```
#Import file from Google Drive

from google.colab import drive

drive.mount('/content/drive/')

#Drive Location
#!ls '/content/drive/My Drive/CPS 580 Project File/Species Classifier Dataset/train_X.pkl'
#!ls '/content/drive/My Drive/CPS 580 Project File/Species Classifier Dataset/test_X.pkl'
#!ls '/content/drive/My Drive/CPS 580 Project File/Species Classifier Dataset/train_Y.pkl'
#!ls '/content/drive/My Drive/CPS 580 Project File/Species Classifier Dataset/test_Y.pkl'
```

```
     Mounted at /content/drive/
```

```
#Set the data values to train_X, test_X, val_X, train_Y, test_Y, and val_Y
import pickle

#Unpickle a Python File and Put it onto data variable
with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_X.pkl
  train_X = pickle.load(X1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_X.pkl'
  test_X = pickle.load(X2, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_Y.pkl
  train_Y = pickle.load(Y1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_Y.pkl'
  test_Y = pickle.load(Y2, encoding = 'latin1')
```

```
#Double Check if the shapes of the datasets are what you saved as
import numpy as np

train_X = train_X[:1600, ]
train_Y = train_Y[:1600, ]
val_X = train_X[-400:,]
val_Y = train_Y[-400:,]

print("Shape of train_X =", np.array(train_X).shape)
print("Shape of train_Y =", np.array(train_Y).shape)

print("Shape of test_X =", np.array(test_X).shape)
print("Shape of test_Y =", np.array(test_Y).shape)

print("Shape of val_X =", np.array(val_X).shape)
print("Shape of val_Y =", np.array(val_Y).shape)
```

```
     Shape of train_X = (1600, 75, 200, 3)
     Shape of train_Y = (1600, 4)
```

```
Shape of test_X = (411, 75, 200, 3)
Shape of test_Y = (411, 4)
Shape of val_X = (400, 75, 200, 3)
Shape of val_Y = (400, 4)
```

```
#Normalize train_X, test_X, val_X
train_X = train_X / 255
test_X = test_X / 255
```

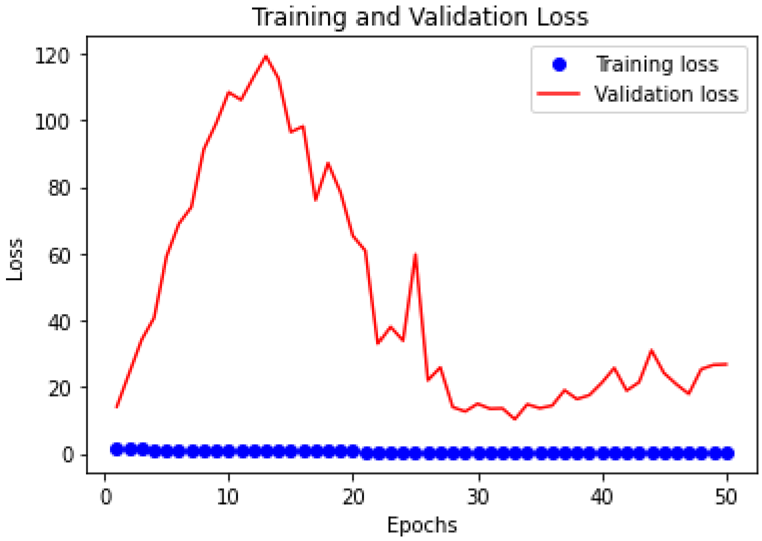
Build Model as a Classifier

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.datasets import mnist
from keras import utils, regularizers, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Convolutional layers
cnn_model = Sequential()
cnn_model.add(Conv2D(16, (3, 3), activation='selu', input_shape=(75, 200, 3)))
cnn_model.add(MaxPooling2D((2, 2)))

cnn_model.add(Conv2D(8, (3, 3), activation='selu'))
cnn_model.add(MaxPooling2D((2, 2)))

cnn_model.add(Conv2D(8, (3, 3), activation='selu'))
cnn_model.add(MaxPooling2D((2, 2)))

cnn_model.add(Conv2D(8, (3, 3), activation='selu'))
cnn_model.add(MaxPooling2D((2, 2)))

# Dense, fully connected layers
cnn_model.add(Flatten())
cnn_model.add(Dense(16, activation='selu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(16, activation='selu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation='softmax'))

# Specify the loss, optimizer and any additional metrics to follow
#opt = optimizers.Adam(learning_rate = 0.0)
cnn_model.compile(loss='categorical_crossentropy', optimizer= 'rmsprop', metrics=['accuracy']

cnn_model.summary()
# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 50, batch_size = 50, validation_data= (val

cnn_model.evaluate(test_X, test_Y)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 73, 198, 16)       448       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 36, 99, 16)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 34, 97, 8)         1160      
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 17, 48, 8)         0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 46, 8)         584       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 23, 8)          0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 21, 8)          584       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 10, 8)          0         
_________________________________________________________________
flatten (Flatten)            (None, 160)               0         
_________________________________________________________________
dense (Dense)                (None, 16)                2576      
_________________________________________________________________
dropout (Dropout)            (None, 16)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272       
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 68        
=================================================================
Total params: 5,692
Trainable params: 5,692
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
32/32 [==============================] - 1s 19ms/step - loss: 1.6587 - accuracy: 0.26
Epoch 2/50
32/32 [==============================] - 0s 13ms/step - loss: 1.3871 - accuracy: 0.32
Epoch 3/50
32/32 [==============================] - 0s 14ms/step - loss: 1.2950 - accuracy: 0.38
Epoch 4/50
32/32 [==============================] - 0s 14ms/step - loss: 1.2068 - accuracy: 0.45
Epoch 5/50
32/32 [==============================] - 0s 13ms/step - loss: 1.1562 - accuracy: 0.46
Epoch 6/50
32/32 [==============================] - 0s 13ms/step - loss: 1.0602 - accuracy: 0.49
Epoch 7/50
32/32 [==============================] - 0s 14ms/step - loss: 1.0108 - accuracy: 0.53
Epoch 8/50
32/32 [==============================] - 0s 14ms/step - loss: 0.9570 - accuracy: 0.56
Epoch 9/50
32/32 [==============================] - 0s 14ms/step - loss: 0.9136 - accuracy: 0.55
Epoch 10/50
32/32 [==============================] - 0s 14ms/step - loss: 0.9095 - accuracy: 0.58
Epoch 11/50
```

