COMPUTER VISION

ASSIGNMENT-4

QUESTION-1

1. Show the calculation of output filter size at each layer of CNN.

We can calculate the output filter size at each layer using the following formula:

output_size = (input_size - kernel_size + 2*padding) / stride + 1

where:

input_size: the size of the input feature map

kernel_size: the size of the convolutional kernel

padding: the amount of zero padding added to the input feature map

stride: the stride of the convolutional kernel

Let's apply this formula to each layer of the CNN:

Layer 1: Conv2D layer with 32 filters and a kernel size of (3, 3), followed by a ReLU activation function:

The input shape is (28, 28, 1) for the MNIST dataset. We don't have any padding, and the stride is 1 by default. So, the output size is:

output_size = (28 - 3 + 2*0) / 1 + 1 = 26

Therefore, the output shape is (26, 26, 32).

Layer 2: MaxPooling2D layer with a pool size of (2, 2):

The input shape is (26, 26, 32) from the previous layer. We have a pool size of (2, 2) and a stride of 2 by default. So, the output size is:

output_size = (26 - 2 + 2*0) / 2 + 1 = 13

Therefore, the output shape is (13, 13, 32).

Layer 3: Conv2D layer with 64 filters and a kernel size of (3, 3), followed by a ReLU activation function:

The input shape is (13, 13, 32) from the previous layer. We don't have any padding, and the stride is 1 by default. So the output size is:

output_size = (13 - 3 + 2*0) / 1 + 1 = 11

Therefore, the output shape is (11, 11, 64).

Layer 4: MaxPooling2D layer with a pool size of (2, 2):

The input shape is (11, 11, 64) from the previous layer. We have a pool size of (2, 2) and a stride of 2 by default. So, the output size is:

output_size = (11 - 2 + 2*0) / 2 + 1 = 6. Therefore, the output shape is (6, 6, 64).

Layer 5: Flatten layer:

The input shape is (6, 6, 64) from the previous layer. We flatten this tensor into a one-dimensional tensor of shape (2304,).

Layer 6: Dense layer with 128 units and a ReLU activation function:

The input shape is (2304,) from the previous layer. We apply a fully connected layer with 128 units and a ReLU activation function.

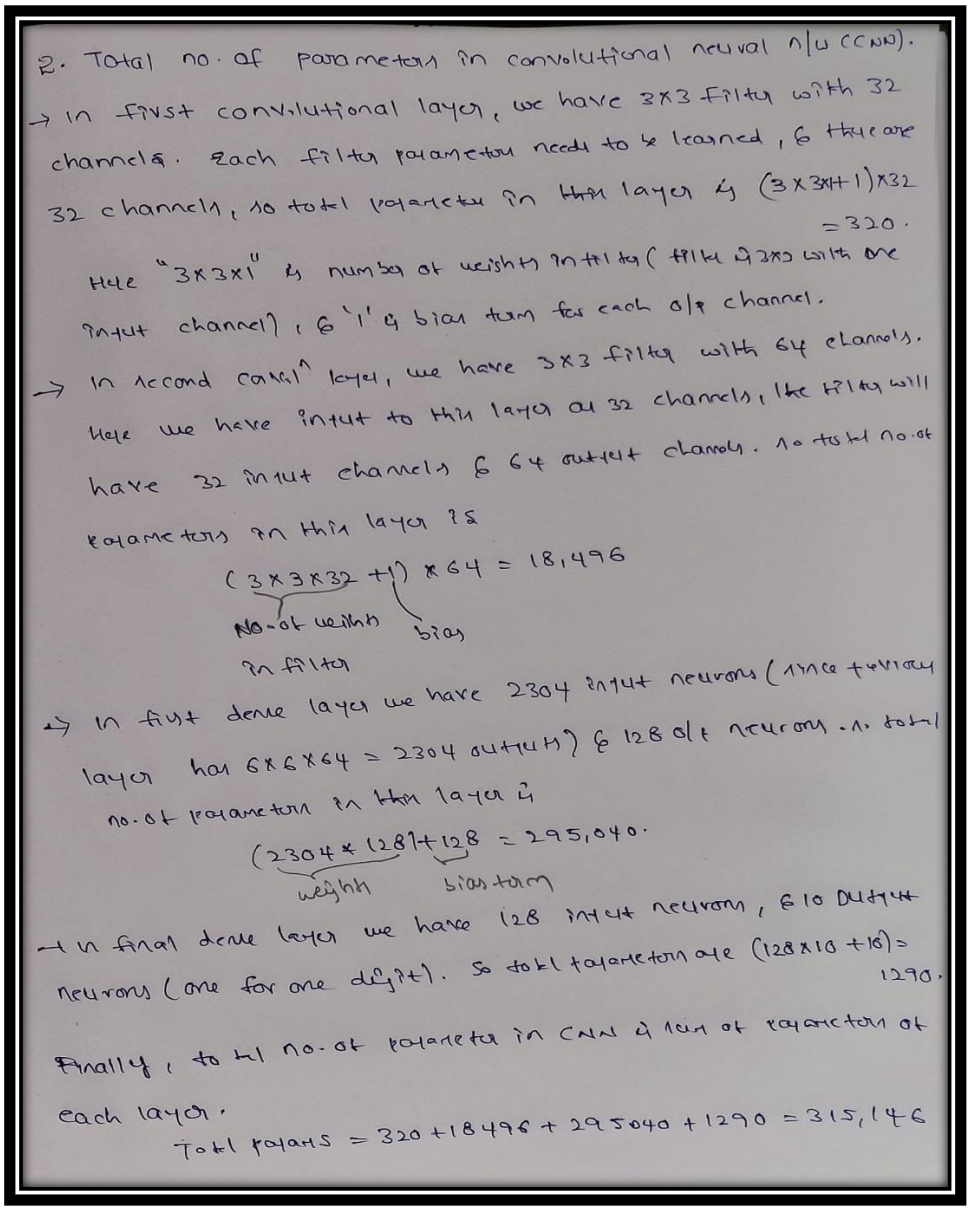
Layer 7: Dropout layer with a rate of 0.5:

