CV Assignment 1

-Vivaswan Nawani
2021217

Question 1: Theory

1

A) MSE is generally used as a loss function in regression problems, where the output range is continuous and can take any value on the real number line. However, in this particular case, we are dealing with a problem of binary classification, where the output is either 0 or 1; that's why it does not make sense to use MSE here.

MSE is also sensitive to outliers as we are taking the square of the error since, in binary classification, all values are at the extremes, i.e. 0 or 1; any outliers can adversely affect our model

MSE is also non-probabilistic, therefore drawing meaningful conclusions from MSE in the case of binary classification can be a challenge, as we expect the probability of each class as output in binary classification.

B) $BCE(y, \hat{y}) = - (y \log (\hat{y}) + (1 - y) \log(1 - \hat{y}))$

C) $BCE(0, 0.9) = -(0*\log(0.9) + (1 - 0)*\log(1 - 0.9)) = - \log(0.1) = 3.32$ (Log base 2)

D) Let L1 = BCE(1, 0.1), L2 = BCE(0, 0.2), L3 = BCE(0, 0.7), then

Total Loss(L) = (L1 + L2 + L3) / 3

= - (1 * log(0.1) + (1 - 0)*log(1 - 0.2) + (1 - 0) * log (1 - 0.7)) / 3

= - (log(0.1) + log(0.8) + log (0.3)) / 3

= - (-3.32 - 0.32 - 1.73) / 3

= 5.37 / 3

= 1.79 (Taking log base 2)

E) $W2 = W1 - \alpha((\partial Lbce / \partial W1) + 2*\lambda*W1)$, where W2 are the new weights, W1 are the old weights and *(del Lbce / del W1) is the gradient of Lbce with respect to W1.*

2

A) Shape of $W^{[2]} = K \times D^{a}$,

Shape of $B^{[2]} = K \times 1$,

Shape of output of hidden layer = $D^{A} \times m$

B)

C)

D)

Mo Tu We Th Fr Sa Su

Date / /

(b) $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \hat{y}_k(1-\hat{y}_k)$

(c) $\frac{\partial \hat{y}_k}{\partial z_i^{[2]}} = -\hat{y}_i \times y_k$

$(i \neq k)$

(d) $\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = \frac{\partial \sum y_i \log(\hat{y}_i)}{\partial z_i^{[2]}}$

$= \frac{\partial \sum y_i \log(\hat{y}_i)}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i^{[2]}}{\partial z_i^{[2]}}$

Mo Tu We Th Fr Sa Su

Date / /

$$\frac{\partial L}{\partial \hat{y}_i^{[1]}} = -\sum \left(\frac{y_k}{\hat{y}_k} \right)$$

$$\frac{\partial \hat{y}_i}{\partial z_i^{[2]}} = \hat{y}_k (1 - \hat{y}_k), \text{ if } i = k$$

$$= -\hat{y}_i \times \hat{y}_k \quad \text{otherwise}$$

Therefore,

If $i = k$,

$$\frac{\partial L}{\partial z_i^{[2]}} = -\sum \left(\frac{y_k}{\hat{y}_k} \right) \times \hat{y}_k (1 - \hat{y}_k)$$

else,

$$\frac{\partial L}{\partial z_j^{[2]}} = -\sum \left(\frac{y_k}{\hat{y}_k} \right) \times (-\hat{y}_i \times \hat{y}_k)$$

E) When we use softmax, we are exponentiating all the raw values present, and then take there sum as the denominator, thus effectively getting probabilities. The problem arises, when the number that we are dealing with is very large, its exponentiation can thus cause an overflow. Similarly, when a number is very large and negative, it will become 0 because of limited precision digits, thus losing its probability which causes loss of information. This is why softmax is numerically unstable.