```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

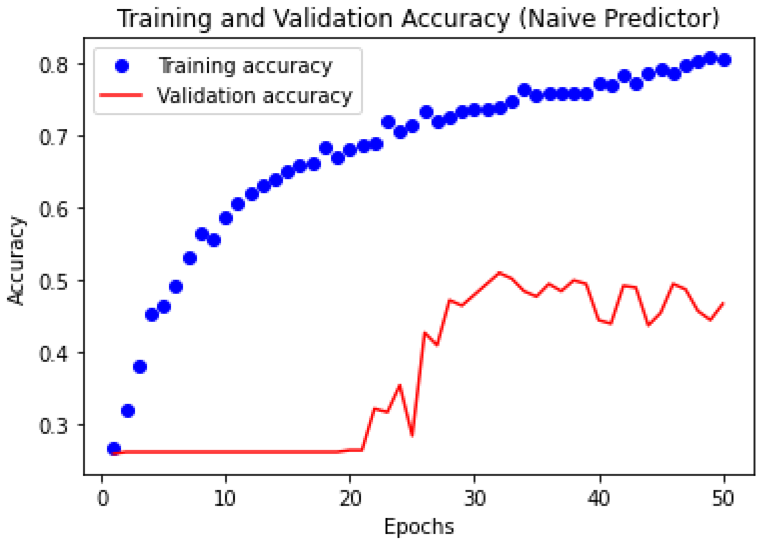
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```



```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)
```

```
plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (Naive Predictor)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```



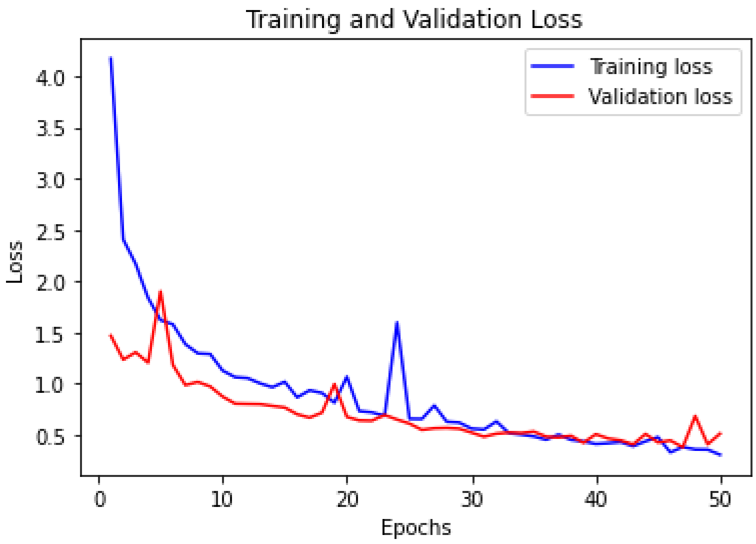
K-fold Validation

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.datasets import mnist
from keras import utils, regularizers, optimizers
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_X.pkl
  train_data = pickle.load(X1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_Y.pkl
  train_targets = pickle.load(Y1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_X.pkl'
  test_X = pickle.load(X2, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_Y.pkl'
  test_Y = pickle.load(Y2, encoding = 'latin1')

k = 5
num_val_samples = len(train_data) // k
num_epochs = 50
```

