Project 3: Deep Learning for Flower Classification and Object Detection

EEL5934: Applied Machine Learning Project 3

Nathan Oliver Noronha
Electrical and Computer Engineering
University of Florida
Gainesville, USA
n.noronha@ufl.edu

***Abstract*—This report is made for the purpose of Project 3 of the course EEL5934: Applied Machine learning systems.The project broadly comprises of two parts. The first part illustrates flower classification using a flowers dataset with 10 classes of flowers while the second demonstrates object detection for a dataset with vehicle images. According to the mentioned requirements, different neural network architecture are compared and the corresponding decisions and explanations are given.**

I. Flower Classification

In this problem, as we have to predict 10 classes of flowers, this is naturally a classification problem. Hence, I will be utilizing techniques related to classification tasks to train the neural network.

A. CNN Architecture and Hyperparameter Experimentation

For the CNN approach, three models were trained to check and the model with better performance was selected. These models are discussed below:

The first model was a custom neural network architecture with 4 convolutional layers and 1 hidden layer. Experimentation with the hyperparameters seemed led to discard the model since at higher hidden layers, the model started to overfit. With 1 hidden layer, the accuracy was near 55 percent. The dataset had only 1658 samples which led me to believe that they were not enough to learn the weights in the network. Hence, tranfer learning was used.

The core reason behind using transfer learning is to reuse the knowledge gained from one task to perform another related task efficiently and accurately. Transfer learning is extremely relevant within the scope of this project which aims to classify a given input image of a flower into a specific flower family and recognize cars in a given image. The training portion of the datasets allows the model to learn particular identifying features by which it can later recognize and classify objects correctly in the testing portion of the dataset.

The second model used was the MobileNet which is known for its speed and low size. A tradeoff for the speed and low size is that the network is small and hence, has lower accuracy compared to larger models. The highest accuracy achieved for this network was approximately 67 percent. This led me to experiment with the next model.

The third model is a popular standard convolutional neural network model called VGG19 has been used for the purpose of image processing and recognition. Hence, the lower layers of the presented architecture consists of VGG19. The VGG19 model serves as the starting point for feature extraction from the images. As part of the image pre-processing steps, the images have been resized from 300x300 to 224x224 which is the reason the input tensor of the model has been given the dimensions of 224x224 pixels with 3 color channels (RGB). The weights used in the model have been set to the pre-computed values of the famous ImageNet dataset. The output of VGG19 is a multi-dimensional tensor which is fed as an input to the Flatten layer which converts it into a one-dimensional array. There are 2 hideen layers which consists of 64 neurons; making use of the rectified linear unit (relu) activation function that takes the flattened features output and learns to interpret them. Relu helps this layer learn non-linear relationships in the features. A hidden layer with lower neurons underfits. The softmax activation function has been used in the output layer with 10 neurons since we need 10 classes of flowers, as per our target data. Softmax helps obtain a probability distribution over the 10 classes which is why its being used for multi-class classification in this project.

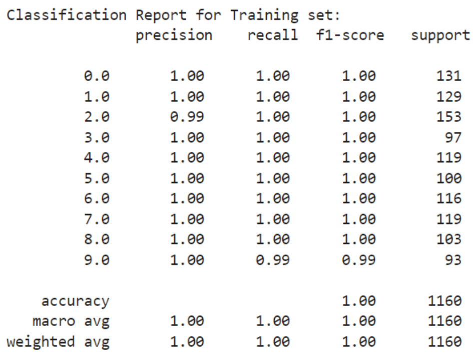
The model uses adam optimizer which adapts the learning rates of individual parameters by computing adaptive learning rates for each parameter based on both the first-order momentum estimates and the second-order estimates of the gradients.

The loss function is set to SparseCategoricalCrossentropy, which expects integer labels and internally performs the necessary conversion to one-hot encoded vectors. It is used in conjunction with softmax activation to convert the model's output into probability distributions over the classes. This helps the loss function to compute the cross-entropy loss between the predicted probabilities and the target classes.