We apply a dropout layer with a rate of 0.5 to the output of the previous layer.

Layer 8: Dense layer with 10 units and a softmax activation function:

The final layer is a dense layer with 10 units and a softmax activation function, which outputs the class probabilities for each of the 10 digits.

2. Calculate the number of parameters in your CNN. Calculation steps should be clearly shown in the report.



3. Report the following on test data: (should be implemented from scratch)

a. Confusion matrix

A confusion matrix is a table that is often used to evaluate the performance of a machine learning model. It shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class. In a binary classification problem, the confusion matrix will have 2 rows and 2 columns, where each row corresponds to the actual class and each column corresponds to the predicted class. The cells in the matrix represent the number of instances that belong to a particular combination of actual and predicted class.

Here's an example of what a confusion matrix might look like for a binary classification problem:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The diagonal elements of the matrix represent the correctly classified instances (TP and TN), while the off-diagonal elements represent the misclassified instances (FP and FN). From the confusion matrix, various performance metrics can be derived, such as accuracy, precision, recall, and F1-score.

Here I have given the confusion matrix for 10 classes.

```
Confusion Matrix:
[[978   0   0   0   0   0   0   1   1   0]
 [  0 1133   0   1   0   0   0   0   1   0]
 [  1   2 1023   0   0   0   0   6   0   0]
 [  0   0   1 1007   0   1   0   0   1   0]
 [  0   0   0   0 979   0   1   0   0   2]
 [  1   0   1   7   0 880   1   1   0   1]
 [  3   2   0   0   1   1 951   0   0   0]
 [  0   2   3   0   0   0   0 1017   1   5]
 [  2   0   4   2   0   1   0   2 961   2]
 [  2   0   0   2   5   3   0   2   4 991]]
```

Fig.1 Confusion Matrix using CNN

b. Overall and class wise accuracy

Overall accuracy is a performance metric that measures the proportion of correct predictions made by a machine learning model across all classes. It is computed as the ratio of the number of correct predictions to the total number of predictions made. Overall accuracy is a useful metric when the classes are balanced, i.e., when the number of instances in each class is roughly equal.

In this question overall accuracy was found to be `99.2%`

However, in imbalanced datasets where one or more classes have significantly more instances than others, overall accuracy can be misleading. In such cases, it's important to look at class-wise accuracy, which measures the proportion of correct predictions made for each individual class.

Class-wise accuracy gives a better understanding of how well the model is performing for each class, and can highlight any biases or errors that might be present. In this question class wise accuracy was found to be

```
Overall Accuracy: 99.20%
Class 0 Accuracy: 99.80%
Class 1 Accuracy: 99.82%
Class 2 Accuracy: 99.13%
Class 3 Accuracy: 99.70%
Class 4 Accuracy: 99.69%
Class 5 Accuracy: 98.65%
Class 6 Accuracy: 99.27%
Class 7 Accuracy: 98.93%
Class 8 Accuracy: 98.67%
Class 9 Accuracy: 98.22%
```

c. ROC curve.

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model. It is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The TPR (also known as sensitivity or recall) is the proportion of actual positive instances that are correctly identified by the model, while the FPR is the proportion of actual negative instances that are incorrectly classified as positive. The ROC curve provides a visual representation of the trade-off between the TPR and FPR at different threshold settings.

The area under the ROC curve (AUC) is a commonly used metric for evaluating the performance of a binary classification model. It represents the overall performance of the model across all possible threshold settings, with a value of 1.0 indicating a perfect classifier and a value of 0.5 indicating a random classifier. A higher AUC indicates better performance curves for individual classes is shown in colab notebook.