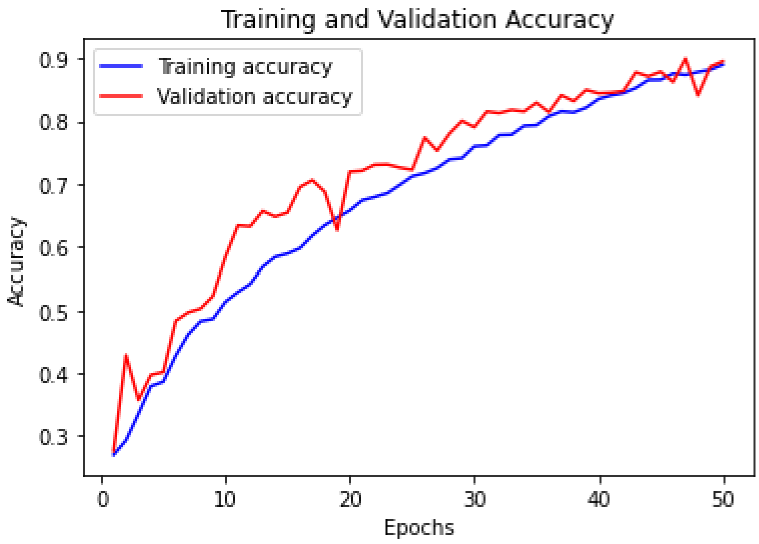
```
all_scores = []

print("Shape of train_X =", np.array(train_data).shape)
print("Shape of train_Y =", np.array(train_targets).shape)

#Normalize train_X, test_X, val_X
train_data = train_data / 255.

# Convolutional layers
def build_model():
  cnn_model = Sequential()
  cnn_model.add(Conv2D(256, (3, 3), activation='elu', input_shape=(75, 200, 3)))
  cnn_model.add(MaxPooling2D((2, 2)))

  cnn_model.add(Conv2D(128, (3, 3), activation='elu'))
  cnn_model.add(MaxPooling2D((2, 2)))

  cnn_model.add(Conv2D(64, (3, 3), activation='elu'))
  cnn_model.add(MaxPooling2D((2, 2)))

  # Dense, fully connected layers
  cnn_model.add(Flatten())
  cnn_model.add(Dense(64, activation='elu'))
  cnn_model.add(Dropout(0.5))
  cnn_model.add(Dense(16, activation='elu'))
  cnn_model.add(Dropout(0.5))
  cnn_model.add(Dense(4, activation='softmax'))
  cnn_model.compile(loss='categorical_crossentropy', optimizer= 'rmsprop', metrics=['accuracy

  return cnn_model

all_loss_hist = []
all_acc_hist = []
all_valloss_hist = []
all_valacc_hist = []

for i in range(k):
  print("Processing Fold #:", i)

  val_data = train_data[i * num_val_samples: (i+1) * num_val_samples]
  val_targets = train_targets[i * num_val_samples: (i+1) * num_val_samples]

  partial_train_data = np.concatenate(
      [train_data[:i * num_val_samples],
       train_data[(i+1) * num_val_samples:]], axis = 0
  )

  partial_train_targets = np.concatenate(
      [train_targets[:i * num_val_samples],
       train_targets[(i+1) * num_val_samples:]], axis = 0
  )
  #Augment the Training Data
  #datagen = ImageDataGenerator(rotation_range = 10, width_shift_range = 0.1, height_shift_ra
  #batch_size = 50

  cnn_model = build_model()

  # Train the model with the training data, set epochs and batch size
  history = cnn_model.fit(partial_train_data, partial_train_targets, epochs = num_epochs, bat
  #val_loss, val_acc = cnn_model.evaluate(val_data, val_targets)
  #all_scores.append(val_acc)
  cnn_model.evaluate(test_X, test_Y)

  loss_hist = history.history['loss']
  acc_hist = history.history['accuracy']
  val_loss_hist = history.history['val_loss']
  val_acc_hist = history.history['val_accuracy']

  all_loss_hist.append(loss_hist)
  all_acc_hist.append(acc_hist)
  all_valloss_hist.append(val_loss_hist)
  all_valacc_hist.append(val_acc_hist)


#all_scores = np.array(all_scores)
#print(np.mean(all_scores))
```

```
Shape of train_X = (2000, 75, 200, 3)
Shape of train_Y = (2000, 4)
Processing Fold #: 0
13/13 [==============================] - 1s 41ms/step - loss: 17799.8789 - accuracy: 0.2
Processing Fold #: 1
13/13 [==============================] - 0s 23ms/step - loss: 17047.9062 - accuracy: 0.2
Processing Fold #: 2
13/13 [==============================] - 0s 22ms/step - loss: 5939.4644 - accuracy: 0.37
Processing Fold #: 3
13/13 [==============================] - 0s 22ms/step - loss: 25676.7324 - accuracy: 0.2
Processing Fold #: 4
13/13 [==============================] - 0s 22ms/step - loss: 303.3780 - accuracy: 0.88
```

```
avg_val_loss = [np.mean([x[i] for x in all_valloss_hist]) for i in range(num_epochs)]
avg_val_acc = [np.mean([x[i] for x in all_valacc_hist]) for i in range(num_epochs)]
avg_acc = [np.mean([x[i] for x in all_acc_hist]) for i in range(num_epochs)]
avg_loss = [np.mean([x[i] for x in all_loss_hist]) for i in range(num_epochs)]


#Plotting training and validation loss
import matplotlib.pyplot as plt

epochs = range(1, len(avg_loss) + 1)

plt.plot(epochs, avg_loss, 'b', label = 'Training loss')
plt.plot(epochs, avg_val_loss, 'r', label = 'Validation loss')
plt.plot(epochs, avg_val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
import matplotlib.pyplot as plt


epochs = range(1, len(avg_acc) + 1)

plt.plot(epochs, avg_acc, 'b', label = 'Training accuracy')
plt.plot(epochs, avg_val_acc, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



Data Augmentation

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.datasets import mnist
from keras import utils, regularizers, optimizers
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_X.pkl
  train_X = pickle.load(X1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_X.pkl'
  test_X = pickle.load(X2, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_Y.pkl
  train_Y = pickle.load(Y1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_Y.pkl'
  test_Y = pickle.load(Y2, encoding = 'latin1')

train_X = train_X[:1600, ]
train_Y = train_Y[:1600, ]
val_X = train_X[-400:,]
val_Y = train_Y[-400:,]

#Normalize train_X, test_X, val_X
train_X = train_X / 255
test_X = test_X / 255


#Build a Model
cnn_model = Sequential()
cnn_model.add(Conv2D(32, (3, 3), activation='elu', input_shape=(75, 200, 3)))
cnn_model.add(MaxPooling2D((2, 2)))

cnn_model.add(Conv2D(8, (3, 3), activation='elu'))
cnn_model.add(MaxPooling2D((2, 2)))

cnn_model.add(Conv2D(8, (3, 3), activation='elu'))
cnn_model.add(MaxPooling2D((2, 2)))

# Dense, fully connected layers
cnn_model.add(Flatten())
cnn_model.add(Dense(128, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(32, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dense(4, activation='softmax'))
cnn_model.compile(loss='categorical_crossentropy', optimizer= 'rmsprop', metrics=['accuracy']

#Augment the Training Data
datagen = ImageDataGenerator(rotation_range = 5, width_shift_range = 0.01, height_shift_range
batch_size = 50

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(datagen.flow(train_X, train_Y, batch_size = batch_size), steps_per_ep

#Test it on the Test Data
cnn_model.evaluate(test_X, test_Y)
```

```
Epoch 1/100
32/32 [==============================] - 5s 150ms/step - loss: 4.5471 - accuracy: 0.24
Epoch 2/100
32/32 [==============================] - 5s 147ms/step - loss: 3.3357 - accuracy: 0.4
Epoch 3/100
32/32 [==============================] - 5s 147ms/step - loss: 2.4745 - accuracy: 0.54
Epoch 4/100
32/32 [==============================] - 5s 148ms/step - loss: 1.9868 - accuracy: 0.60
Epoch 5/100
32/32 [==============================] - 5s 147ms/step - loss: 1.6101 - accuracy: 0.6
Epoch 6/100
32/32 [==============================] - 5s 149ms/step - loss: 1.4469 - accuracy: 0.6
Epoch 7/100
32/32 [==============================] - 5s 153ms/step - loss: 1.2331 - accuracy: 0.70
Epoch 8/100
32/32 [==============================] - 5s 147ms/step - loss: 1.0997 - accuracy: 0.7
Epoch 9/100
32/32 [==============================] - 5s 148ms/step - loss: 1.1357 - accuracy: 0.6
Epoch 10/100
32/32 [==============================] - 5s 146ms/step - loss: 0.9291 - accuracy: 0.7
Epoch 11/100
32/32 [==============================] - 5s 147ms/step - loss: 0.9680 - accuracy: 0.74
Epoch 12/100
32/32 [==============================] - 5s 147ms/step - loss: 0.8641 - accuracy: 0.7
Epoch 13/100
32/32 [==============================] - 5s 148ms/step - loss: 0.8414 - accuracy: 0.74
Epoch 14/100
32/32 [==============================] - 5s 148ms/step - loss: 0.7992 - accuracy: 0.7
Epoch 15/100
32/32 [==============================] - 5s 148ms/step - loss: 0.8502 - accuracy: 0.74
Epoch 16/100
32/32 [==============================] - 5s 147ms/step - loss: 0.7561 - accuracy: 0.7
Epoch 17/100
32/32 [==============================] - 5s 148ms/step - loss: 0.7161 - accuracy: 0.7
Epoch 18/100
32/32 [==============================] - 5s 146ms/step - loss: 0.6725 - accuracy: 0.80
Epoch 19/100
32/32 [==============================] - 5s 147ms/step - loss: 0.7675 - accuracy: 0.7
Epoch 20/100
32/32 [==============================] - 5s 147ms/step - loss: 0.6687 - accuracy: 0.79
Epoch 21/100
32/32 [==============================] - 5s 147ms/step - loss: 0.6510 - accuracy: 0.80
Epoch 22/100
32/32 [==============================] - 5s 145ms/step - loss: 0.6957 - accuracy: 0.80
```

```
Epoch 23/100
32/32 [==============================] - 5s 146ms/step - loss: 0.6265 - accuracy: 0.8
Epoch 24/100
32/32 [==============================] - 5s 147ms/step - loss: 0.6596 - accuracy: 0.8
Epoch 25/100
32/32 [==============================] - 5s 147ms/step - loss: 0.6345 - accuracy: 0.8
Epoch 26/100
32/32 [==============================] - 5s 147ms/step - loss: 0.6118 - accuracy: 0.8
Epoch 27/100
32/32 [==============================] - 5s 147ms/step - loss: 0.5881 - accuracy: 0.84
Epoch 28/100
32/32 [==============================] - 5s 147ms/step - loss: 0.5639 - accuracy: 0.84
Epoch 29/100
32/32 [==============================] - 5s 148ms/step - loss: 0.6182 - accuracy: 0.80
```

