EECE-8740:Neural Networks
Assignment 2

Felix Havugimana
U Number:U00743094

September 28, 2021

1 Problem 1: CNN model for CIFAR-10 Classification Task

1.1 Introduction

In this problem, we are asked to design a Deep Convolutional Neural Network (CNN) to perform a classification task on Cifar10 dataset. Cifar10 dataset has 10 classes of 3D images namely 'airplance', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck'. Images have shape of 32 X 32 X 3. The task is to design a CNN network depicted in Fig.

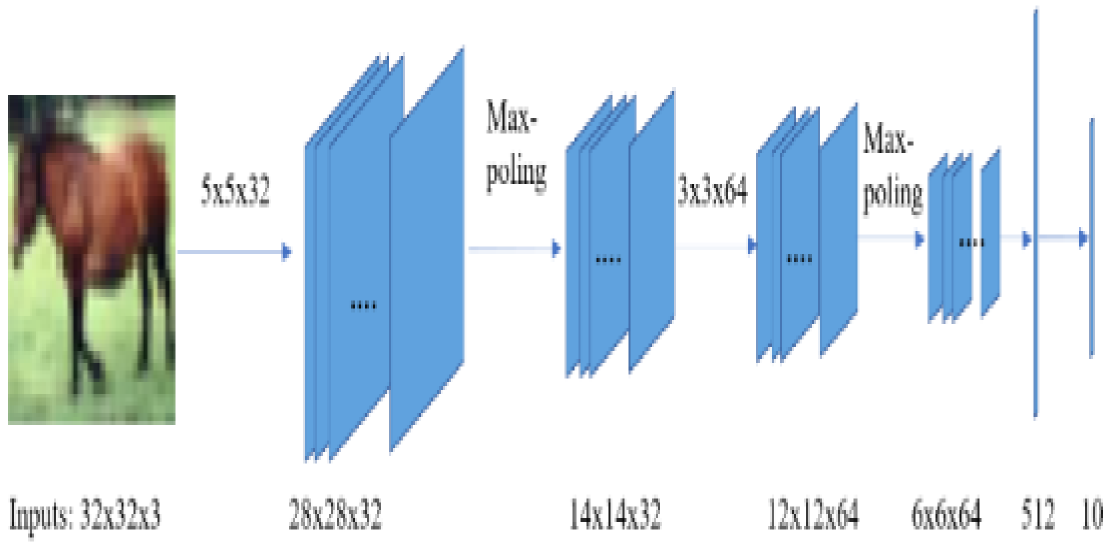


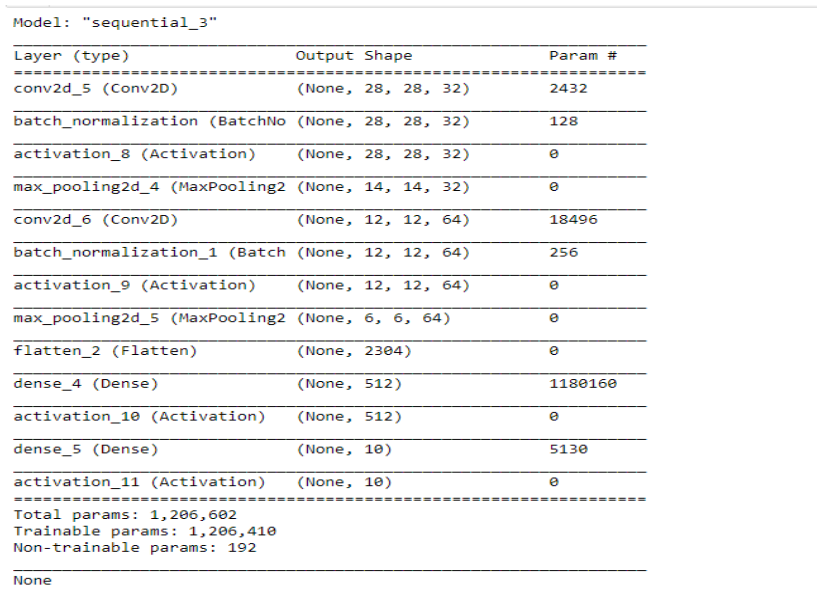
Figure 1: Problem1 Model architecture

1.2 Methodology

I designed the architecture following the problem description above. The CNN model was trained on CIFAR-10 dataset which contains 60000 RGB images with size of 32 by 32 by 3. I used 50000 images and 10000 images for training and testing respectively. After loading images, I scaled images from the range 0-255 to the range 0-1. Moreover, CIFAR-10 images belong to 10 classes, therefore I used hot encoding to encode labels for classification. See section. 5 for details about model architecture.