In this question ROC curve, I got is shown below:

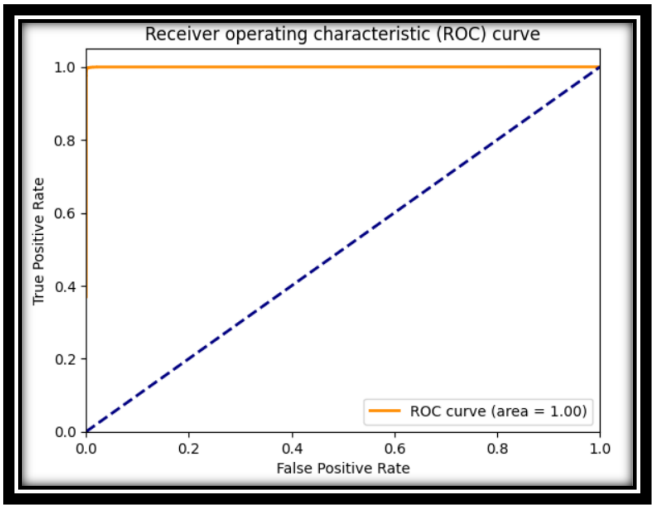


Fig.2 ROC Curve using CNN

4. Report loss curve during training.

The loss curve during training is a plot of the loss function values computed at each iteration of the training process. The loss function is a mathematical function that measures the difference between the predicted output of a machine learning model and the actual output. During training, the model's parameters are adjusted to minimize the value of the loss function, which leads to better performance on the training data.

The loss curve provides insight into how well the model is learning and improving during training. A decreasing loss curve indicates that the model is learning and improving, while an increasing loss curve indicates that the model is not learning or overfitting to the training data. Overfitting occurs when the model is too complex and memorizes the training data instead of learning general patterns that can be applied to new data. In this question loss curve is shown below,

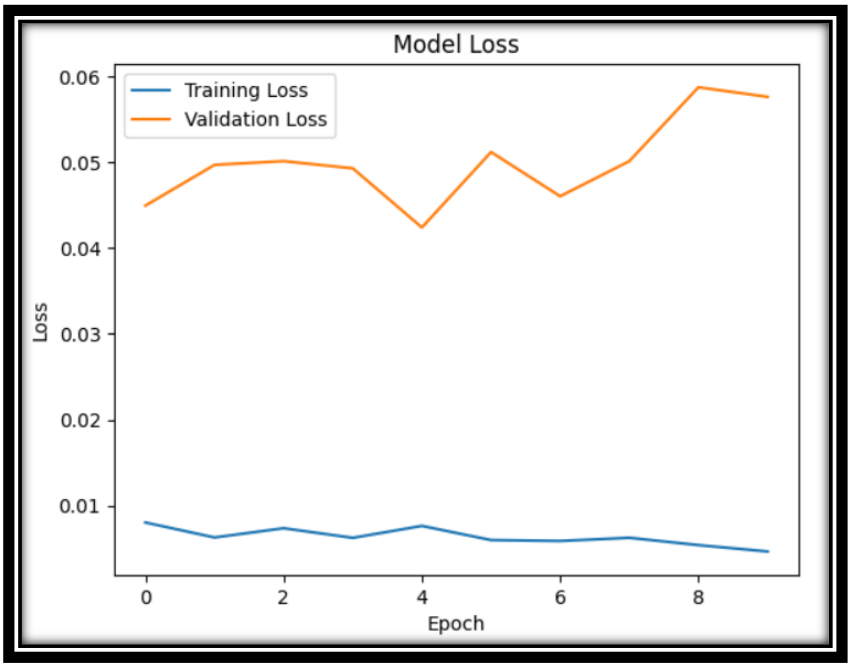


Fig.3 Training Loss Curve

Ideally, we want the training loss curve to decrease as the number of epochs increases, which means the model is getting better at predicting the output. Here we can see that loss curve is gradually decreasing.

5. Replace your CNN with resnet18 and compare it with all metrics given in part 3. Comment on the final performance of your CNN and resnet18.

ResNet (short for "Residual Networks") is a deep neural network architecture which is one of the most popular deep learning architectures for image recognition tasks, and has achieved state-of-the-art results on a number of benchmarks. The key innovation in ResNet is the use of "skip connections" or "shortcut connections" that allow the network to learn residual functions. These connections allow information to bypass one or more layers in the network, which helps to address the problem of vanishing gradients that can occur in very deep networks.

ResNet models come in several different depths, ranging from the relatively shallow ResNet-18 to the very deep ResNet-152. The deeper models have shown to be particularly effective at tasks such as image recognition and object detection. confusion matrix is a table that is often used to evaluate the performance of a machine learning model.

a. Confusion matrix

Confusion matrix for 10 classes using resnet18 is shown below:

```
Confusion Matrix:
[[977   0   2   0   0   0   0   1   0   0]
 [0  1124   0   0   0   4   4   3   0   0]
 [0   0  1028   0   0   0   0   4   0   0]
 [0   0   2  996   0  10   0   1   1   0]
 [0   0   0   0  965   2   1   6   0   8]
 [1   0   0   1   0  889   1   0   0   0]
 [4   1   0   0   0   4  949   0   0   0]
 [0   2   2   0   0   0   0  1023   0   1]
 [0   1   2   1   0   4   0   3  957   6]
 [0   0   0   0   2   4   0   2   0  1001]]
```

Fig.4 Confusion Matrix using resnet18

b. Overall and class wise accuracy.

