ECE763-Project 3: Babysitting the training of DNN

Problem: To Practice the babysitting method of training neural network

- The dataset used for training the model is the FDDB face images. Nonface images have also been cropped from the background of the same dataset.
- Total training images=1000 (500 face + 500 nonface)
- Total testing images=100 (50 face and 50 non face images)
- Image Size= 20 X 20 X 3 RGB
- The model used for babysitting is a simple 2 layer fully connected network consisting of: one hidden layer with input as the flattened image and 20 neurons followed by ReLU activation, followed by another fully connected layer with 2 outputs corresponding to the two output classes(face and nonface).
- Keras sequential model was used for designing this model, hence the layers added were flatten and dense. Weight regularization used is L2 and optimizer used is SGD(Stochastic gradient descent).
- The steps followed are:
  1. Data acquisition, processing, augmentation
  2. Sanity check and loss check
  3. Hyper parameter tuning
- The process of babysitting with every step in the tuning of Learning Rate and Regularization for face recognition is explained in detail below with code screenshots.
- The babysitting process was also done with CIFAR 10 data set for understanding the process. This is explained later for reference.

Face Detection: babysitting the training of neural network

A. Data Acquisition from dataset folder

Data is already extracted from FDDB and stored in the dataset folder. Images are obtained from folder and arranged into train and test data. Labels are created as ‘0’ for face and ‘1’ for nonface.

```
path=r'/content/drive/My Drive/dataset/'
img_face_train=[]
for i in range(trainsize//2):
    pat= path + 'dataset_face_train/im{}.jpg'.format(i+1)
    image=cv2.imread(pat)
    img_face_train.append(image)
img_face_train=np.array(img_face_train)


img_face_test=[]
for i in range(testsize//2):
    pat= path + 'dataset_face_test/im{}.jpg'.format(i+1)
    image=cv2.imread(pat)
    img_face_test.append(image)
img_face_test=np.array(img_face_test)


img_nonface_train=[]
for i in range(trainsize//2):
    pat= path + 'dataset_nonface_train/im{}.jpg'.format(i+1)
    image=cv2.imread(pat)
    img_nonface_train.append(image)
img_nonface_train=np.array(img_nonface_train)
```

```
img_nonface_test=[]
for i in range(testsize//2):
    pat= path + 'dataset_nonface_test/im{}.jpg'.format(i+1)
    image=cv2.imread(pat)
    img_nonface_test.append(image)
img_nonface_test=np.array(img_nonface_test)

trainY=np.array([[0]]*(trainsize//2) + [[1]]*(trainsize//2))
testY=np.array([[0]]*(testsize//2) + [[1]]*(testsize//2))
trainY = to_categorical(trainY)
testY = to_categorical(testY)
trainX=np.append(img_face_train,img_nonface_train,axis=0)
testX=np.append(img_face_test,img_nonface_test,axis=0)
print(trainX.shape,type(trainX),testX.shape,type(testX))
```

B. Data Preprocessing and Augmentation:

Normalization is done by subtracting mean and dividing by the standard deviation. Augmentation is done using the ImageDataGenerator function in Keras which contains different augmentation schemes.

```
train_norm = trainX.astype('float32')
test_norm = testX.astype('float32')
# normalize to range 0-1
trainX = train_norm / 255.0
testX = test_norm / 255.0

trainX-=np.mean(trainX,axis=0)
trainX/=np.std(trainX,axis=0)
```

```
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
```

C. Defining the network model:

Keras sequential model is obtained as below followed by compilation.

```
# define cnn model
def define_model(W,L):
  model = Sequential()
  model.add(Flatten(input_shape=(20,20,3)))
  model.add(Dense(10, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(W)))
  model.add(Dense(2, activation='softmax'))
  # compile model
  opt = SGD(learning_rate=L, momentum=0.9)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model
```

D. Training the model with different regularizations and learning rates:

```
Weight_Decay=0
Learning_Rate=0.000001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,
                             height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1)
```