1.3 Model architecture

My CNN is composed by two 2D convolution layers for feature selection, one full connected layers, and softmax layer for classification. Each convolution layer is followed by max pooling and batch



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 28, 28, 32)        2432      
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 32)        128       
_________________________________________________________________
activation_8 (Activation)    (None, 28, 28, 32)        0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 64)        18496     
_________________________________________________________________
batch_normalization_1 (Batch (None, 12, 12, 64)        256       
_________________________________________________________________
activation_9 (Activation)    (None, 12, 12, 64)        0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_4 (Dense)              (None, 512)               1180160   
_________________________________________________________________
activation_10 (Activation)   (None, 512)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 10)                5130      
_________________________________________________________________
activation_11 (Activation)   (None, 10)                0         
=================================================================
Total params: 1,206,602
Trainable params: 1,206,410
Non-trainable params: 192
_________________________________________________________________
None
```

Figure 2: Model summary

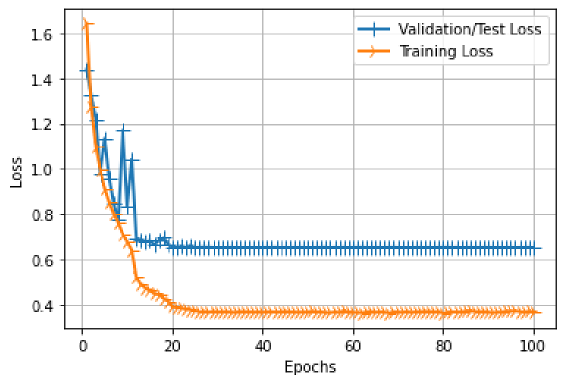
normalization layers with Relu as activation function. To control overfitting, I added drop out layers with dropout rates of 0.20, 0.15, and 0.35 for the fist conv layer, second conv layer, and full connected layer respectively. see Fig. 1 for model requirement and Fig.2 for the model summary.

1.4 Experimental Results

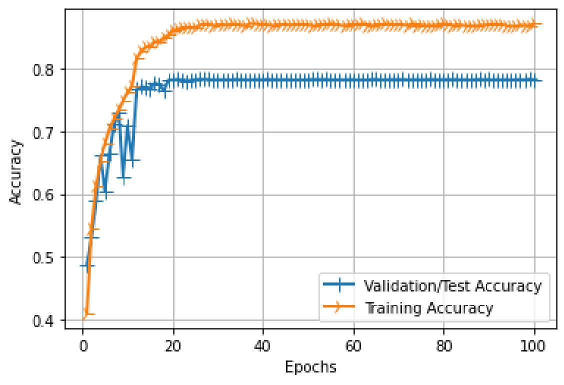
1.4.1 Training and testing logs

The model described above was trained for 100 epochs with batch size of 32 images. I trained the model to minimize cross-categorical entropy loss using Stochastic Gradient Descent (SGD) as model optimizer. For good and faster convergence, I used adaptive learning late starting at 0.01 with momentum of 0.9. I used Keras call backs namely ModelCheckPoint and ReduceLROnPlateau, where the former helps to save only the best model during the training and later helps to automatically lower the learning by a given factor when model did not improve after a certain number of epochs.

In fig. **??** plotted both loss and accuracy curves which shows how loss and accuracy values changed after each epoch during training.



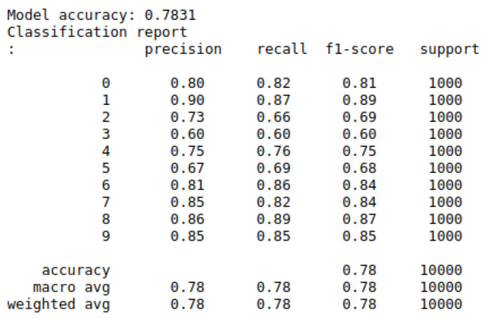
(a) Loss curvel



(b) Accuracy curve

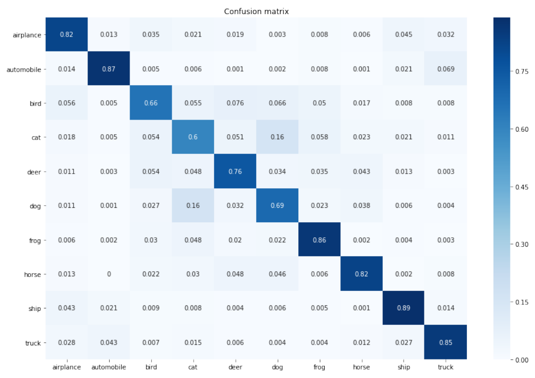
Figure 3: (a) Loss curve and (b) accuracy for CNN model in Fig.2 trained on CIFAR-10 dataset classfication task

Fig. 4 shows classification report and confusion matrix for our CNN model. From classification report, we can see that our model reached classification accuracy, Recall, and precision of 78%,78%,and 78% respectively. The confusion matrix helps us see the performance on the model on each class, by indicating the percentage of correct classifications and how much the model confused one class from the other.