Using resnet18 the **overall accuracy** was **99.09%. Class wise accuracy** is shown below**:**

```
Overall Accuracy: 99.09%
Class 0 Accuracy: 99.69%
Class 1 Accuracy: 99.03%
Class 2 Accuracy: 99.61%
Class 3 Accuracy: 98.61%
Class 4 Accuracy: 98.27%
Class 5 Accuracy: 99.66%
Class 6 Accuracy: 99.06%
Class 7 Accuracy: 99.51%
Class 8 Accuracy: 98.25%
Class 9 Accuracy: 99.21%
```

c. ROC curve.

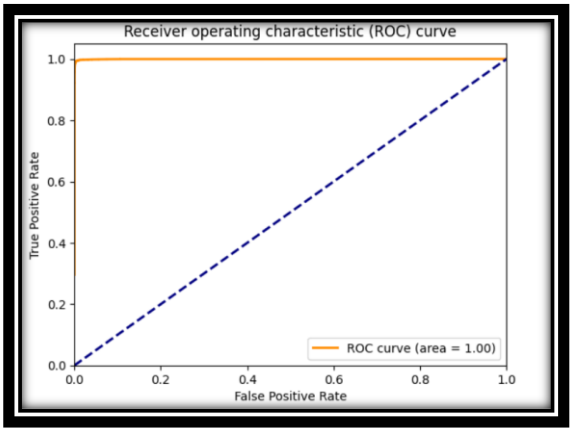
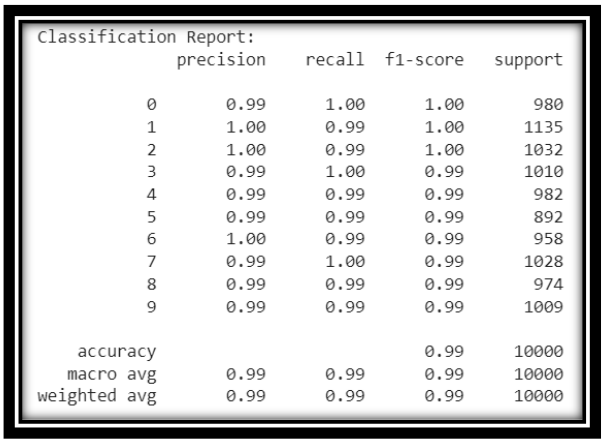


Fig.5 ROC Curve using renet18

Comment on the final performance of your CNN and resnet18.

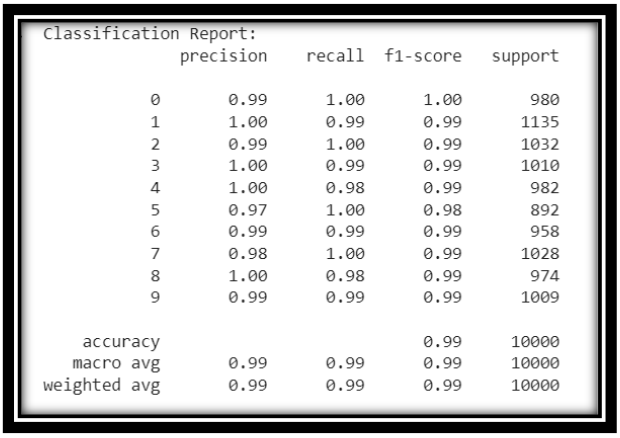
Classification report of CNN and resnet18:



Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	980
1	1.00	0.99	1.00	1135
2	1.00	0.99	1.00	1032
3	0.99	1.00	0.99	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958
7	0.99	1.00	0.99	1028
8	0.99	0.99	0.99	974
9	0.99	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Fig.6 Report using CNN



Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	980
1	1.00	0.99	0.99	1135
2	0.99	1.00	0.99	1032
3	1.00	0.99	0.99	1010
4	1.00	0.98	0.99	982
5	0.97	1.00	0.98	892
6	0.99	0.99	0.99	958
7	0.98	1.00	0.99	1028
8	1.00	0.98	0.99	974
9	0.99	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Fig.7 Report using renet18

The results suggested that the CNN model performs slightly better than the ResNet18 model in terms of overall accuracy, with an accuracy of 99.2% compared to 99.09%. In roc curve both CNN and Resnet18 have shown the equal performance on test data. Additionally, the precision, recall and F1-score of the CNN model is same as that of the ResNet18 model, with a value of 0.99

Collab Link:

https://colab.research.google.com/drive/1XHq-jwwqasn8OD8Q3ObKeD3SM1YHlir2?usp=share_link

QUESTION-2

Image captioning refers to the process of generating a textual description of the content of an image. This is typically achieved through the use of artificial intelligence and deep learning algorithms, which analyse the visual features of an image and generate a corresponding caption. Image captioning has numerous applications, including assisting visually impaired individuals to understand images, enhancing search engine capabilities, and improving the accessibility of social media platforms.