B. Performance in Training and Validation

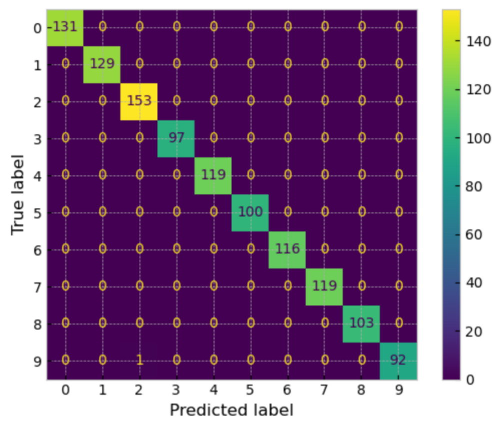
The above-discussed model was trained on the given flower dataset. All the trained models had the problem of overfitting.

This implies that size of the training is too small which can be inferred from 1658 samples. The model is memorizing the training samples instead of learning the underlying patterns. To avoid overfitting, different techniques like regularization using a dropout of 50 percent was used with batch normalization. Batch Normalization helps to stabilize the training process by ensuring that inputs to each layer are centered around zero and have a similar scale.



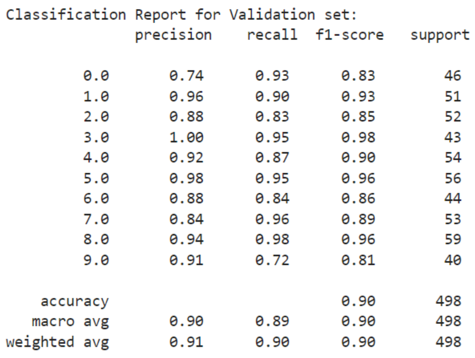
```
Classification Report for Training set:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       131
         1.0       1.00      1.00      1.00       129
         2.0       0.99      1.00      1.00       153
         3.0       1.00      1.00      1.00        97
         4.0       1.00      1.00      1.00       119
         5.0       1.00      1.00      1.00       100
         6.0       1.00      1.00      1.00       116
         7.0       1.00      1.00      1.00       119
         8.0       1.00      1.00      1.00       103
         9.0       1.00      0.99      0.99        93

    accuracy                           1.00      1160
   macro avg       1.00      1.00      1.00      1160
weighted avg       1.00      1.00      1.00      1160
```

Fig. 1. Classification Report for Training Set



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	131	0	0	0	0	0	0	0	0	0
1	0	129	0	0	0	0	0	0	0	0
2	0	0	153	0	0	0	0	0	0	0
3	0	0	0	97	0	0	0	0	0	0
4	0	0	0	0	119	0	0	0	0	0
5	0	0	0	0	0	100	0	0	0	0
6	0	0	0	0	0	0	116	0	0	0
7	0	0	0	0	0	0	0	119	0	0
8	0	0	0	0	0	0	0	0	103	0
9	0	0	1	0	0	0	0	0	0	92

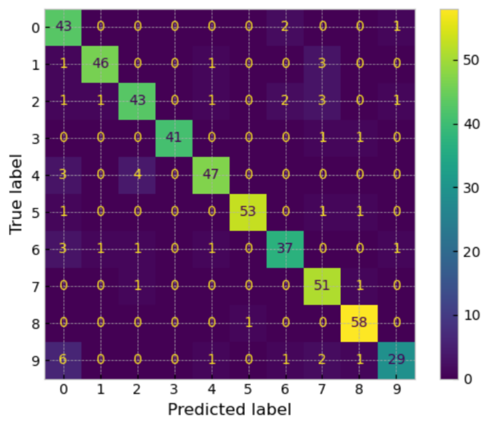
Fig. 2. Confusion Matrix for Training Set



```
Classification Report for Validation set:
               precision    recall  f1-score   support

         0.0       0.74      0.93      0.83        46
         1.0       0.96      0.90      0.93        51
         2.0       0.88      0.83      0.85        52
         3.0       1.00      0.95      0.98        43
         4.0       0.92      0.87      0.90        54
         5.0       0.98      0.95      0.96        56
         6.0       0.88      0.84      0.86        44
         7.0       0.84      0.96      0.89        53
         8.0       0.94      0.98      0.96        59
         9.0       0.91      0.72      0.81        40

    accuracy                           0.90       498
   macro avg       0.90      0.89      0.90       498
weighted avg       0.91      0.90      0.90       498
```