Steps in babysitting

1. Checking with and without normalization: for fixed Learning Rate=1e-6 and Regularization=0.01

   Without normalization:

```
Weight_Decay=1e-6;Learning_Rate=0.01
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator()
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 6ms/step - loss: 0.6912 - accuracy: 0.5919 - val_loss: 0.6907 - val_accuracy: 0.6700
Epoch 2/10
15/15 [==============================] - 0s 7ms/step - loss: 0.6905 - accuracy: 0.5534 - val_loss: 0.6903 - val_accuracy: 0.5200
Epoch 3/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6904 - accuracy: 0.5502 - val_loss: 0.6898 - val_accuracy: 0.5000
Epoch 4/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6899 - accuracy: 0.6506 - val_loss: 0.6891 - val_accuracy: 0.7500
Epoch 5/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6887 - accuracy: 0.7667 - val_loss: 0.6884 - val_accuracy: 0.7800
Epoch 6/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6887 - accuracy: 0.6453 - val_loss: 0.6878 - val_accuracy: 0.5400
Epoch 7/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6879 - accuracy: 0.6902 - val_loss: 0.6872 - val_accuracy: 0.7200
Epoch 8/10
15/15 [==============================] - 0s 7ms/step - loss: 0.6865 - accuracy: 0.6634 - val_loss: 0.6865 - val_accuracy: 0.5300
Epoch 9/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6866 - accuracy: 0.7062 - val_loss: 0.6855 - val_accuracy: 0.8400
Epoch 10/10
15/15 [==============================] - 0s 7ms/step - loss: 0.6850 - accuracy: 0.7885 - val_loss: 0.6849 - val_accuracy: 0.5400
<keras.callbacks.callbacks.History at 0x7fabe9653630>
```

   With normalization:

```
Weight_Decay=1e-6;Learning_Rate=0.01
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator()
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 7ms/step - loss: 0.4111 - accuracy: 0.7863 - val_loss: 0.1291 - val_accuracy: 0.9300
Epoch 2/10
15/15 [==============================] - 0s 7ms/step - loss: 0.1523 - accuracy: 0.9370 - val_loss: 0.0771 - val_accuracy: 0.9800
Epoch 3/10
15/15 [==============================] - 0s 8ms/step - loss: 0.1027 - accuracy: 0.9701 - val_loss: 0.0675 - val_accuracy: 0.9800
Epoch 4/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0914 - accuracy: 0.9626 - val_loss: 0.0704 - val_accuracy: 0.9700
Epoch 5/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0673 - accuracy: 0.9771 - val_loss: 0.0605 - val_accuracy: 0.9800
Epoch 6/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0491 - accuracy: 0.9923 - val_loss: 0.0455 - val_accuracy: 1.0000
Epoch 7/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0453 - accuracy: 0.9915 - val_loss: 0.0590 - val_accuracy: 0.9700
Epoch 8/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0341 - accuracy: 0.9927 - val_loss: 0.0404 - val_accuracy: 0.9800
Epoch 9/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0258 - accuracy: 0.9956 - val_loss: 0.0395 - val_accuracy: 0.9800
Epoch 10/10
15/15 [==============================] - 0s 8ms/step - loss: 0.0264 - accuracy: 0.9969 - val_loss: 0.0331 - val_accuracy: 0.9900
<keras.callbacks.callbacks.History at 0x7fabeb744b38>
```

   We get better accuracy with normalization, hence we use normalization

2. Checking with and without Data Augmentation for learning rate=0.00001 and Regularization=1e-6:

Without Augmentation:

```
Weight_Decay=1e-6;Learning_Rate=0.00001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator()
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 6ms/step - loss: 0.7280 - accuracy: 0.5994 - val_loss: 0.7512 - val_accuracy: 0.5400
Epoch 2/10
15/15 [==============================] - 0s 7ms/step - loss: 0.7175 - accuracy: 0.5994 - val_loss: 0.7320 - val_accuracy: 0.5800
Epoch 3/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6766 - accuracy: 0.6389 - val_loss: 0.7126 - val_accuracy: 0.6000
Epoch 4/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6677 - accuracy: 0.6333 - val_loss: 0.6957 - val_accuracy: 0.6300
Epoch 5/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6626 - accuracy: 0.6560 - val_loss: 0.6793 - val_accuracy: 0.6400
Epoch 6/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6459 - accuracy: 0.6592 - val_loss: 0.6656 - val_accuracy: 0.6700
Epoch 7/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6422 - accuracy: 0.6711 - val_loss: 0.6529 - val_accuracy: 0.6800
Epoch 8/10
15/15 [==============================] - 0s 8ms/step - loss: 0.5973 - accuracy: 0.6948 - val_loss: 0.6417 - val_accuracy: 0.6800
Epoch 9/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6093 - accuracy: 0.6963 - val_loss: 0.6314 - val_accuracy: 0.6900
Epoch 10/10
15/15 [==============================] - 0s 8ms/step - loss: 0.6067 - accuracy: 0.6812 - val_loss: 0.6217 - val_accuracy: 0.6900
<keras.callbacks.callbacks.History at 0x7fabea8dd208>
```