Try VGG16

```
import numpy as np
import keras
from keras import backend, optimizers
from keras.models import Sequential
from keras.layers import Activation
from keras.layers.core import Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import *
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
import pickle
from keras import utils, regularizers, optimizers
%matplotlib inline

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_X.pkl
  train_X = pickle.load(X1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_X.pkl'
  test_X = pickle.load(X2, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_Y.pkl
  train_Y = pickle.load(Y1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_Y.pkl'
  test_Y = pickle.load(Y2, encoding = 'latin1')

train_X = train_X[:1600, ]
train_Y = train_Y[:1600, ]
val_X = train_X[-400:,]
val_Y = train_Y[-400:,]

train_X = train_X / 255.
```

```
test_X = test_X / 255.
```

Hyperparameter Testing 1

```
vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
58892288/58889256 [==============================] - 1s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 75, 200, 3)]      0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
=================================================================
Total params: 14,714,688
```

```
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
tensorflow.python.keras.engine.functional.Functional
```

```
cnn_model = Sequential()
for layer in vgg16_model.layers:
  cnn_model.add(layer)

cnn_model.summary()
type(cnn_model)
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

```
_________________________________________________________________
tensorflow.python.keras.engine.sequential.Sequential
```

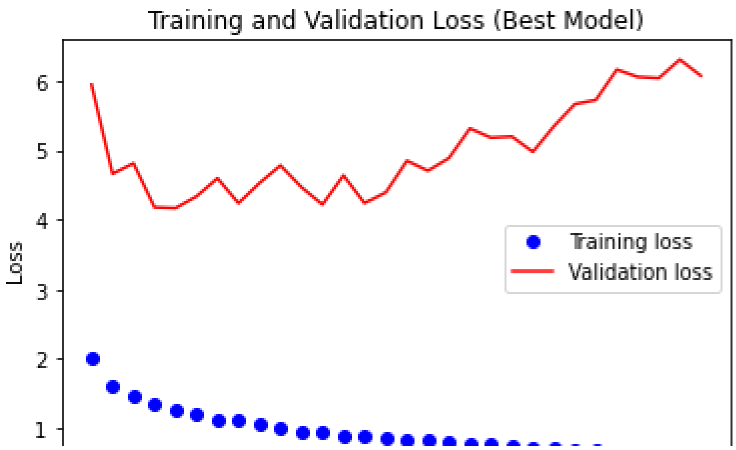
```
for layer in cnn_model.layers:
  layer.trainable = False

cnn_model.add(Flatten())
cnn_model.add(Dense(32, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(16, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation = 'softmax'))
cnn_model.summary()

optimizer = keras.optimizers.Nadam(lr = 0.0001)
cnn_model.compile(optimizer = optimizer, loss = 'categorical_crossentropy', metrics = ['accur

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 30, batch_size = 10, validation_data=(val_

cnn_model.evaluate(test_X, test_Y)
```

```
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 6144)              0
_________________________________________________________________
dense_22 (Dense)             (None, 32)                196640
_________________________________________________________________
dropout_14 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_23 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_15 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_24 (Dense)             (None, 4)                 68
=================================================================
Total params: 14,911,924
Trainable params: 197,236
Non-trainable params: 14,714,688
_________________________________________________________________
Epoch 1/30
  1/160 [..............................] - ETA: 0s - loss: 2.6692 - accuracy: 0.2000W
158/160 [============================>.] - ETA: 0s - loss: 2.0084 - accuracy: 0.4335W
160/160 [==============================] - 5s 29ms/step - loss: 2.0075 - accuracy: 0.4
Epoch 2/30
160/160 [==============================] - 5s 28ms/step - loss: 1.6014 - accuracy: 0.
```