```
Model accuracy: 0.7831
Classification report
:              precision    recall  f1-score   support

           0       0.80      0.82      0.81      1000
           1       0.90      0.87      0.89      1000
           2       0.73      0.66      0.69      1000
           3       0.60      0.60      0.60      1000
           4       0.75      0.76      0.75      1000
           5       0.67      0.69      0.68      1000
           6       0.81      0.86      0.84      1000
           7       0.85      0.82      0.84      1000
           8       0.86      0.89      0.87      1000
           9       0.85      0.85      0.85      1000

    accuracy                           0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.78      0.78      0.78     10000
```

(a) Classification report



(b) Confusion matrix

Figure 4: (a) Classification report and (b) Confusion matrix in Fig.2 trained on CIFAR-10 dataset classification task

1.4.2 Discussion

The above results show the performance of the CNN model on CIFAR-10 dataset. From the accuracy and loss curve we can see a smooth training with small gap between train and validation curves. Moreover, from confusion matrix we can that the model did a descent job in classifying some classes such as automobile, airplane, frog and ship and it performed poorly in other classes such as cat and bird. There is a still a room for improving the model to achieve much better performance.

1.5 Conclusion

To sum up, in this assignment I designed a deep CNN model for image classification on CIFAR-10 dataset. The model was trained for 100 epochs with Relu activation function, adam optimizer, and categorical cross entropy. By training the model over 100 epochs, I reached classification accuracy of 78%. Even though the model performance may not be optimal, it showed consistence in terms of accuracy, recall, precision, and F-1 measures which proves its reliability.

2 Problem 2: Scene Understanding

2.1 Introduction

In this part , I was required to design a deep CNN model for scene understanding classification task. Contrary to Part.1, I will explore more advanced techniques such as data augmentation, momentum, weight regularization, and different optimers to achieve better classification performance.

2.2 Methodology

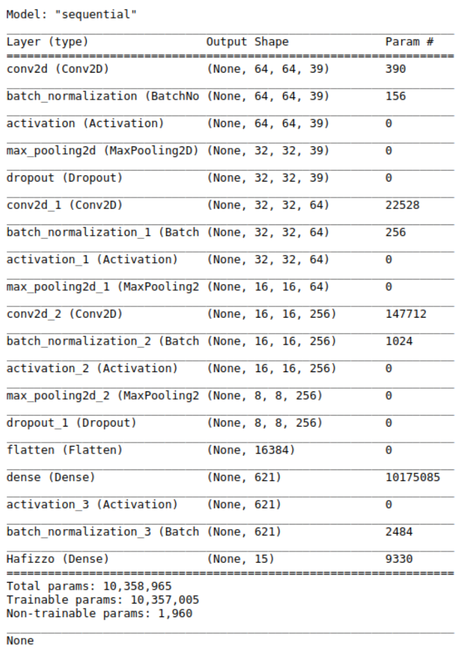
15 scene dataset contains gray scale images from 15 classes namely bedroom, building store , industry, living room, kitchen , highway,coast, office, mountain, inside city , forest, suburb, street, and open country. The task is to design a deep CNN model to classify the above scene. The designed model is consists of convolution, pooling, normalization, drop pout, and fully connected layers. I used ELU as activation function, and I tested three optimizes: SGD, Adam, and RMSProp.

In section. 2.4, I report classification performance report, training and validation curves, and confusion matrix for each optimizer.

2.3 Deep Learning Architecture

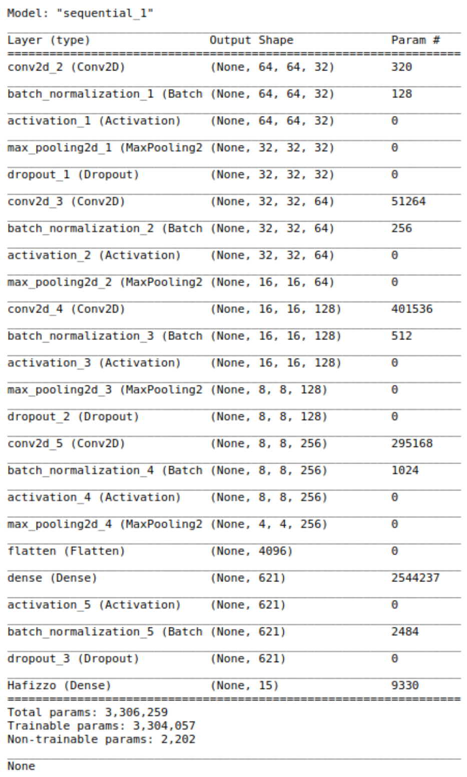
I designed a shallow sequential CNN model which consists of two main networks. The first network is a feature extractor network composed of conv-max pooling layers and its main function is to extract spatial features from the input image. The classification network is composed of a fully connected layer followed by a softmax layer for image classification.

For Adam and RMSpPROP optimzers, the model has 3 conv-pooling layers, one fully connected layer and one softmax layer. Each conv-max layer is followed by a drop out layer withe dropout rate of 0.35, and batch normalization layer. I also used drop out layer after after full connected layer for RMSPROP. After trying different model configurations, I designed a different model for SGd optimizer since the above model was performing poorly with SGD. The SGD model is much deeper than the others. Model architectures are depicted in Fig. 5.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 39)        390
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 64, 39)        156
_________________________________________________________________
activation (Activation)      (None, 64, 64, 39)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 39)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 39)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        22528
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 256)       147712
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 621)               10175085
_________________________________________________________________
activation_3 (Activation)    (None, 621)               0
_________________________________________________________________
batch_normalization_3 (Batch (None, 621)               2484
_________________________________________________________________
Hafizzo (Dense)              (None, 15)                9330
=================================================================
Total params: 10,358,965
Trainable params: 10,357,005
Non-trainable params: 1,960
_________________________________________________________________
None
```