With Augmentation:

```
Weight_Decay=1e-6;Learning_Rate=0.00001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 24ms/step - loss: 0.6968 - accuracy: 0.6167 - val_loss: 0.6227 - val_accuracy: 0.6800
Epoch 2/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6612 - accuracy: 0.6132 - val_loss: 0.6159 - val_accuracy: 0.7000
Epoch 3/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6418 - accuracy: 0.6338 - val_loss: 0.6090 - val_accuracy: 0.7000
Epoch 4/10
15/15 [==============================] - 0s 26ms/step - loss: 0.6472 - accuracy: 0.6357 - val_loss: 0.6022 - val_accuracy: 0.7100
Epoch 5/10
15/15 [==============================] - 0s 24ms/step - loss: 0.6229 - accuracy: 0.6571 - val_loss: 0.5957 - val_accuracy: 0.7100
Epoch 6/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6401 - accuracy: 0.6357 - val_loss: 0.5895 - val_accuracy: 0.7100
Epoch 7/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6495 - accuracy: 0.6496 - val_loss: 0.5834 - val_accuracy: 0.7100
Epoch 8/10
15/15 [==============================] - 0s 26ms/step - loss: 0.6155 - accuracy: 0.6677 - val_loss: 0.5774 - val_accuracy: 0.7100
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6218 - accuracy: 0.6912 - val_loss: 0.5715 - val_accuracy: 0.7100
Epoch 10/10
15/15 [==============================] - 0s 25ms/step - loss: 0.6105 - accuracy: 0.6656 - val_loss: 0.5662 - val_accuracy: 0.7100
<keras.callbacks.callbacks.History at 0x7fabe9ebb278>
```

We get better val accuracy with augmentation, hence we use data augmentation

3. Check for overfit:

   For low regularization=1e-6, high learning rate=0.1

   We can see data is over fit and we get accuracy=1

```
Weight_Decay=0;Learning_Rate=0.001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=600,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 593/600
15/15 [==============================] - 0s 26ms/step - loss: 0.0229 - accuracy: 0.9968 - val_loss: 1.1577 - val_accuracy: 0.5000
Epoch 594/600
15/15 [==============================] - 0s 25ms/step - loss: 0.0136 - accuracy: 1.0000 - val_loss: 1.1578 - val_accuracy: 0.5000
Epoch 595/600
15/15 [==============================] - 0s 25ms/step - loss: 0.0241 - accuracy: 0.9923 - val_loss: 1.1585 - val_accuracy: 0.5000
Epoch 596/600
15/15 [==============================] - 0s 27ms/step - loss: 0.0273 - accuracy: 0.9927 - val_loss: 1.1582 - val_accuracy: 0.5000
Epoch 597/600
15/15 [==============================] - 0s 25ms/step - loss: 0.0168 - accuracy: 0.9978 - val_loss: 1.1594 - val_accuracy: 0.5000
Epoch 598/600
15/15 [==============================] - 0s 26ms/step - loss: 0.0242 - accuracy: 0.9915 - val_loss: 1.1588 - val_accuracy: 0.5000
Epoch 599/600
15/15 [==============================] - 0s 26ms/step - loss: 0.0199 - accuracy: 0.9936 - val_loss: 1.1603 - val_accuracy: 0.5000
Epoch 600/600
15/15 [==============================] - 0s 25ms/step - loss: 0.0271 - accuracy: 0.9906 - val_loss: 1.1632 - val_accuracy: 0.5000
```

