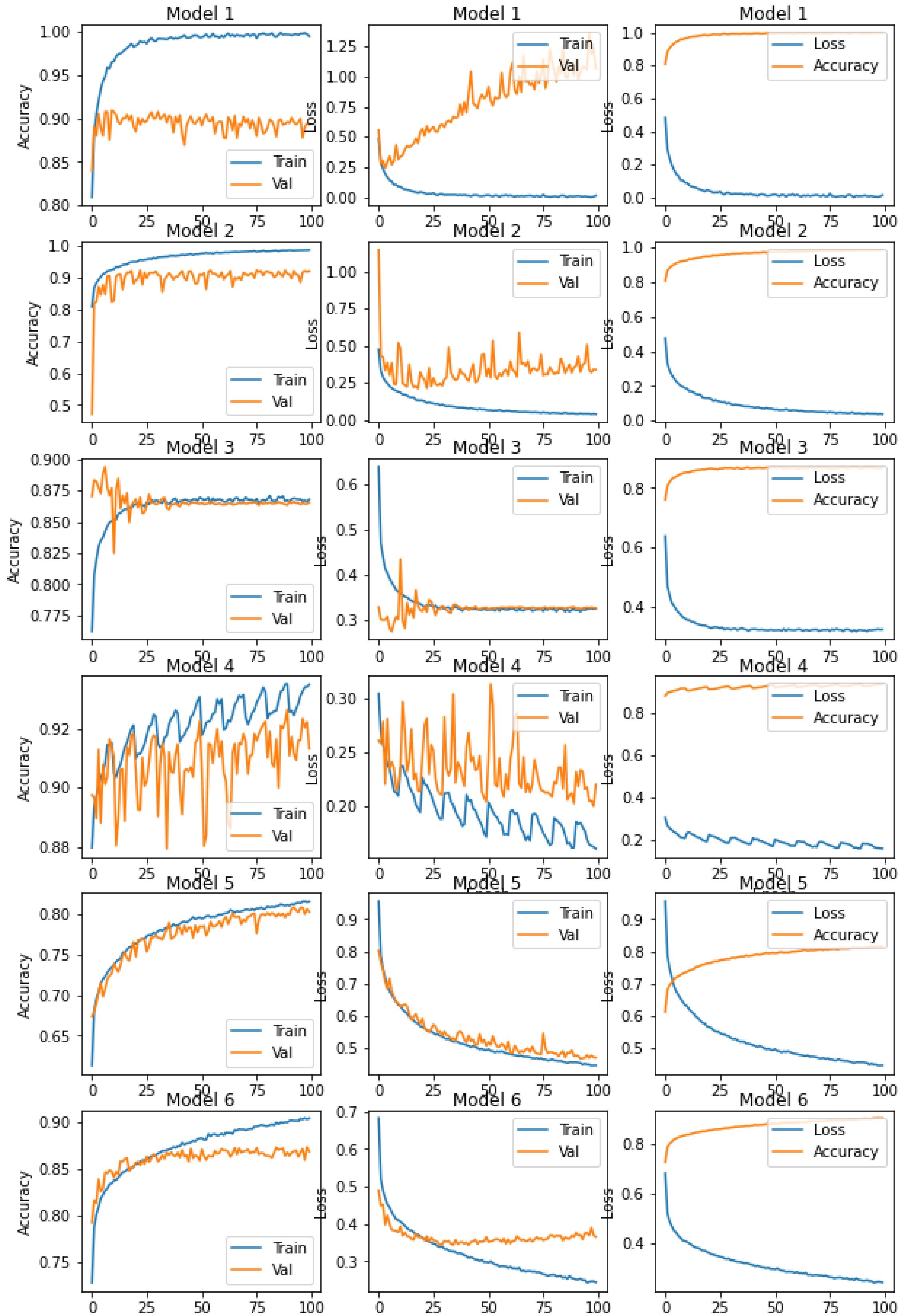
```
<matplotlib.legend.Legend at 0x7f4ee84bfd10>
```





In [ ]:

```
compare_accuracy_loss(history_list)
```



Kaggle Submission

In [ ]:
```
kaggle_data = np.load('/content/drive/MyDrive/mnist/fashion_mnist_dataset_kaggle
```

In [ ]:
```
kaggle_data
```

In [ ]:
```
# Predicting the test set result with optimized model
kaggle_data_test = kaggle_data['features'].reshape((-1, 28, 28, 1))
scores = model.predict(kaggle_data_test)
#convert outputs from 0 - 4 to 1 -5
```

In [ ]:
```
#visualize
predictions = np.argmax(scores, axis=1)
predictions[0:20]
```

Out[ ]:
```
array([3, 2, 1, 0, 0, 1, 1, 0, 2, 4, 2, 3, 4, 1, 3, 0, 0, 2, 4, 1])
```

In [ ]:
```
targets = replace_values(predictions, [0, 1, 2, 3, 4], [1, 2, 3, 4, 5])
```

In [ ]:
```
result = pd.DataFrame(columns=['id','target'])
result['id'] = kaggle_data['id']
result['target'] = targets

result.set_index(keys='id', inplace=True)

result.to_csv('mnist_submission.csv')
from google.colab import files
files.download("mnist_submission.csv")
```

References

1. https://www.kaggle.com/sainiamit/fashion-mnist-with-92-accuracy-in-cnn
2. https://www.microsoft.com/en-us/research/wp-content/uploads/2003/08/icdar03.pdf
3. https://arxiv.org/abs/1608.03983 4.https://www.kaggle.com/babbler/mnist-data-augmentation-with-elastic-distortion
4. https://www.kaggle.com/residentmario/keras-optimizers?scriptVersionId=8011721&cellId=15