```
Epoch 3/30
160/160 [==============================] - 5s 28ms/step - loss: 1.4558 - accuracy: 0.
Epoch 4/30
160/160 [==============================] - 4s 28ms/step - loss: 1.3350 - accuracy: 0.
Epoch 5/30
160/160 [==============================] - 4s 28ms/step - loss: 1.2688 - accuracy: 0.
Epoch 6/30
160/160 [==============================] - 4s 28ms/step - loss: 1.1909 - accuracy: 0.
Epoch 7/30
160/160 [==============================] - 4s 27ms/step - loss: 1.1179 - accuracy: 0.
Epoch 8/30
160/160 [==============================] - 4s 27ms/step - loss: 1.0998 - accuracy: 0.
Epoch 9/30
160/160 [==============================] - 4s 27ms/step - loss: 1.0575 - accuracy: 0.
Epoch 10/30
160/160 [==============================] - 4s 27ms/step - loss: 1.0041 - accuracy: 0.
Epoch 11/30
160/160 [==============================] - 4s 27ms/step - loss: 0.9468 - accuracy: 0.
Epoch 12/30
160/160 [==============================] - 4s 27ms/step - loss: 0.9445 - accuracy: 0.
Epoch 13/30
160/160 [==============================] - 4s 27ms/step - loss: 0.8922 - accuracy: 0.
Epoch 14/30
160/160 [==============================] - 4s 27ms/step - loss: 0.8734 - accuracy: 0.
Epoch 15/30
160/160 [==============================] - 4s 27ms/step - loss: 0.8594 - accuracy: 0.
```

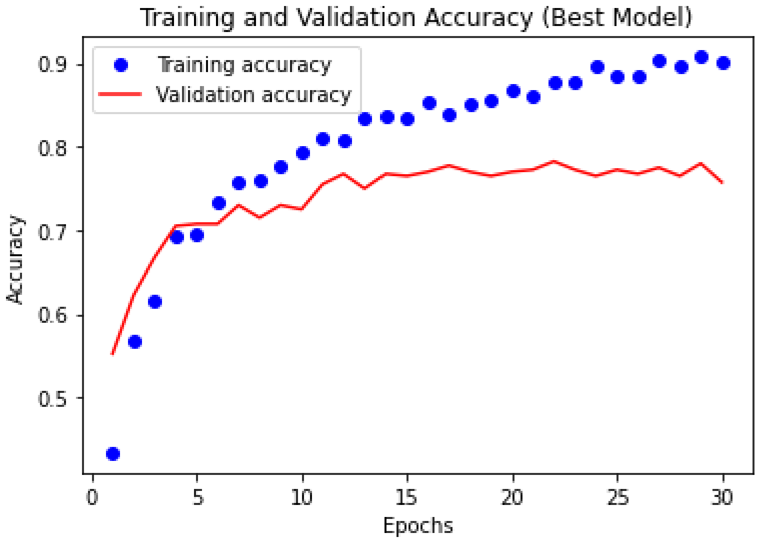
```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss (Best Model)")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 31)
```



```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (Best Model)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 31)
```



Hyperparameter Testing 2

```
vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)

cnn_model = Sequential()
for layer in vgg16_model.layers:
  cnn_model.add(layer)

cnn_model.summary()
type(cnn_model)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         [(None, 75, 200, 3)]      0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
```

```
for layer in cnn_model.layers:
  layer.trainable = False

cnn_model.add(Flatten())
cnn_model.add(Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(16, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation = 'softmax'))
cnn_model.summary()

#optimizer = keras.optimizers.Nadam(lr = 0.0001)
cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 50, batch_size = 15, validation_data=(val_

cnn_model.evaluate(test_X, test_Y)
```

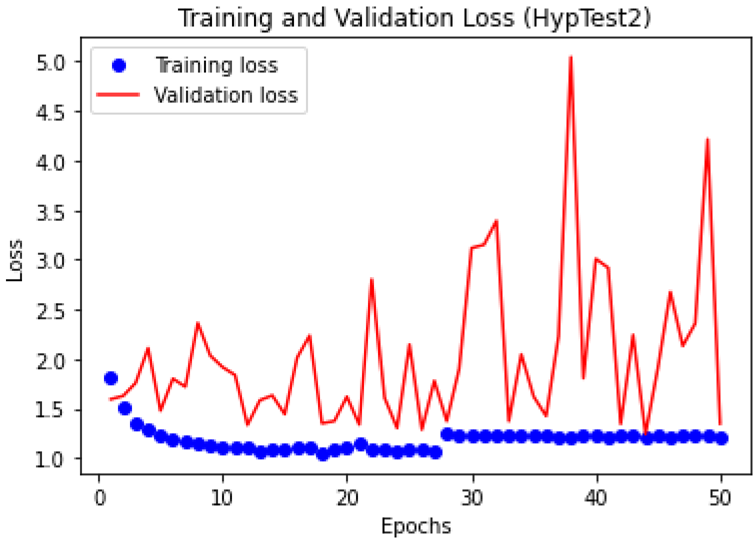
```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
```

```
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
_________________________________________________________________
flatten_8 (Flatten)          (None, 6144)              0         
_________________________________________________________________
dense_25 (Dense)             (None, 32)                196640    
_________________________________________________________________
dropout_16 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_26 (Dense)             (None, 16)                528       
_________________________________________________________________
dropout_17 (Dropout)         (None, 16)                0         
_________________________________________________________________
dense_27 (Dense)             (None, 4)                 68        
=================================================================
Total params: 14,911,924
Trainable params: 197,236
Non-trainable params: 14,714,688
_________________________________________________________________
Epoch 1/50
107/107 [==============================] - 4s 35ms/step - loss: 1.8096 - accuracy: 0.
```

```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)
```

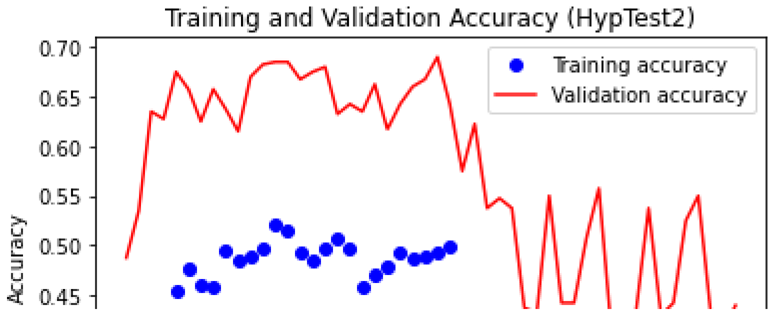
```
plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss (HypTest2)")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```



```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (HypTest2)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```



Hyperparameter Testing 3