Fig. 3. Classification Report for Validation Set

While training, the model uses a early stopping callback which can help in overfitting by stopping the training when validation loss, stops improving with a patience level of 10 (10 Epochs of decreasing validation loss.)



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	43	0	0	0	0	0	2	0	0	1
1	1	46	0	0	1	0	0	3	0	0
2	1	1	43	0	1	0	2	3	0	1
3	0	0	0	41	0	0	0	1	1	0
4	3	0	4	0	47	0	0	0	0	0
5	1	0	0	0	0	53	0	1	1	0
6	3	1	1	0	1	0	37	0	0	1
7	0	0	1	0	0	0	0	51	1	0
8	0	0	0	0	0	1	0	0	58	0
9	6	0	0	0	1	0	1	2	1	29

Fig. 4. Confusion Matrix for Validation Set

C. Learning Curves

Below is the plot of the learning curves while training.

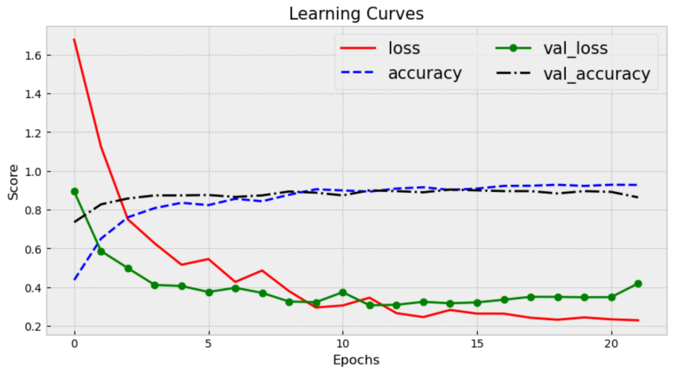
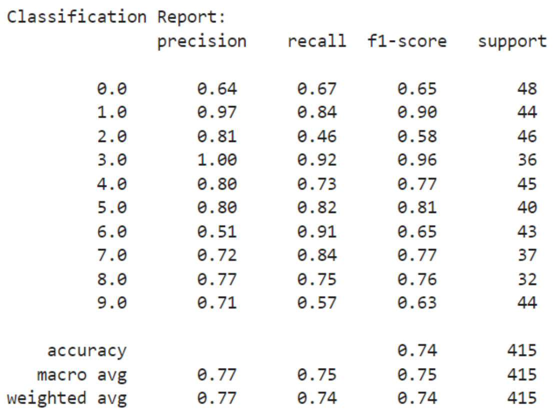


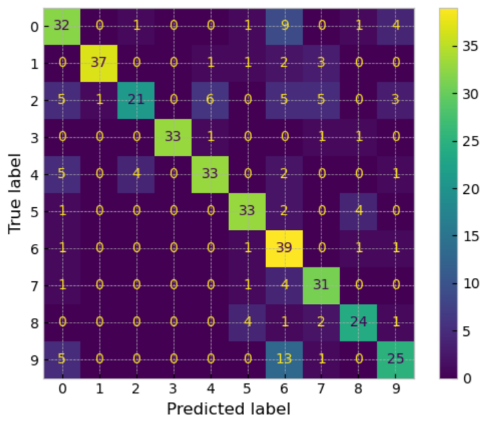
Fig. 5. Learning Curves for Flower Classification model, trained with transfer learning using VGG19 model.