   Over fit with accuracy=1

4. Sanity Check:

   Keep constant learning rate=0.001 and increase regularization

   (i) Regularization= 0 (no regularization)

```
Weight_Decay=0;Learning_Rate=0.001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 23ms/step - loss: 0.5936 - accuracy: 0.6774 - val_loss: 0.4214 - val_accuracy: 0.7800
Epoch 2/10
15/15 [==============================] - 0s 24ms/step - loss: 0.3670 - accuracy: 0.8408 - val_loss: 0.2630 - val_accuracy: 0.9000
Epoch 3/10
15/15 [==============================] - 0s 26ms/step - loss: 0.3061 - accuracy: 0.8654 - val_loss: 0.2252 - val_accuracy: 0.9200
Epoch 4/10
15/15 [==============================] - 0s 26ms/step - loss: 0.2719 - accuracy: 0.8833 - val_loss: 0.2078 - val_accuracy: 0.9200
Epoch 5/10
15/15 [==============================] - 0s 25ms/step - loss: 0.2398 - accuracy: 0.9079 - val_loss: 0.1924 - val_accuracy: 0.9100
Epoch 6/10
15/15 [==============================] - 0s 26ms/step - loss: 0.2588 - accuracy: 0.8921 - val_loss: 0.1735 - val_accuracy: 0.9200
Epoch 7/10
15/15 [==============================] - 0s 27ms/step - loss: 0.2266 - accuracy: 0.9115 - val_loss: 0.1699 - val_accuracy: 0.9200
Epoch 8/10
15/15 [==============================] - 0s 25ms/step - loss: 0.2238 - accuracy: 0.9068 - val_loss: 0.1561 - val_accuracy: 0.9300
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: 0.2206 - accuracy: 0.9167 - val_loss: 0.1459 - val_accuracy: 0.9200
Epoch 10/10
15/15 [==============================] - 0s 25ms/step - loss: 0.1960 - accuracy: 0.9231 - val_loss: 0.1455 - val_accuracy: 0.9300
<keras.callbacks.callbacks.History at 0x7fabea0e5eb8>
```

(ii) Regularization=1000

```
Weight_Decay=1000;Learning_Rate=0.001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 23ms/step - loss: 20932.9559 - accuracy: 0.5182 - val_loss: 16475.1188 - val_accuracy: 0.4300
Epoch 2/10
15/15 [==============================] - 0s 24ms/step - loss: 4638.9565 - accuracy: 0.4968 - val_loss: 1739.5540 - val_accuracy: 0.4200
Epoch 3/10
15/15 [==============================] - 0s 25ms/step - loss: 910.6010 - accuracy: 0.4947 - val_loss: 1.4076 - val_accuracy: 0.5700
Epoch 4/10
15/15 [==============================] - 0s 26ms/step - loss: 173.3468 - accuracy: 0.4797 - val_loss: 69.2000 - val_accuracy: 0.5700
Epoch 5/10
15/15 [==============================] - 0s 26ms/step - loss: 38.3471 - accuracy: 0.4915 - val_loss: 30.3881 - val_accuracy: 0.5400
Epoch 6/10
15/15 [==============================] - 0s 25ms/step - loss: 9.1839 - accuracy: 0.5032 - val_loss: 3.9939 - val_accuracy: 0.5600
Epoch 7/10
15/15 [==============================] - 0s 24ms/step - loss: 2.4816 - accuracy: 0.5267 - val_loss: 0.9108 - val_accuracy: 0.5700
Epoch 8/10
15/15 [==============================] - 0s 25ms/step - loss: 1.1291 - accuracy: 0.5583 - val_loss: 0.8559 - val_accuracy: 0.5500
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: 0.8495 - accuracy: 0.6100 - val_loss: 0.7931 - val_accuracy: 0.6400
Epoch 10/10
15/15 [==============================] - 0s 26ms/step - loss: 0.7540 - accuracy: 0.6346 - val_loss: 0.7695 - val_accuracy: 0.6400
<keras.callbacks.callbacks.History at 0x7fabe93636d8>
```

Loss increases

(iii) Regularization=10000

```
Weight_Decay=10000;Learning_Rate=0.001
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 23ms/step - loss: inf - accuracy: 0.5000 - val_loss: inf - val_accuracy: 0.5000
Epoch 2/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5118 - val_loss: nan - val_accuracy: 0.5000
Epoch 3/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.4904 - val_loss: nan - val_accuracy: 0.5000
Epoch 4/10
15/15 [==============================] - 0s 26ms/step - loss: nan - accuracy: 0.4861 - val_loss: nan - val_accuracy: 0.5000
Epoch 5/10
15/15 [==============================] - 0s 26ms/step - loss: nan - accuracy: 0.5073 - val_loss: nan - val_accuracy: 0.5000
Epoch 6/10
15/15 [==============================] - 0s 26ms/step - loss: nan - accuracy: 0.5118 - val_loss: nan - val_accuracy: 0.5000
Epoch 7/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5043 - val_loss: nan - val_accuracy: 0.5000
Epoch 8/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.4989 - val_loss: nan - val_accuracy: 0.5000
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5021 - val_loss: nan - val_accuracy: 0.5000
Epoch 10/10
15/15 [==============================] - 0s 26ms/step - loss: nan - accuracy: 0.4896 - val_loss: nan - val_accuracy: 0.5000
<keras.callbacks.callbacks.History at 0x7fabe92b1fd0>
```