```
vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)

cnn_model = Sequential()
for layer in vgg16_model.layers:
  cnn_model.add(layer)

cnn_model.summary()
type(cnn_model)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         [(None, 75, 200, 3)]      0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
```

```
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
```

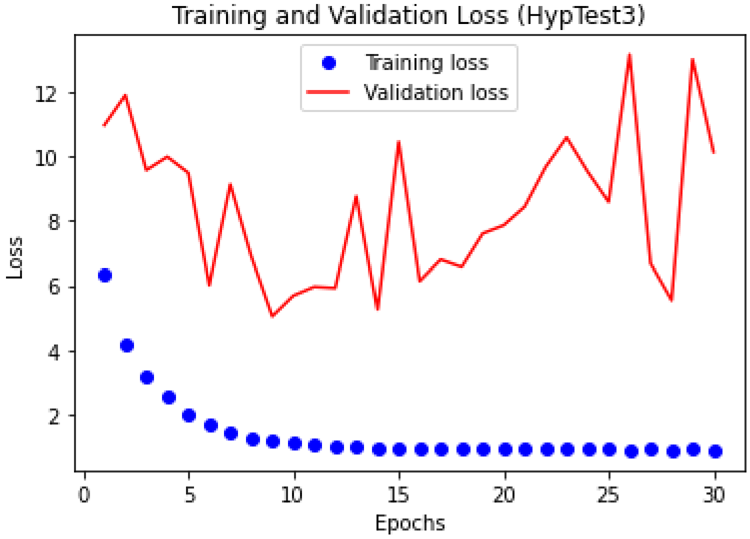
```
for layer in cnn_model.layers:
  layer.trainable = False

cnn_model.add(Flatten())
cnn_model.add(Dense(256, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(128, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(128, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation = 'softmax'))
cnn_model.summary()

#optimizer = keras.optimizers.Nadam(lr = 0.0001)
cnn_model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accur

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 30, batch_size = 10, validation_data=(val_

cnn_model.evaluate(test_X, test_Y)
```

```
Model: "sequential_9"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
_________________________________________________________________
flatten_9 (Flatten)          (None, 6144)              0         
_________________________________________________________________
dense_28 (Dense)             (None, 256)               1573120   
_________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_29 (Dense)             (None, 128)               32896     
_________________________________________________________________
dropout_19 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_30 (Dense)             (None, 128)               16512     
_________________________________________________________________
dropout_20 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_31 (Dense)             (None, 4)                 516       
=================================================================
Total params: 16,337,732
Trainable params: 1,623,044
```

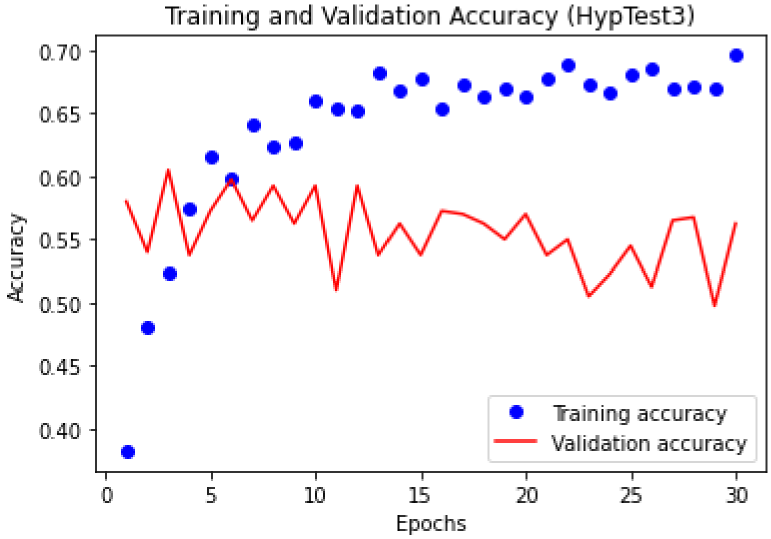
```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss (HypTest3)")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 31)
```



```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (HypTest3)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 31)
```



Hyperparameter Testing 4

```
vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)

cnn_model = Sequential()
for layer in vgg16_model.layers:
  cnn_model.add(layer)

cnn_model.summary()
type(cnn_model)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 75, 200, 3)]      0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
```

```
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
```

```
for layer in cnn_model.layers:
  layer.trainable = False

cnn_model.add(Flatten())
cnn_model.add(Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(64, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation = 'softmax'))
cnn_model.summary()

optimizer = keras.optimizers.Nadam(lr = 0.0001)
cnn_model.compile(optimizer = 'nadam', loss = 'categorical_crossentropy', metrics = ['accurac

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 50, batch_size = 16, validation_data=(val_

cnn_model.evaluate(test_X, test_Y)
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
_________________________________________________________________
flatten_10 (Flatten)         (None, 6144)              0
_________________________________________________________________
dense_32 (Dense)             (None, 64)                393280
_________________________________________________________________
```

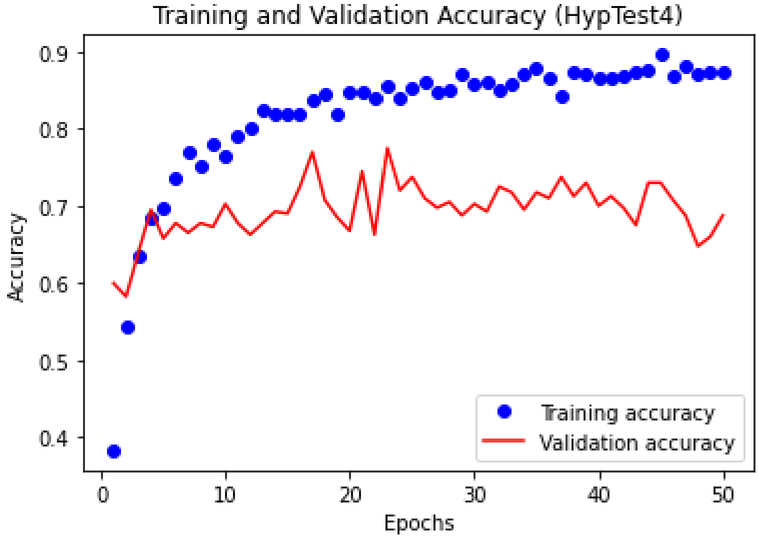
```
dropout_21 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_33 (Dense)             (None, 64)                4160      
_________________________________________________________________
dropout_22 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_34 (Dense)             (None, 64)                4160      
_________________________________________________________________
dropout_23 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_35 (Dense)             (None, 4)                 260       
=================================================================
Total params: 15,116,548
Trainable params: 401,860
```