(a) CNN arcitecture for RMSPROP and Adam



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 64, 64, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 32)        128
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 64)        51264
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
activation_2 (Activation)    (None, 32, 32, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       401536
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 256)         0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 621)               2544237
_________________________________________________________________
activation_5 (Activation)    (None, 621)               0
_________________________________________________________________
batch_normalization_5 (Batch (None, 621)               2484
_________________________________________________________________
dropout_3 (Dropout)          (None, 621)               0
_________________________________________________________________
Hafizzo (Dense)              (None, 15)                9330
=================================================================
Total params: 3,306,259
Trainable params: 3,304,057
Non-trainable params: 2,202
_________________________________________________________________
None
```

(b) CNN arcitecture for SGD

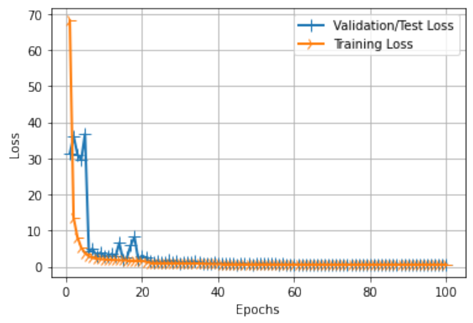
Figure 5: CNN model architectures for scene understanding task.

2.4 Experimental Results

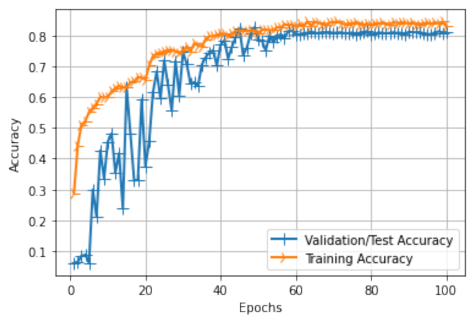
2.4.1 Training and Testing Loss

The models described above was trained for 100 epochs with batch size of 32 images. I trained the model to minimize cross-categorical entropy loss using Stochastic Gradient Descent (SGD), RMSPROP, and Adam optimizer. For good and faster convergence, I used adaptive learning late starting at 0.0001, 0.001, 0.01 for RMSPROP, SGD, and Adam respectively, with momentum of 0.9 where