D. Test Results



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

         0.0       0.64      0.67      0.65        48
         1.0       0.97      0.84      0.90        44
         2.0       0.81      0.46      0.58        46
         3.0       1.00      0.92      0.96        36
         4.0       0.80      0.73      0.77        45
         5.0       0.80      0.82      0.81        40
         6.0       0.51      0.91      0.65        43
         7.0       0.72      0.84      0.77        37
         8.0       0.77      0.75      0.76        32
         9.0       0.71      0.57      0.63        44

    accuracy                           0.74       415
   macro avg       0.77      0.75      0.75       415
weighted avg       0.77      0.74      0.74       415
```

Fig. 6. Classification Report for Test Set



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	32	0	1	0	0	1	9	0	1	4
1	0	37	0	0	1	1	2	3	0	0
2	5	1	21	0	6	0	5	5	0	3
3	0	0	0	33	1	0	0	1	1	0
4	5	0	4	0	33	0	2	0	0	1
5	1	0	0	0	0	33	2	0	4	0
6	1	0	0	0	0	1	39	0	1	1
7	1	0	0	0	0	1	4	31	0	0
8	0	0	0	0	0	4	1	2	24	1
9	5	0	0	0	0	0	13	1	0	25

Fig. 7. Confusion Matrix for Test Set

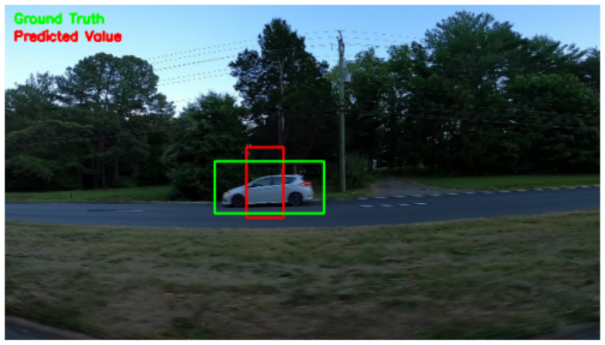
II. Object Detection

In this problem, the objective is to detect vehicles in given images. The given images are training data while the target data are continuous values for the bounding box with two X-axis and Y-axis coordinates defining the height and the width of the rectangle for the bounding box. As we have to predict continuous values, this is a regression problem. Hence, I will be using functions more suited to regression tasks while training the neural network. The target data has been normalized for the model to better generalize the predicted values.

A. *CNN Architecture and Hyperparameter Experimentation*

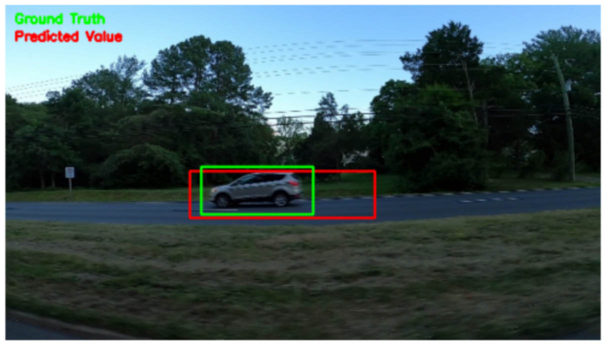
For the CNN model, the above-mentioned VGG19 model is used here. The lower layers of the model, which consist of the convolutional layers use the lower layers of VGG19. However, the image is not resized here as the bounding box in the target is defined for the original size of 380x676. The weights used in the model have been set to the pre-computed values of the famous ImageNet dataset. The upper layers are sequential with 3 hidden layers. The first layer consists of 128 neurons, second consists of 64 and the third consists of 32 neurons. Each layer uses the relu activation function. Each layer has a Batch Normalization and Dropout regularization of 50 percent. These are used as there was the problem of the vanishing gradient while only using the relu activation function. The output layer has 4 neurons for 4 corresponding outputs with the sigmoid activation function, since we have normalized the target data between 0 and 1. The predicted output will also be between 0 and 1, which will be scaled later.

The model uses adam optimizer which adapts the learning rates of individual parameters by computing adaptive learning rates for each parameter based on both the first-order momentum estimates and the second-order estimates of the gradients.