```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss (HypTest4)")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```

```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (HypTest4)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 51)
```



Hyperparaemeter Testing 5

```
vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)

cnn_model = Sequential()
for layer in vgg16_model.layers:
  cnn_model.add(layer)

cnn_model.summary()
cnn_model.summary()
type(cnn_model)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_5 (InputLayer)         [(None, 75, 200, 3)]      0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928     
_________________________________________________________________
```

```
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
```

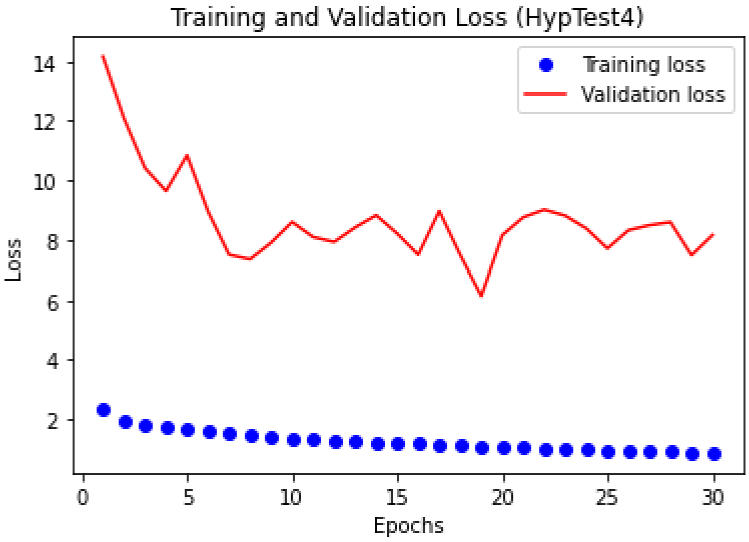
```
for layer in cnn_model.layers:
  layer.trainable = False

cnn_model.add(Flatten())
cnn_model.add(Dense(32, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(16, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(16, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
cnn_model.add(Dropout(0.5))
cnn_model.add(Dense(4, activation = 'softmax'))
cnn_model.summary()

optimizer = keras.optimizers.Nadam(lr = 0.0001)
cnn_model.compile(optimizer = optimizer, loss = 'categorical_crossentropy', metrics = ['accur

# Train the model with the training data, set epochs and batch size
history = cnn_model.fit(train_X, train_Y, epochs = 30, batch_size = 10, validation_data=(val_

cnn_model.evaluate(test_X, test_Y)
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
```

```
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
_________________________________________________________________
flatten_11 (Flatten)         (None, 6144)              0
_________________________________________________________________
dense_36 (Dense)             (None, 32)                196640
_________________________________________________________________
dropout_24 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_37 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_25 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_38 (Dense)             (None, 16)                272
_________________________________________________________________
dropout_26 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_39 (Dense)             (None, 4)                 68
=================================================================
Total params: 14,912,196
Trainable params: 197,508
```

```
history_dict = history.history
print(history_dict.keys())

import matplotlib.pyplot as plt

history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss (HypTest4)")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
range(1, 31)
```



```
history_dict = history.history
print(history_dict.keys())

train_loss = history_dict['accuracy']
val_loss = history_dict['val_accuracy']

epochs = range(1, len(history_dict['accuracy']) + 1)
print(epochs)

plt.plot(epochs, train_loss, 'bo', label = 'Training accuracy')
plt.plot(epochs, val_loss, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy (HypTest4)")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

VGG16 with K-fold validation

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
from keras.datasets import mnist
from keras import utils, regularizers, optimizers
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import keras
from keras import backend
from keras.models import Sequential
from keras.layers.convolutional import *
import pickle
import itertools
%matplotlib inline

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_X.pkl
  train_data = pickle.load(X1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_X.pkl'
  test_data = pickle.load(X2, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/train_Y.pkl
  train_targets = pickle.load(Y1, encoding = 'latin1')

with open('/content/drive/My Drive/CPS 580 Project File/Source Classifier Dataset/test_Y.pkl'
  test_Y = pickle.load(Y2, encoding = 'latin1')

train_data = train_data / 255.
test_data = test_data / 255.


vgg16_model = keras.applications.vgg16.VGG16(weights='imagenet', input_shape=(75, 200, 3), in
vgg16_model.summary()

type(vgg16_model)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 75, 200, 3)]      0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 75, 200, 64)       1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 75, 200, 64)       36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 37, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 37, 100, 128)      73856
```