Procedure:

-First after importing necessary libraries I have loaded a text file containing image captions for the Flickr8k dataset and prints the first 300 characters of the file.

- Then load_descriptions function takes in the doc string which contains the image descriptions and returns a dictionary object where the keys are the image IDs and the values are the list of descriptions associated with the corresponding image ID.

- Then after InceptionV3 model has been loaded successfully. This model is pretrained on the ImageNet dataset and is widely used for image classification and feature extraction tasks. We used this model to extract features from our images. After all pre-processing of data and converting them to vector format and encoding of test and train data is done.

- The data generator function is used to generate batches of data for training the model using the fit_generator () method. The generator takes in four arguments: descriptions (a dictionary containing image descriptions), photos, wordtoix (a dictionary mapping words to integers), max_length (the maximum length of the description sequences), and num_photos_per_batch (the number of photos to include in each batch). he generator then yields a batch of data consisting of the X1 (image features), X2 (input sequences), and y (output sequences) arrays.

-Then a LSTM Model is defined. The model takes two inputs, the image features and sequences of word indexes, and outputs a probability distribution over the vocabulary for the next word in the sequence.

-Training is done using this model and training loss curve is plotted as shown in **Fig.1** and the trained weights of the model have been saved to the file.

-Finally, prediction is done on a random image caption is generated as shown in the result.

1. Split the dataset into train and test sets appropriately. You can further split the train set for validation. Train your model on the train set. Report loss curve during training.

The loss curve during training is a plot of the loss function values computed at each iteration of the training process. The loss function is a mathematical function that measures the difference between the predicted output of a machine learning model and the actual output. During training, the model's parameters are adjusted to minimize the value of the loss function, which leads to better performance on the training data.

The loss curve provides insight into how well the model is learning and improving during training. A decreasing loss curve indicates that the model is learning and improving, while an increasing loss curve indicates that the model is not learning or overfitting to the training data. Overfitting occurs when the model is too complex and memorizes the training data instead of learning general patterns that can be applied to new data. In this question loss curve is shown below:

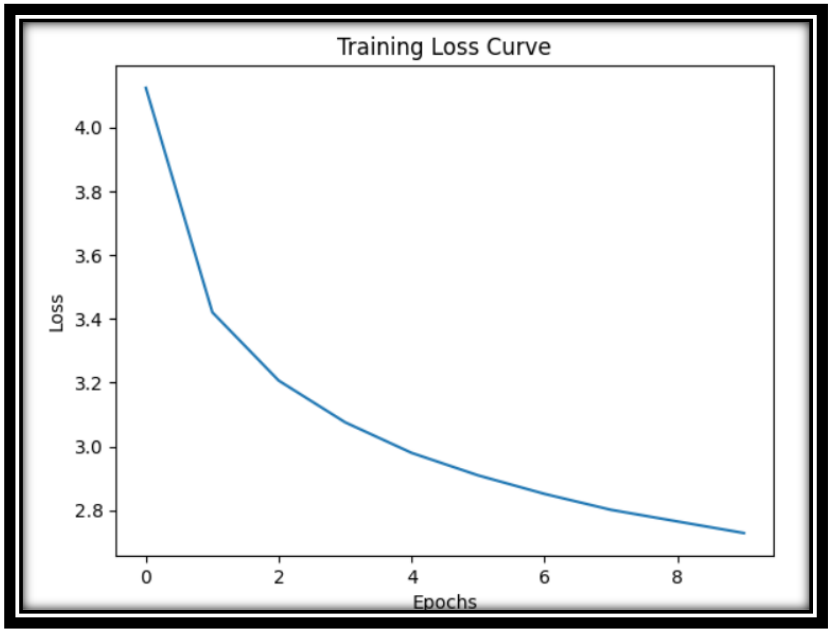


Fig.1 Training Loss Curve

Ideally, we want the training loss curve to decrease as the number of epochs increases, which means the model is getting better at predicting the output. However, if the training loss curve decreases too quickly, it may indicate that the model is overfitting the training data, which means it is memorizing the training data rather than generalizing to new data. On the other hand, if the training loss curve is not decreasing or increasing, it may indicate that the model is underfitting the data, which means it is not able to capture the underlying patterns in the data. Here In our we can say that model was able to generalize the data very well.

2.Choose an existing evaluation metric or propose your metric to evaluate your model. Specify the reason behind your selection/proposal of the metric. Report the final results on the test set.