it's applicable. I used Keras call backs namely ModelCheckPoint and ReduceLROnPlateau, where the former helps to save only the best model during the training and later helps to automatically lower the learning by a given factor when model did not improve after a certain number of epochs.

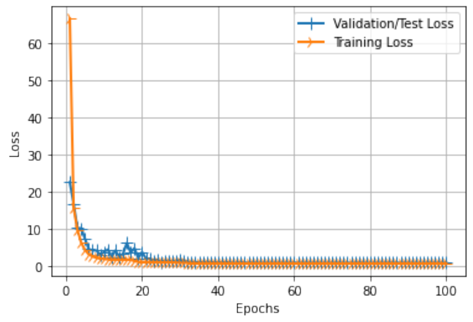
Fig.**??** shows training and validation curves (loss and accuracy) for each optimizer.
Fig. 7 shows classification report and confusion matrices for our optimizer.



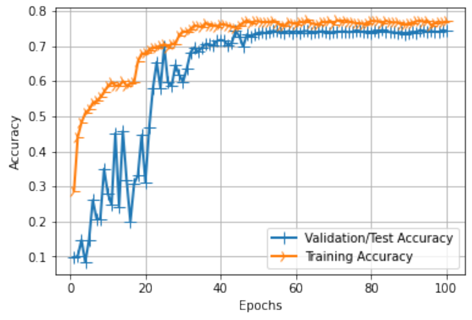
(a) Loss curve for RMSPROP



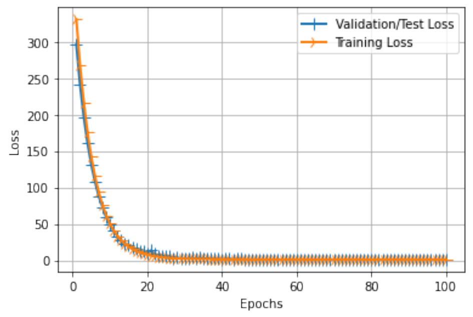
(b) Accuracy curve for RMSPROP



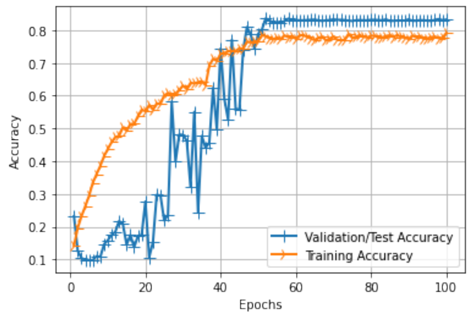
(c) Loss curve for Adam



(d) Accuracy curve for Adam



(e) Loss curve for SGD



(f) Accuracy curve for SGD

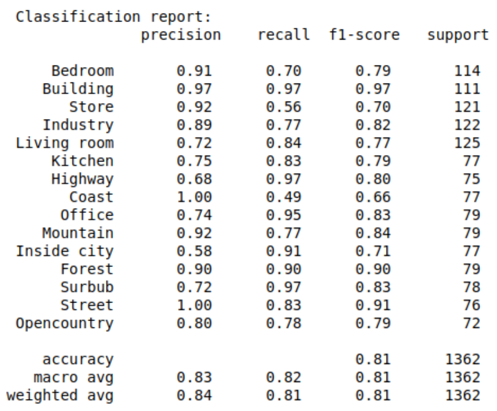
Figure 6: Training and Loss curves for CNN model trained with different optimizers {a,b}, {c,d } , {e,f}, Loss and Accuracy curve for RMSPROP, Adam, and SGD respectively

2.4.2 Discussion and Comparison

From Fig. 6 we can see that all models have very good loss and accuracy curves as the models is trained for 100 epochs. The curves show rough and flactuous training for the 40 epochs but attained a smooth training afterwards reaching a stable point with a infinitesimal gap between training and validation curves.

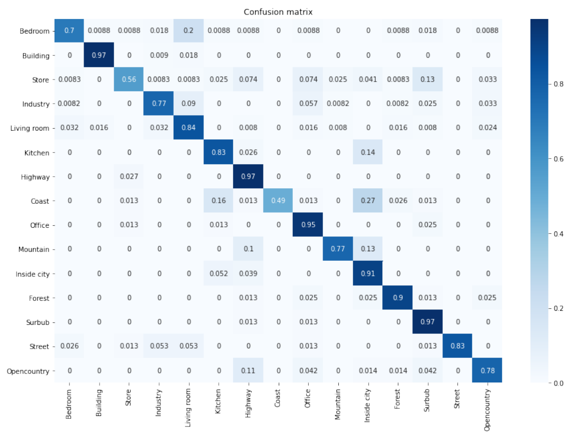
In terms of classification performance, all models reached a very reliable and stable classification performance on scene dataset with the highest accuracy of 83. Comparing performance of the three optimizers, SGD outperfomed other optimizers with accuracy of 83% where as Adan gave the least performance with accuracy of 74%.

Computationally all model took almost the same amount of time during training with average time=311sec on RTX 2080Ti GPU.

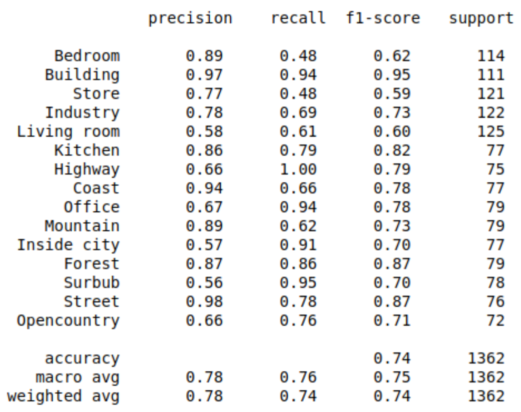


