# Cognitive Computing: Keras CNN Model Traing Assignment

## Aim:

Perform model training and compute Accuracy for randomly collected subset (12K) of dataset (60K) and test it against different combinations of Callbacks. Observe its effect on accuracy

## Assumptions/Pre-requisites:

Construct your model against 60K data and make it more accurate using specific factor (Callbacks in this case). Design your best model with different parameters to test it against random images data.

### Image Summary Per Class: Randomly taken images count per class

```
airplane 1180
automobile 1135
bird 1138
cat 1227
deer 1129
dog 1269
frog 1220
horse 1242
ship 1227
truck 1233
```

```
# Get random images
seed_val = 10
test_data = np.random.randint(0, 60000, size=12000)
x_train = full_data_x[test_data]
y_train = full_data_y[test_data]
#print(len(data_x))
print('y train data: ', y_train.shape[0])

from scipy.stats import itemfreq
count= itemfreq(data_y)
for i in range (0,len(count)):
    print(labels['label_names'][i],count[i][1])
```

```
y train data:  12000
airplane 1180
automobile 1135
bird 1138
cat 1227
deer 1129
dog 1269
frog 1220
horse 1242
ship 1227
truck 1233
```

**Experiment 1: Add ModelCheckpoint and EarlyStopping Callbacks**

Accuracy: 0.66

```
# checkpoint
outputFolder = './output-callback'
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)
filepath=outputFolder+"/weights-{epoch:02d}-{val_acc:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, \
                             save_best_only=False, save_weights_only=True, \
                             mode='auto', period=10)
#callbacks_list = [checkpoint]


earlystop = EarlyStopping(monitor='val_acc', min_delta=0.0001, patience=5, \
                          verbose=1, mode='auto')

# Custom callback for test data per epoch
custom_callback = LossAccPerEpochCallback((x_test, y_test))

callbacks_list = [earlystop, checkpoint]
```

Result:

```
Using real-time data augmentation.
Epoch 1/5
1562/1562 [==============================] - 71s - loss: 1.6335 - acc: 0.4058 - val_loss: 1.3302 - val_acc: 0.5378
Epoch 2/5
1562/1562 [==============================] - 71s - loss: 1.2397 - acc: 0.5624 - val_loss: 1.0113 - val_acc: 0.6463
Epoch 3/5
1562/1562 [==============================] - 71s - loss: 1.1289 - acc: 0.6060 - val_loss: 0.9032 - val_acc: 0.6851
Epoch 4/5
1562/1562 [==============================] - 72s - loss: 1.1003 - acc: 0.6248 - val_loss: 0.8902 - val_acc: 0.6880
Epoch 5/5
1562/1562 [==============================] - 71s - loss: 1.1033 - acc: 0.6291 - val_loss: 0.9164 - val_acc: 0.6873
```

```
# Evaluate model with test data set and share sample prediction results
evaluation = model.evaluate_generator(datagen.flow(x_test, y_test,
                                                   batch_size=batch_size,
                                                   shuffle=False),
                                      steps=x_test.shape[0] // batch_size,
                                      workers=4)
print('Model Accuracy = %.2f' % (evaluation[1]))
```

```
Model Accuracy = 0.66
```

**Observation**: Using combination sometimes increased the accuracy.

**Experiment 2: Add custom Callback**

Accuracy: 0.10

```
# checkpoint
outputFolder = './output-callback'
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)
filepath=outputFolder+"/weights-{epoch:02d}-{val_acc:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, \
                             save_best_only=False, save_weights_only=True, \
                             mode='auto', period=10)
#callbacks_list = [checkpoint]


earlystop = EarlyStopping(monitor='val_acc', min_delta=0.0001, patience=5, \
                          verbose=1, mode='auto')

# Custom callback for test data per epoch
custom_callback = LossAccPerEpochCallback((x_test, y_test))

callbacks_list = [custom_callback]
```

Result:

```
Using real-time data augmentation.
Epoch 1/5
1562/1562 [==============================] - 73s - loss: 2.3028 - acc: 0.0995 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 2/5
1562/1562 [==============================] - 71s - loss: 2.3028 - acc: 0.0985 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 3/5
1562/1562 [==============================] - 71s - loss: 2.3028 - acc: 0.0981 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 4/5
1562/1562 [==============================] - 71s - loss: 2.3028 - acc: 0.0989 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 5/5
1562/1562 [==============================] - 71s - loss: 2.3028 - acc: 0.0989 - val_loss: 2.3026 - val_acc: 0.1000
```

```
# Evaluate model with test data set and share sample prediction results
evaluation = model.evaluate_generator(datagen.flow(x_test, y_test,
                                                   batch_size=batch_size,
                                                   shuffle=False),
                                      steps=x_test.shape[0] // batch_size,
                                      workers=4)
print('Model Accuracy = %.2f' % (evaluation[1]))
```

```
Model Accuracy = 0.10
```