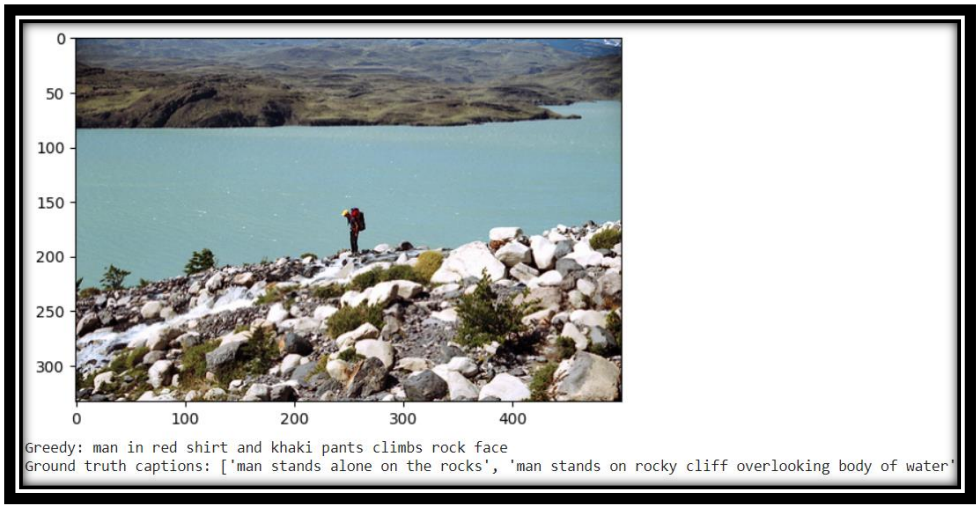
Here in order to find the accuracy of the model I have chosen **word error rate (WER)** as a performance metric.

Word Error Rate (WER) is a metric that is commonly used to evaluate the performance of recognition systems. It measures the difference between the original transcript and the transcript produced by the recognition system is computed by dividing the total number of word errors (insertions, deletions, and substitutions) by the total number of words in the original transcript. The resulting ratio is multiplied by 100 to obtain a percentage.

WER is often used to compare the performance of different captioning systems. However, it should be noted that WER does not take into account the semantic correctness of the transcript, and may not fully capture the quality of the recognition system in all situations.

Here I have a WER of 0.44, it means that my captioning system has an **accuracy of 56%**. This means that out of every 100 words predicted, my system is making errors in recognizing approximately 46 of them. But system was able to caption the image very good. But this performance metric (WER) takes absolute difference between captions. This performance metric doesn’t consider the context of the image.

Results:



Colab Link:

https://colab.research.google.com/drive/1nXHjMrkGXIyS9WC3NNFifUh1fPZ1ADdI?usp=share_link

Credits:

https://github.com/hlamba28/Automatic-Image-Captioning.git

QUESTION-3

YOLO (You Only Look Once) is an object detection algorithm that can detect objects within an image and determine their class and location in real-time with high accuracy. Unlike other object detection algorithms that use a sliding window approach, YOLO divides the image into a grid of cells and predicts the bounding box and class probabilities for each cell. This approach allows YOLO to process images quickly and efficiently, making it suitable for real-time applications such as autonomous driving, robotics, and surveillance. Here I have implemented the one of the latest versions of YOLO, i.e., YOLOv8, which improves the accuracy and speed of the algorithm.

Procedure:

Here are the steps I have implemented to train YOLOv8 version of YOLO to train on deer custom dataset and to detect deer image in test data.

1.Preparing dataset: Deer image dataset with annotation of each image with bounding boxes and class labels for each object is formed.

2.Configure the YOLO model: YOLO models come in different versions (e.g., YOLOv3, YOLOv4), and each one has its own configuration file that specifies hyperparameters, such as learning rate, batch size, and image input size. Here I have used YOLOV8 model to detect the deer images.

3.Download pre-trained weights: I have download pre-trained weights for the YOLOV8 model. Then I have fine-tuned these weights on my custom dataset.

4.Train the model: I have used the annotated dataset to train the YOLO model. This typically involves feeding batches of images into the model and adjusting the weights based on the error between the predicted bounding boxes and the ground-truth bounding boxes.

5.Evaluate the model: Once the model is trained, I have given some test images to predict the deer in the test images.

Performance metric chosen:

For YOLO (You Only Look Once) object detection, there are several performance metrics that can be used to evaluate the model's performance. The best metric depends on the specific application and the desired trade-off between precision and recall.

Here I have used IoU for YOLO object detection. For matching ground truth with predicted bounding boxes, it is good measure.

Intersection over Union (IoU): IoU measures the overlap between the ground truth bounding boxes and the predicted bounding boxes. A high IoU score indicates that the predicted bounding box is close to the ground truth bounding box.

In general, IoU is a good metric for evaluating the accuracy of object localization. Accuracy is a commonly used performance metric for image classification tasks. It represents the percentage of correctly classified images out of the total number of images in the dataset. For example, if a model classifies 90 out of 100 images correctly, its accuracy is 90%. In general, accuracy is a good metric when the dataset is balanced, meaning that each class has roughly the same number of images.