```
Classification report:
              precision    recall  f1-score   support

     Bedroom       0.91      0.70      0.79       114
    Building       0.97      0.97      0.97       111
       Store       0.92      0.56      0.70       121
    Industry       0.89      0.77      0.82       122
 Living room       0.72      0.84      0.77       125
     Kitchen       0.75      0.83      0.79        77
     Highway       0.68      0.97      0.80        75
       Coast       1.00      0.49      0.66        77
      Office       0.74      0.95      0.83        79
    Mountain       0.92      0.77      0.84        79
 Inside city       0.58      0.91      0.71        77
      Forest       0.90      0.90      0.90        79
      Surbub       0.72      0.97      0.83        78
      Street       1.00      0.83      0.91        76
 Opencountry       0.80      0.78      0.79        72

    accuracy                           0.81      1362
   macro avg       0.83      0.82      0.81      1362
weighted avg       0.84      0.81      0.81      1362
```

(a) Classification Report for RMSPROP

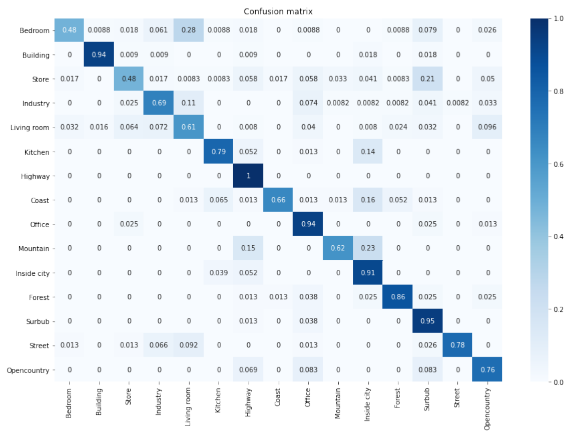


(b) Confusion matrix for RMSPROP

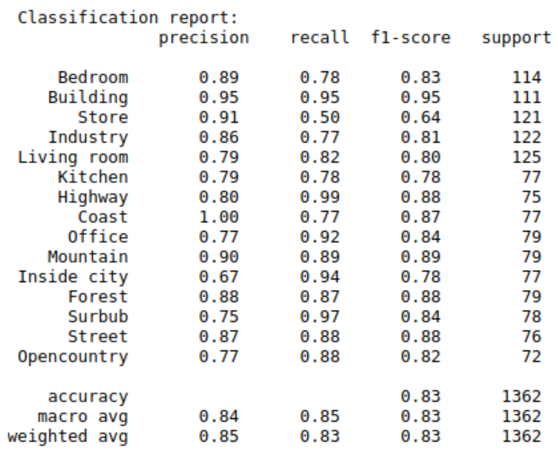