```
IOU for imagevid_4_10040.jpg:  0.7857395455751038
```

Fig. 8. Example 1: Training set performance with IoU with bounding box for ground truth and predicted values



```
IOU for imagevid_4_9980.jpg:  0.7882091818854406
```

Fig. 9. Example 2: Training set performance with IoU with bounding box for ground truth and predicted values

B. *Performance in Training and Validation*

C. *Learning Curves*

10 shows the learning curves for the object detection model trained using VGG19 model

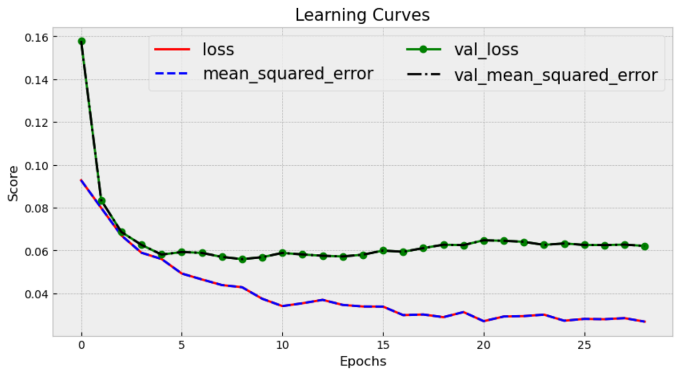


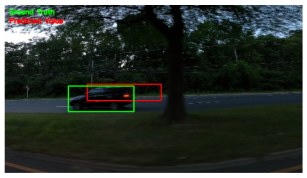
Fig. 10. Learning Curves for Object Detection model, trained with transfer learning using VGG19 model.

D. *Test Results*

1) *Case: No target labels in test set:* As we had no target labels for the test set, MakeSense AI was used to create

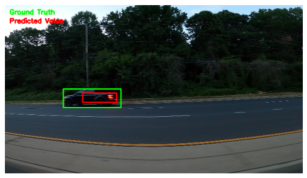
a few test samples. The test was then performed on these samples. For the evaluation of performance, a metric called as Intersection over Union (IoU) is used which measures the extent of overlap between two bounding boxes. In the context of tasks such as object detection and segmentation, IoU assesses the degree of overlap between the ground truth and prediction regions. Using IoU, we can measure the overlapping Region of Interest.

2) Case: No vehicle present in the image: For this problem, the data was manually added, with bounding box values of 0,0,0,0 which correspond to no bounding box as there is no car.



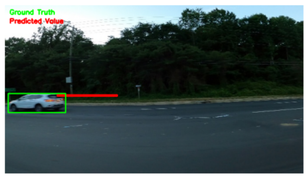
```
IOU for imagevid_5_27920.jpg:  0.5620777788383019
```

Fig. 11. Example 1: Test set performance with IoU with bounding box for ground truth(MakeSense AI) and predicted values



```
IOU for imagevid_5_28420.jpg:  0.7032507808032376
```

Fig. 12. Example 2: Test set performance with IoU with bounding box for ground truth (MakeSense AI) and predicted values



```
IOU for imagevid_5_28260.jpg:  0.06671339524870454
```

Fig. 13. Example 2: Failed case in Test set performance with IoU with bounding box for ground truth (MakeSense AI) and predicted values

3) Qualitative and Quantitative Results:

Acknowledgment

I would like to extend our sincere appreciation and gratitude to Dr Catia Silva and the Teaching Assistant's for their valuable support and feedback. I would also like to thank the department of Electrical and Computer Engineering at the University of Florida.

References

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition (Aurélien Géron, 2019)
[2] Python Machine Learning - Third Edition (Sebastian Raschka , Vahid Mirjalili, 2019)
[3] https://www.tensorflow.org/
[4] https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/