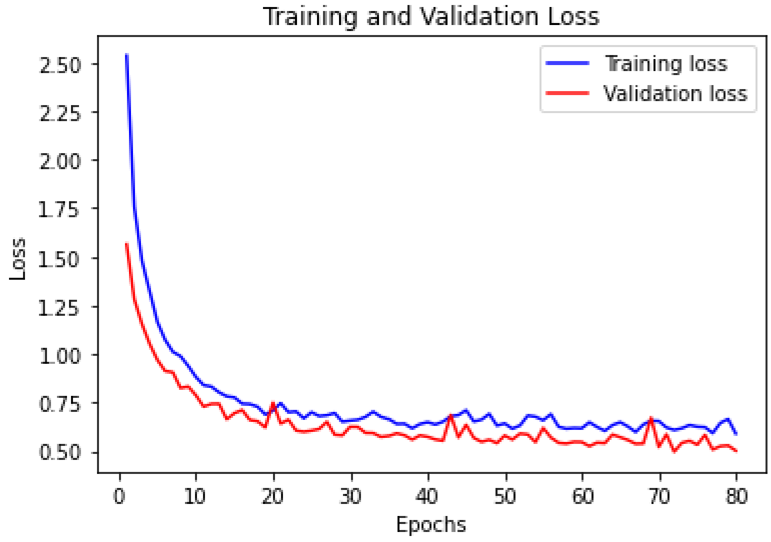
```
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 37, 100, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 18, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 18, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 18, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 9, 25, 256)        0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 9, 25, 512)        1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 9, 25, 512)        2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 6, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
tensorflow.python.keras.engine.functional.Functional
```

```
k = 5
num_val_samples = len(train_data) // k
num_epochs = 80
all_scores = []

print("Shape of train_X =", np.array(train_data).shape)
print("Shape of train_Y =", np.array(train_targets).shape)

#Normalize train_X, test_X, val_X

# Convolutional layers
def build_model():
  cnn_model = Sequential()
  for layer in vgg16_model.layers:
    cnn_model.add(layer)

  for layer in cnn_model.layers:
    layer.trainable = False

  cnn_model.add(Flatten())
  cnn_model.add(Flatten())
  cnn_model.add(Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.01)))
  cnn_model.add(Dropout(0.5))
  cnn_model.add(Dense(32, kernel_regularizer = regularizers.l2(0.01), activation='elu'))
  cnn_model.add(Dropout(0.5))
  cnn_model.add(Dense(4, activation = 'softmax'))

  return cnn_model

all_loss_hist = []
all_acc_hist = []
all_valloss_hist = []
all_valacc_hist = []

#datagen = ImageDataGenerator(rotation_range = 5, width_shift_range = 0.01, height_shift_rang
#batch_size = 40

for i in range(k):
  print("Processing Fold #:", i)

  val_data = train_data[i * num_val_samples: (i+1) * num_val_samples]
  val_targets = train_targets[i * num_val_samples: (i+1) * num_val_samples]

  partial_train_data = np.concatenate(
      [train_data[:i * num_val_samples],
       train_data[(i+1) * num_val_samples:]], axis = 0
  )

  partial_train_targets = np.concatenate(
      [train_targets[:i * num_val_samples],
       train_targets[(i+1) * num_val_samples:]], axis = 0
  )

  cnn_model = build_model()

  # Train the model with the training data, set epochs and batch size
  cnn_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accura

  # Train the model with the training data, set epochs and batch size
  history = cnn_model.fit(partial_train_data, partial_train_targets, epochs = num_epochs, bat

  cnn_model.evaluate(test_X, test_Y)

  loss_hist = history.history['loss']
  acc_hist = history.history['accuracy']
  val_loss_hist = history.history['val_loss']
  val_acc_hist = history.history['val_accuracy']

  all_loss_hist.append(loss_hist)
  all_acc_hist.append(acc_hist)
  all_valloss_hist.append(val_loss_hist)
  all_valacc_hist.append(val_acc_hist)
```

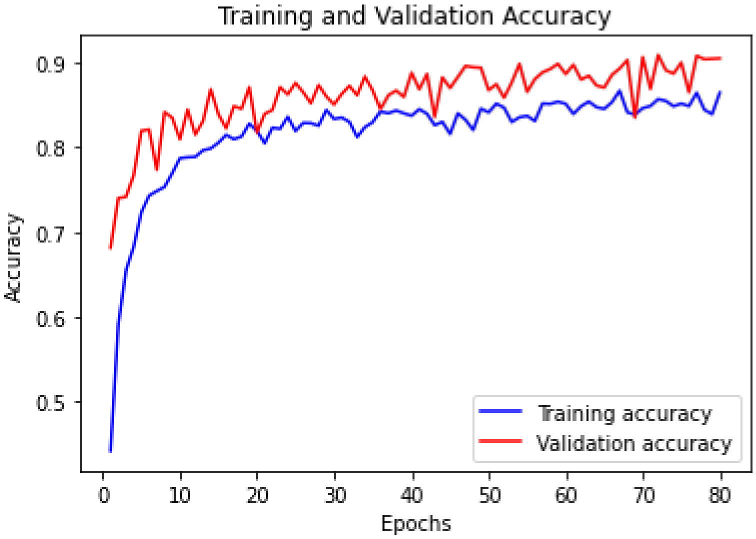
```
Shape of train_X = (2000, 75, 200, 3)
Shape of train_Y = (2000, 4)
Processing Fold #: 0
13/13 [==============================] - 1s 43ms/step - loss: 0.4847 - accuracy: 0.9124
Processing Fold #: 1
13/13 [==============================] - 1s 43ms/step - loss: 0.4785 - accuracy: 0.9075
Processing Fold #: 2
13/13 [==============================] - 1s 43ms/step - loss: 0.4769 - accuracy: 0.9075
Processing Fold #: 3
13/13 [==============================] - 1s 43ms/step - loss: 0.4548 - accuracy: 0.9173
Processing Fold #: 4
13/13 [==============================] - 1s 43ms/step - loss: 0.4699 - accuracy: 0.9367
```

```
avg_val_loss = [np.mean([x[i] for x in all_valloss_hist]) for i in range(num_epochs)]
avg_val_acc = [np.mean([x[i] for x in all_valacc_hist]) for i in range(num_epochs)]
avg_acc = [np.mean([x[i] for x in all_acc_hist]) for i in range(num_epochs)]
avg_loss = [np.mean([x[i] for x in all_loss_hist]) for i in range(num_epochs)]


#Plotting training and validation loss
import matplotlib.pyplot as plt

print(history_dict.keys())

epochs = range(1, len(avg_loss) + 1)

plt.plot(epochs, avg_loss, 'b', label = 'Training loss')
plt.plot(epochs, avg_val_loss, 'r', label = 'Validation loss')
plt.title("Training and Validation Loss")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
epochs = range(1, len(avg_acc) + 1)

plt.plot(epochs, avg_acc, 'b', label = 'Training accuracy')
plt.plot(epochs, avg_val_acc, 'r', label = 'Validation accuracy')
plt.title("Training and Validation Accuracy")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


Training and Validation Accuracy
Accuracy
Epochs
Training accuracy
Validation accuracy