Loss explodes

5. Sanity Check 2:

For a constant Regularization=0.001 and change Learning

(i) Learning Rate=1e-6

```
Weight_Decay=0.001;Learning_Rate=1e-6
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 24ms/step - loss: 0.9673 - accuracy: 0.4754 - val_loss: 0.8710 - val_accuracy: 0.4800
Epoch 2/10
15/15 [==============================] - 0s 26ms/step - loss: 0.9665 - accuracy: 0.4872 - val_loss: 0.8676 - val_accuracy: 0.4800
Epoch 3/10
15/15 [==============================] - 0s 24ms/step - loss: 0.9881 - accuracy: 0.4850 - val_loss: 0.8638 - val_accuracy: 0.4900
Epoch 4/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9896 - accuracy: 0.5000 - val_loss: 0.8599 - val_accuracy: 0.4900
Epoch 5/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9956 - accuracy: 0.4833 - val_loss: 0.8561 - val_accuracy: 0.4900
Epoch 6/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9801 - accuracy: 0.4890 - val_loss: 0.8523 - val_accuracy: 0.4900
Epoch 7/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9642 - accuracy: 0.4947 - val_loss: 0.8487 - val_accuracy: 0.4900
Epoch 8/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9907 - accuracy: 0.4690 - val_loss: 0.8449 - val_accuracy: 0.4900
Epoch 9/10
15/15 [==============================] - 0s 26ms/step - loss: 0.8915 - accuracy: 0.5271 - val_loss: 0.8413 - val_accuracy: 0.5000
Epoch 10/10
15/15 [==============================] - 0s 25ms/step - loss: 0.9559 - accuracy: 0.5044 - val_loss: 0.8378 - val_accuracy: 0.5000
<keras.callbacks.callbacks.History at 0x7fabe91b8d68>
```

(ii) Learning Rate=1e-3

```
Weight_Decay=0.001;Learning_Rate=1e-3
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 23ms/step - loss: 0.7339 - accuracy: 0.6303 - val_loss: 0.4883 - val_accuracy: 0.7800
Epoch 2/10
15/15 [==============================] - 0s 23ms/step - loss: 0.4528 - accuracy: 0.8120 - val_loss: 0.3372 - val_accuracy: 0.8700
Epoch 3/10
15/15 [==============================] - 0s 25ms/step - loss: 0.3966 - accuracy: 0.8419 - val_loss: 0.3013 - val_accuracy: 0.8700
Epoch 4/10
15/15 [==============================] - 0s 26ms/step - loss: 0.3714 - accuracy: 0.8600 - val_loss: 0.2643 - val_accuracy: 0.8800
Epoch 5/10
15/15 [==============================] - 0s 25ms/step - loss: 0.3286 - accuracy: 0.8875 - val_loss: 0.2496 - val_accuracy: 0.8800
Epoch 6/10
15/15 [==============================] - 0s 26ms/step - loss: 0.3171 - accuracy: 0.8932 - val_loss: 0.2358 - val_accuracy: 0.9000
Epoch 7/10
15/15 [==============================] - 0s 26ms/step - loss: 0.3209 - accuracy: 0.8814 - val_loss: 0.2230 - val_accuracy: 0.9300
Epoch 8/10
15/15 [==============================] - 0s 26ms/step - loss: 0.2754 - accuracy: 0.9060 - val_loss: 0.2104 - val_accuracy: 0.9200
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: 0.2828 - accuracy: 0.9112 - val_loss: 0.2006 - val_accuracy: 0.9200
Epoch 10/10
15/15 [==============================] - 0s 25ms/step - loss: 0.2917 - accuracy: 0.8921 - val_loss: 0.1974 - val_accuracy: 0.9200
<keras.callbacks.callbacks.History at 0x7fabe9017748>
```

Accuracy increases

(iii) Learning Rate=1e1

```
Weight_Decay=0.001;Learning_Rate=1e1
model = define_model(Weight_Decay,Learning_Rate)
datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
datagen.fit(trainX)
model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