```
              precision    recall  f1-score   support

     Bedroom       0.89      0.48      0.62       114
    Building       0.97      0.94      0.95       111
       Store       0.77      0.48      0.59       121
    Industry       0.78      0.69      0.73       122
 Living room       0.58      0.61      0.60       125
     Kitchen       0.86      0.79      0.82        77
     Highway       0.66      1.00      0.79        75
       Coast       0.94      0.66      0.78        77
      Office       0.67      0.94      0.78        79
    Mountain       0.89      0.62      0.73        79
 Inside city       0.57      0.91      0.70        77
      Forest       0.87      0.86      0.87        79
      Surbub       0.56      0.95      0.70        78
      Street       0.98      0.78      0.87        76
 Opencountry       0.66      0.76      0.71        72

    accuracy                           0.74      1362
   macro avg       0.78      0.76      0.75      1362
weighted avg       0.78      0.74      0.74      1362
```

(c) Classification report for Adam

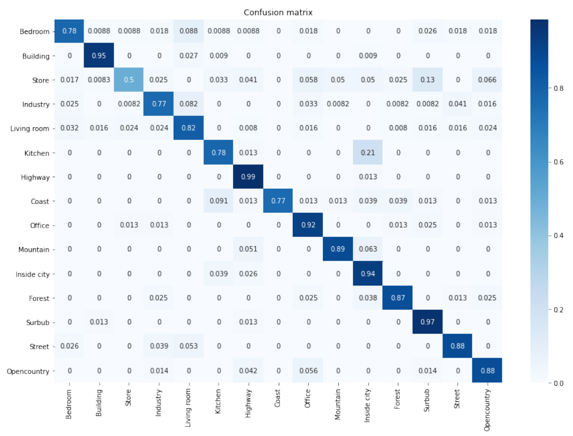


(d) Confusion matrix for Adam



```
Classification report:
              precision    recall  f1-score   support

     Bedroom       0.89      0.78      0.83       114
    Building       0.95      0.95      0.95       111
       Store       0.91      0.50      0.64       121
    Industry       0.86      0.77      0.81       122
 Living room       0.79      0.82      0.80       125
     Kitchen       0.79      0.78      0.78        77
     Highway       0.80      0.99      0.88        75
       Coast       1.00      0.77      0.87        77
      Office       0.77      0.92      0.84        79
    Mountain       0.90      0.89      0.89        79
 Inside city       0.67      0.94      0.78        77
      Forest       0.88      0.87      0.88        79
      Surbub       0.75      0.97      0.84        78
      Street       0.87      0.88      0.88        76
 Opencountry       0.77      0.88      0.82        72

    accuracy                           0.83      1362
   macro avg       0.84      0.85      0.83      1362
weighted avg       0.85      0.83      0.83      1362
```

(e) Classification report for SGD



(f) Confusion matrix for for SGD

Figure 7: Classification Report and Confusion matrices for CNN model trained with different optimizer{a,b}, {c,d } , {e,f}, Classification report and confusion matrix for RMSPROP, Adam, and SGD respectively

2.5 Conclusion

In short, I successfully designed and tested CNN model to classify 15 scenes from scene dataset. The model used ELU as activation function and it was trained over 100 epochs to minimize categorical cross entropy. The model was trained with different data augmentation techniques, learning late, momentum, and optimers. After training the model, I attained the highest classification performance with SGD with accuracy of 83%. The model accuracy was 74% and 81 for Adam and RMSPROP respectively. Though the model performance is high enough, we could improve performance more by trying different model architecture, different activation functions, other optimizer such as Ada delta, and Ada grad.

References