**Experiment 3: Add only ModelCheckpoint Callback**

Accuracy: 0.60

```
# Add Calbacks
# checkpoint
outputFolder = './output-callback'
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)
filepath=outputFolder+"/weights-{epoch:02d}-{val_acc:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, \
                             save_best_only=False, save_weights_only=True, \
                             mode='auto', period=10)
#callbacks_list = [checkpoint]

earlystop = EarlyStopping(monitor='val_acc', min_delta=0.0001, patience=5, \
                          verbose=1, mode='auto')

# Custom callback for test data per epoch
custom_callback = LossAccPerEpochCallback((x_train, y_train))

callbacks_list = [checkpoint]
```

Result:

```
Using real-time data augmentation.
Epoch 1/5
375/375 [==============================] - 19s - loss: 1.2203 - acc: 0.5671 - val_loss: 0.9812 - val_acc: 0.6495
Epoch 2/5
375/375 [==============================] - 19s - loss: 1.1877 - acc: 0.5791 - val_loss: 0.9975 - val_acc: 0.6586
Epoch 3/5
375/375 [==============================] - 19s - loss: 1.1382 - acc: 0.6028 - val_loss: 0.9344 - val_acc: 0.6695
Epoch 4/5
375/375 [==============================] - 19s - loss: 1.1272 - acc: 0.6091 - val_loss: 0.9408 - val_acc: 0.6679
Epoch 5/5
374/375 [============================>.] - ETA: 0s - loss: 1.1095 - acc: 0.6177Epoch 00004: saving model to ./output-callback/weights-04-0.67.hdf5
375/375 [==============================] - 20s - loss: 1.1087 - acc: 0.6176 - val_loss: 0.9607 - val_acc: 0.6663
```

```
# Evaluate model with test data set and share sample prediction results
evaluation = model.evaluate_generator(datagen.flow(x_train, y_train,
                                                   batch_size=batch_size,
                                                   shuffle=False),
                                      steps=x_train.shape[0] // batch_size,
                                      workers=4)
print('Model Accuracy = %.2f' % (evaluation[1]))
```

```
Model Accuracy = 0.63
```

**Observation:** Adding ModelCheckpoint Callback helped model to train itself using the history saved after every epoch. Saving best accuracy from running epochs helped model to maintain the accuracy level for further epochs

**Experiment 4: Add only EarlyStopping Callback**

Accuracy: 0.56

```
# Add Calbacks
# checkpoint
outputFolder = './output-callback'
if not os.path.exists(outputFolder):
    os.makedirs(outputFolder)
filepath=outputFolder+"/weights-{epoch:02d}-{val_acc:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1, \
                             save_best_only=False, save_weights_only=True, \
                             mode='auto', period=10)
#callbacks_list = [checkpoint]

earlystop = EarlyStopping(monitor='val_acc', min_delta=0.0001, patience=5, \
                          verbose=1, mode='auto')

# Custom callback for test data per epoch
custom_callback = LossAccPerEpochCallback((x_train, y_train))

callbacks_list = [earlystop]
```

Result:

```
Using real-time data augmentation.
Epoch 1/5
375/375 [==============================] - 20s - loss: 1.9329 - acc: 0.2838 - val_loss: 1.8951 - val_acc: 0.3458
Epoch 2/5
375/375 [==============================] - 20s - loss: 1.6093 - acc: 0.4153 - val_loss: 1.5469 - val_acc: 0.4561
Epoch 3/5
375/375 [==============================] - 20s - loss: 1.4660 - acc: 0.4730 - val_loss: 1.2511 - val_acc: 0.5541
Epoch 4/5
375/375 [==============================] - 20s - loss: 1.3770 - acc: 0.5023 - val_loss: 1.2032 - val_acc: 0.5741
Epoch 5/5
375/375 [==============================] - 20s - loss: 1.2946 - acc: 0.5372 - val_loss: 1.1429 - val_acc: 0.5946
```

```
# Evaluate model with test data set and share sample prediction results
evaluation = model.evaluate_generator(datagen.flow(x_train, y_train,
                                                   batch_size=batch_size,
                                                   shuffle=False),
                                      steps=x_train.shape[0] // batch_size,
                                      workers=4)
print('Model Accuracy = %.2f' % (evaluation[1]))
```

```
Model Accuracy = 0.56
```