```
Epoch 1/10
15/15 [==============================] - 0s 23ms/step - loss: nan - accuracy: 0.5031 - val_loss: nan - val_accuracy: 0.5000
Epoch 2/10
15/15 [==============================] - 0s 23ms/step - loss: nan - accuracy: 0.4901 - val_loss: nan - val_accuracy: 0.5000
Epoch 3/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5096 - val_loss: nan - val_accuracy: 0.5000
Epoch 4/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.4915 - val_loss: nan - val_accuracy: 0.5000
Epoch 5/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5128 - val_loss: nan - val_accuracy: 0.5000
Epoch 6/10
15/15 [==============================] - 0s 26ms/step - loss: nan - accuracy: 0.4904 - val_loss: nan - val_accuracy: 0.5000
Epoch 7/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.4990 - val_loss: nan - val_accuracy: 0.5000
Epoch 8/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5044 - val_loss: nan - val_accuracy: 0.5000
Epoch 9/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.4989 - val_loss: nan - val_accuracy: 0.5000
Epoch 10/10
15/15 [==============================] - 0s 25ms/step - loss: nan - accuracy: 0.5010 - val_loss: nan - val_accuracy: 0.5000
<keras.callbacks.callbacks.History at 0x7fabe8dcaeb8>
```

Loss explodes => high learning rate

6. Hyper Parameter tuning:
   - Based on previous training values on the model we take a range of values for Learning Rate and Regularization and train the model.
   - We find the parameters for which loss is minimum/accuracy is maximum.
   - Initially larger range is taken. This is called coarse search.
   - From value obtained we shrink the range of search. This is called fine search.

Code for hyperparameter tuning:

```
max_count=10
for count in range(max_count):
  REG=10**random.uniform(-6,0)
  LR=10**random.uniform(-6,-1)
  model = define_model(REG,LR)
  datagen = ImageDataGenerator(rotation_range=15,width_shift_range=0.1,height_shift_range=0.1,horizontal_flip=True,)
  datagen.fit(trainX)
  model.fit_generator(datagen.flow(trainX,trainY, batch_size=64),
                    steps_per_epoch=trainX.shape[0] // batch_size,epochs=10,
                    verbose=1,validation_data=(testX,testY))
```

Result:

```
Regularization= 0.0004976926448961536 Learning Rate= 0.009957543610125987
Epoch 1/5
15/15 [==============================] - 0s 23ms/step - loss: 0.4764 - accuracy:
0.7874 - val_loss: 0.2510 - val_accuracy: 0.8900
Epoch 2/5
15/15 [==============================] - 0s 24ms/step - loss: 0.2776 - accuracy:
0.8942 - val_loss: 0.1733 - val_accuracy: 0.9200
Epoch 3/5
15/15 [==============================] - 0s 25ms/step - loss: 0.2323 - accuracy:
0.9295 - val_loss: 0.1078 - val_accuracy: 0.9900
Epoch 4/5
15/15 [==============================] - 0s 26ms/step - loss: 0.1888 - accuracy:
0.9380 - val_loss: 0.0955 - val_accuracy: 0.9900
Epoch 5/5
15/15 [==============================] - 0s 26ms/step - loss: 0.1825 - accuracy:
0.9466 - val_loss: 0.0943 - val_accuracy: 0.9700
Regularization= 0.0009819451632456027 Learning Rate= 1.4029698359848643e-05
Epoch 1/5
15/15 [==============================] - 0s 24ms/step - loss: 1.6474 - accuracy:
0.3429 - val_loss: 1.4554 - val_accuracy: 0.3900
Epoch 2/5
```