Comparison between HOG detector and YOLO:

HOG: HOG (Histogram of Oriented Gradients) is a feature extraction technique used in computer vision and image processing. HOG works by dividing the image into small cells and computing the gradient orientation and magnitude for each pixel in the cell. These gradient values are then used to create a histogram of gradient orientations over the cell. The cell histograms are then normalized and concatenated to form a feature vector that describes the local gradient structure of the image. The idea behind HOG is to describe the gradient of the image in a local region, which can be used to represent the local appearance and shape of an object. The accuracy of HOG detector that was implemented in assignment-3 was found to be **0.75**.HOG detector has misclassified some of the images as shown in figure below.

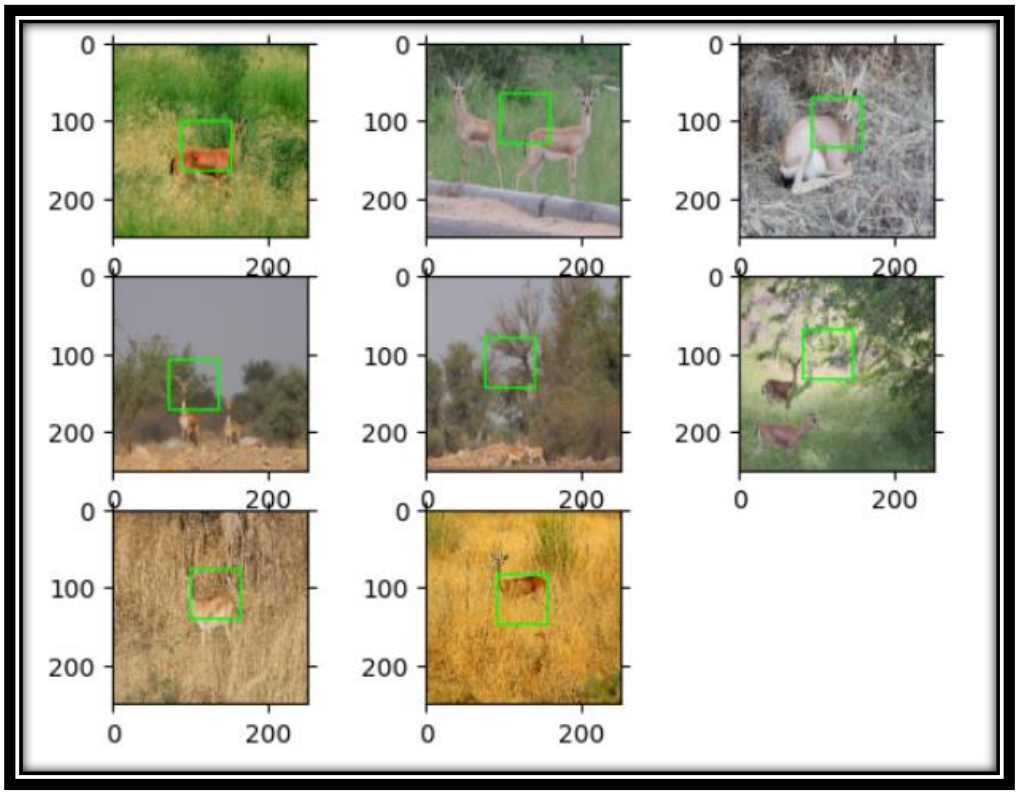


Fig.1 Result using HOG detector

YOLOV8:

YOLO uses a single neural network to make predictions, which allows for end-to-end training and optimization. The network consists of convolutional layers that extract features from the image, followed by fully connected layers that predict the class probabilities and bounding box coordinates for each cell. The algorithm is based on a single convolutional neural network (CNN) that simultaneously predicts object bounding boxes and their corresponding class probabilities.

Here precision and recall are used as evaluation metrics to measure the performance of the model. Precision measures the fraction of true positives out of all the positive predictions made by the model, while recall measures the fraction of true positives out of all the actual positive instances in the dataset. A precision-recall curve is a graph that shows the trade-off between precision and recall for different classification thresholds. The Precision-Recall curve of the model during training is shown below:

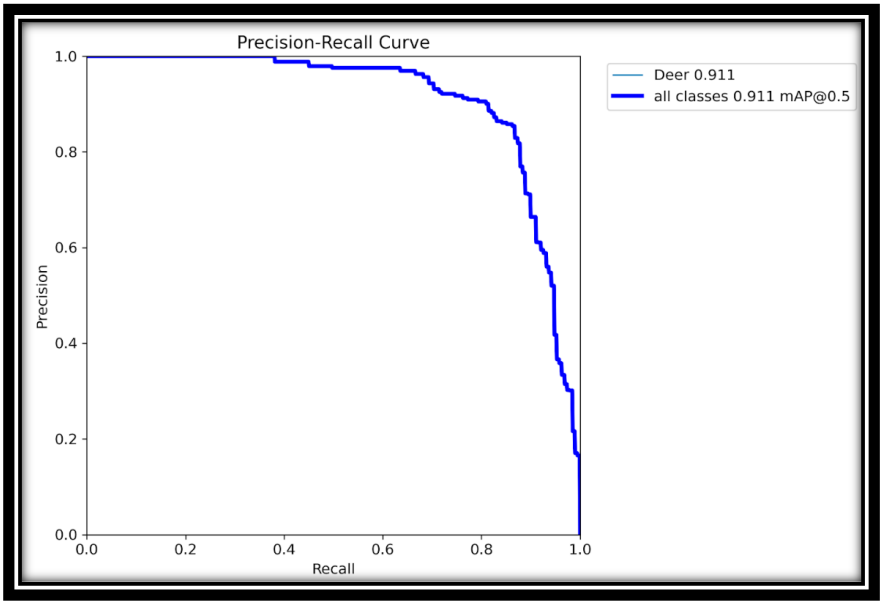


Fig.2 P-R Curve

During the training of a machine learning model, a confusion matrix is used as an evaluation metric to measure the performance of the model. A confusion matrix is a table that shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for a binary classification problem.

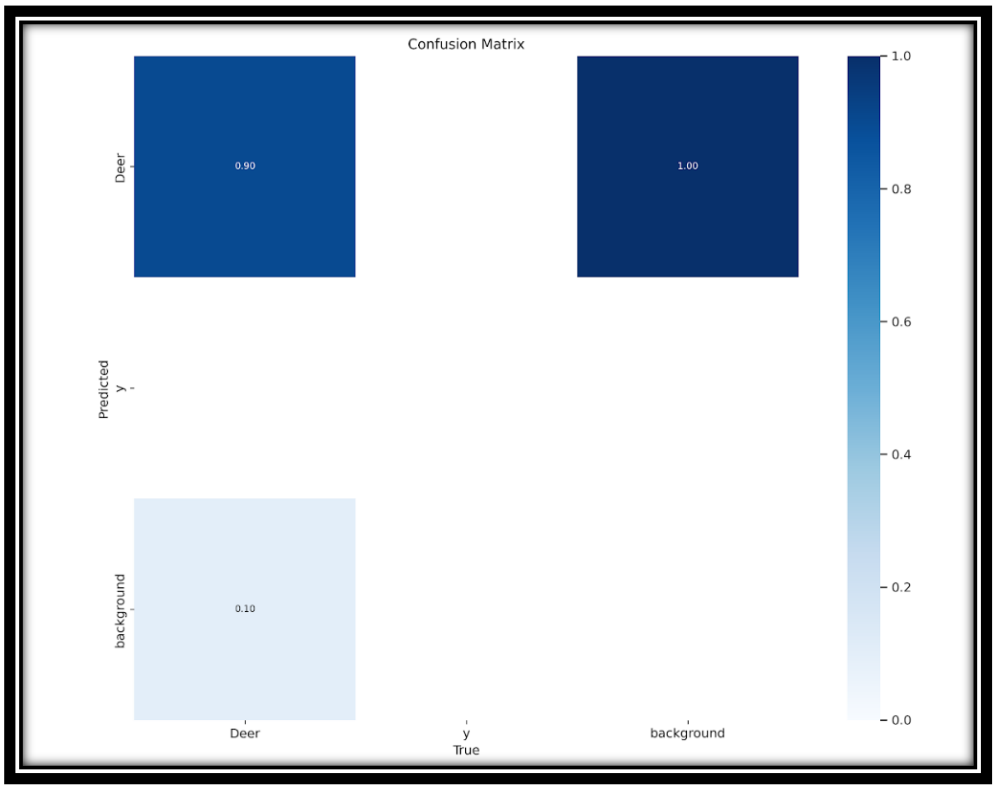


Fig.3 Confusion Matrix

Confusion matrix shows that True positive i.e., model has predicted 90 percent correctly the deer images.

Here are the results of **YOLOV8** model on test images:

```
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [01:41<00:00, 11.32s/it]
                   all        139        189       0.86      0.841      0.911      0.667
                  Deer        139        189       0.86      0.841      0.911      0.667
```

The evaluation the performance of the model on a test set is given above picture. The evaluation metrics used are precision (P), recall (R), and mean average precision. The evaluation has been performed on a total of 139 images and the model has detected 189 instances of objects. The precision and recall for all the classes combined are 0.86 and 0.841, respectively, and the mean average precision is 0.911. The last column "mAP50-95" is indicating the mAP score over a wider range of IoU thresholds. These evaluation metrics suggest that the model is performing reasonably well in detecting objects, especially Deer class. Yolo has missed the detection where there were two deer in the image. Here I have run only 4 epochs for training my YOLO model because of Laptop GPU constraint. If We can increase the number if epochs the precision and accuracy will increase exponentially.

Fig.4 Result using YOLO detector

Thus, from above comparisons we can conclude that yolov8 model has performed well than HOG detector.

Collab Link:

https://colab.research.google.com/drive/1leVra7e9rX6ZH08256ro_F2oT0jaiRjy?usp=share_link

Credits:

https://github.com/pjreddie/darknet.git