```
15/15 [==============================] - 0s 24ms/step - loss: 1.4723 - accuracy:
0.3643 - val_loss: 1.3322 - val_accuracy: 0.3800
Epoch 3/5
15/15 [==============================] - 0s 26ms/step - loss: 1.3539 - accuracy:
0.3558 - val_loss: 1.2107 - val_accuracy: 0.3700
Epoch 4/5
15/15 [==============================] - 0s 25ms/step - loss: 1.2122 - accuracy:
0.3479 - val_loss: 1.1063 - val_accuracy: 0.3800
Epoch 5/5
15/15 [==============================] - 0s 26ms/step - loss: 1.0653 - accuracy:
0.3814 - val_loss: 1.0225 - val_accuracy: 0.3900
Regularization= 0.0006936704337253108 Learning Rate= 0.023647209703063228
Epoch 1/5
15/15 [==============================] - 0s 24ms/step - loss: 0.6479 - accuracy:
0.7917 - val_loss: 0.3092 - val_accuracy: 0.9000
Epoch 2/5
15/15 [==============================] - 0s 23ms/step - loss: 0.3771 - accuracy:
0.8921 - val_loss: 0.2892 - val_accuracy: 0.9200
Epoch 3/5
15/15 [==============================] - 0s 24ms/step - loss: 0.2553 - accuracy:
0.9092 - val_loss: 0.1563 - val_accuracy: 0.9500
Epoch 4/5
15/15 [==============================] - 0s 26ms/step - loss: 0.2267 - accuracy:
0.9327 - val_loss: 0.1301 - val_accuracy: 0.9500
Epoch 5/5
15/15 [==============================] - 0s 25ms/step - loss: 0.1872 - accuracy:
0.9434 - val_loss: 0.1149 - val_accuracy: 0.9800
Regularization= 0.5761702698763669 Learning Rate= 0.07188863407309697
Epoch 1/5
15/15 [==============================] - 0s 25ms/step - loss: 8.0028 - accuracy:
0.7382 - val_loss: 1.4067 - val_accuracy: 0.9000
Epoch 2/5
15/15 [==============================] - 0s 23ms/step - loss: 2.0169 - accuracy:
0.7340 - val_loss: 2.0138 - val_accuracy: 0.8200
Epoch 3/5
15/15 [==============================] - 0s 26ms/step - loss: 1.9916 - accuracy:
0.7333 - val_loss: 1.1940 - val_accuracy: 0.8500
Epoch 4/5
15/15 [==============================] - 0s 24ms/step - loss: 1.0118 - accuracy:
0.8004 - val_loss: 0.6489 - val_accuracy: 0.9200
Epoch 5/5
15/15 [==============================] - 0s 27ms/step - loss: 0.7805 - accuracy:
0.8184 - val_loss: 0.7781 - val_accuracy: 0.8100
Regularization= 0.02080252799814441 Learning Rate= 9.510638091984489e-06
Epoch 1/5
15/15 [==============================] - 0s 24ms/step - loss: 1.8119 - accuracy:
0.5139 - val_loss: 1.8229 - val_accuracy: 0.4900
Epoch 2/5
15/15 [==============================] - 0s 24ms/step - loss: 1.7187 - accuracy:
0.5235 - val_loss: 1.7760 - val_accuracy: 0.4900
Epoch 3/5
15/15 [==============================] - 0s 26ms/step - loss: 1.6920 - accuracy:
0.5513 - val_loss: 1.7336 - val_accuracy: 0.5000
Epoch 4/5
15/15 [==============================] - 0s 24ms/step - loss: 1.6542 - accuracy:
0.5577 - val_loss: 1.6994 - val_accuracy: 0.5100
Epoch 5/5
15/15 [==============================] - 0s 26ms/step - loss: 1.6051 - accuracy:
0.5990 - val_loss: 1.6743 - val_accuracy: 0.5400
Regularization= 0.0007701708223108339 Learning Rate= 0.00041135436642474434
Epoch 1/5
15/15 [==============================] - 0s 22ms/step - loss: 0.6206 - accuracy:
0.6891 - val_loss: 0.5736 - val_accuracy: 0.7100
```

```
Epoch 2/5
15/15 [==============================] - 0s 25ms/step - loss: 0.5084 - accuracy:
0.7511 - val_loss: 0.4445 - val_accuracy: 0.8300
Epoch 3/5
15/15 [==============================] - 0s 25ms/step - loss: 0.4353 - accuracy:
0.8152 - val_loss: 0.3705 - val_accuracy: 0.8600
Epoch 4/5
15/15 [==============================] - 0s 26ms/step - loss: 0.4198 - accuracy:
0.8387 - val_loss: 0.3228 - val_accuracy: 0.9100
Epoch 5/5
15/15 [==============================] - 0s 25ms/step - loss: 0.3573 - accuracy:
0.8719 - val_loss: 0.3037 - val_accuracy: 0.9100
Regularization= 0.02597090269567742 Learning Rate= 0.05210002876562015
Epoch 1/5
15/15 [==============================] - 0s 24ms/step - loss: 2.1917 - accuracy:
0.7318 - val_loss: 1.3807 - val_accuracy: 0.9000
Epoch 2/5
15/15 [==============================] - 0s 22ms/step - loss: 1.2663 - accuracy:
0.8782 - val_loss: 0.8126 - val_accuracy: 0.9600
Epoch 3/5
15/15 [==============================] - 0s 26ms/step - loss: 1.3691 - accuracy:
0.8397 - val_loss: 1.6568 - val_accuracy: 0.8700
Epoch 4/5
15/15 [==============================] - 0s 26ms/step - loss: 0.9865 - accuracy:
0.8761 - val_loss: 0.8959 - val_accuracy: 0.8900
Epoch 5/5
15/15 [==============================] - 0s 25ms/step - loss: 0.7187 - accuracy:
0.8953 - val_loss: 0.3821 - val_accuracy: 0.9700
Regularization= 0.0006131708565397743 Learning Rate= 3.1631100464508677e-05
Epoch 1/5
15/15 [==============================] - 0s 24ms/step - loss: 0.7044 - accuracy:
0.6165 - val_loss: 0.6157 - val_accuracy: 0.6600
Epoch 2/5
15/15 [==============================] - 0s 24ms/step - loss: 0.6720 - accuracy:
0.6448 - val_loss: 0.5925 - val_accuracy: 0.6700
Epoch 3/5
15/15 [==============================] - 0s 25ms/step - loss: 0.6698 - accuracy:
0.6393 - val_loss: 0.5705 - val_accuracy: 0.7100
Epoch 4/5
15/15 [==============================] - 0s 23ms/step - loss: 0.6563 - accuracy:
0.6571 - val_loss: 0.5521 - val_accuracy: 0.7100
Epoch 5/5
15/15 [==============================] - 0s 25ms/step - loss: 0.6412 - accuracy:
0.6560 - val_loss: 0.5358 - val_accuracy: 0.7200
Regularization= 0.0050510393411975696 Learning Rate= 1.172767314994243e-06
Epoch 1/5
15/15 [==============================] - 0s 22ms/step - loss: 1.2358 - accuracy:
0.4904 - val_loss: 1.4104 - val_accuracy: 0.4700
Epoch 2/5
15/15 [==============================] - 0s 24ms/step - loss: 1.2703 - accuracy:
0.4669 - val_loss: 1.4060 - val_accuracy: 0.4700
Epoch 3/5
15/15 [==============================] - 0s 27ms/step - loss: 1.2740 - accuracy:
0.4833 - val_loss: 1.4013 - val_accuracy: 0.4700
Epoch 4/5
15/15 [==============================] - 0s 26ms/step - loss: 1.2798 - accuracy:
0.4562 - val_loss: 1.3963 - val_accuracy: 0.4700
Epoch 5/5
15/15 [==============================] - 0s 26ms/step - loss: 1.2185 - accuracy:
0.4850 - val_loss: 1.3915 - val_accuracy: 0.4700
Regularization= 0.00016419077079051593 Learning Rate= 0.009751198396703617
Epoch 1/5
```

```
15/15 [==============================] - 0s 23ms/step - loss: 0.4917 - accuracy: 0.8056 - val_loss: 0.2517 - val_accuracy: 0.8900
Epoch 2/5
15/15 [==============================] - 0s 25ms/step - loss: 0.2479 - accuracy: 0.9021 - val_loss: 0.1470 - val_accuracy: 0.9600
Epoch 3/5
15/15 [==============================] - 0s 25ms/step - loss: 0.2363 - accuracy: 0.9090 - val_loss: 0.1053 - val_accuracy: 0.9600
Epoch 4/5
15/15 [==============================] - 0s 25ms/step - loss: 0.1925 - accuracy: 0.9250 - val_loss: 0.0923 - val_accuracy: 0.9600
Epoch 5/5
15/15 [==============================] - 0s 25ms/step - loss: 0.1733 - accuracy: 0.9408 - val_loss: 0.0999 - val_accuracy: 0.9600
```

We find from this list that there is minimum loss, maximum accuracy for `Regularization= 0.02597090269567742 Learning Rate= 0.05210002876562015`

Testing for these values of Regularization and Learning Rate, we get following result:

```
_, acc = model.evaluate(testX, testY, verbose=0)
print('> %.3f' % (acc * 100.0))
# learning curves
summarize_diagnostics(history)
```

```
> 98.000
```

Additional Exercise1: The babysitting process was initially done using CIFAR 10 dataset consisting of 10 classes of images including automobile, bird, cat etc. A small portion of the total dataset of 10000 images was used for this process. The dataset is obtained using Keras library as follows. Remaining processes are same as mentioned for face detection.

```
from keras.datasets import cifar10
# load dataset
(trainX, trainY), (testX, testY) = cifar10.load_data()
# one hot encode target values
print(testX[0])
print(testX.shape,type(testX))
trainY = to_categorical(trainY)
testY = to_categorical(testY)
```

Additional Exercise2: Instead of a simple fully connected network, a convolutional neural network was used for the same process to increase accuracy. This model consists of 2Dconvolutional layers followed by max pooling. Structure of such a network is given below.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W), input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W)))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W)))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', kernel_regularizer=l2(W)))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten(input_shape=(32,32,3)))
model.add(Dense(50, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(W)))
model.add(Dense(10, activation='softmax'))
```