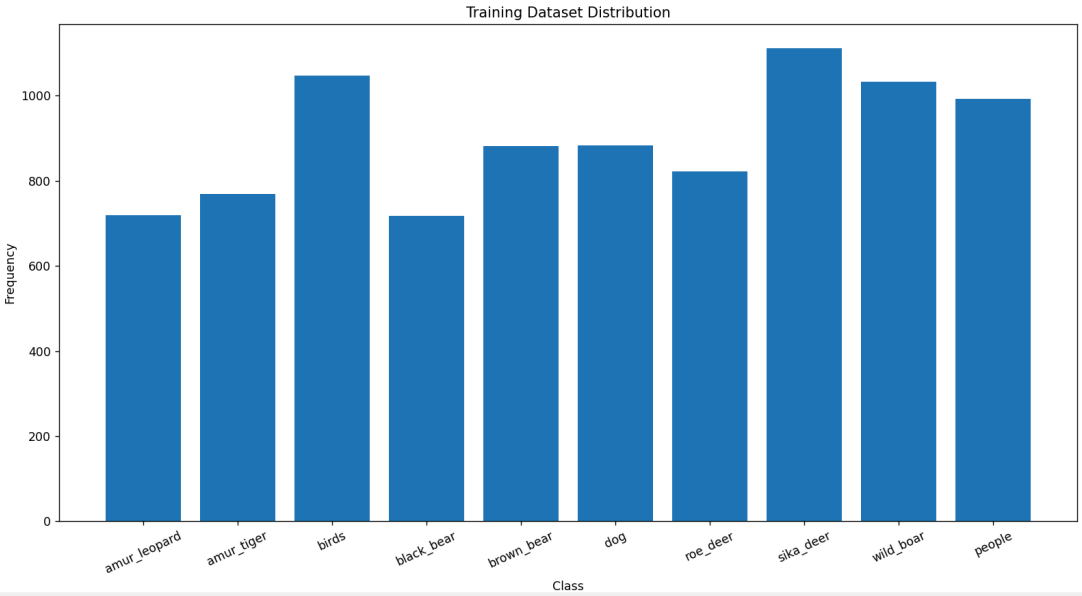
Question 2: Image Classification

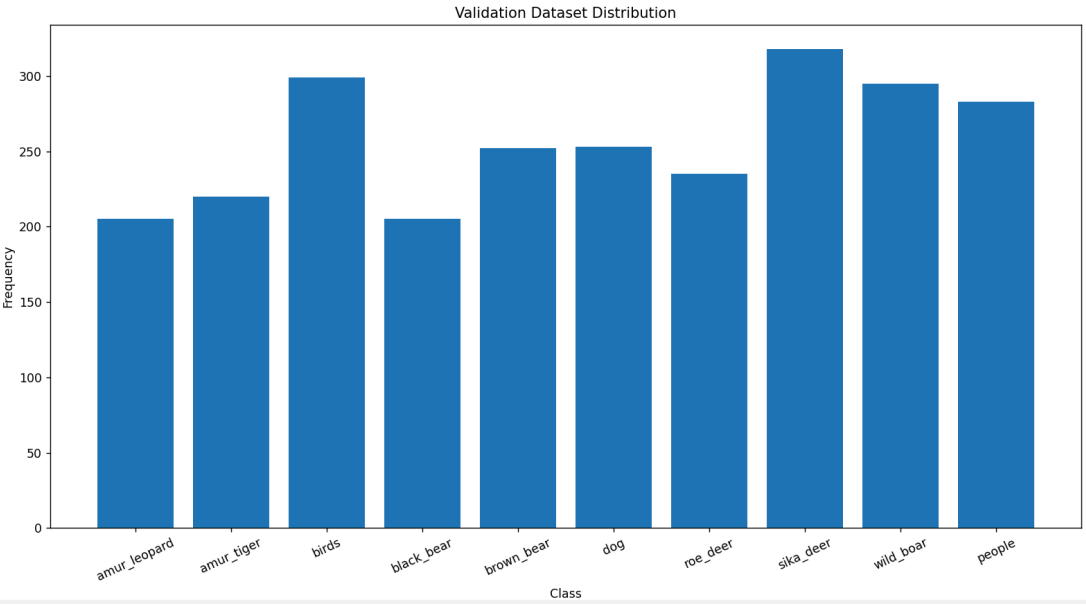
Code Overview:

1) First, image paths are extracted and stored in the Image_Path_List and their corresponding labels are also stored in labels
2) Then we do a stratified split of data for training, validation and testing.
3) We use the custom_dataset class, which takes the image paths and labels as input and returns the corresponding dataset.
4) We then, build a dataloader.
5) Then, we visualize the class distribution and plot a batch of image from the dataloader.
6) We define general functions for training and testing, which can be used by any model in the code.
7) We then, initialize and use models as required in the question, and alongside we initialize WandB as well to log the required metrics.
8) Feature extraction and the 2 subsequent functions below that are used for tSNE plots.
9) Certain code that was used during debugging has been commented out, you can ignore that and refer to some of the comments present in the code for explanation

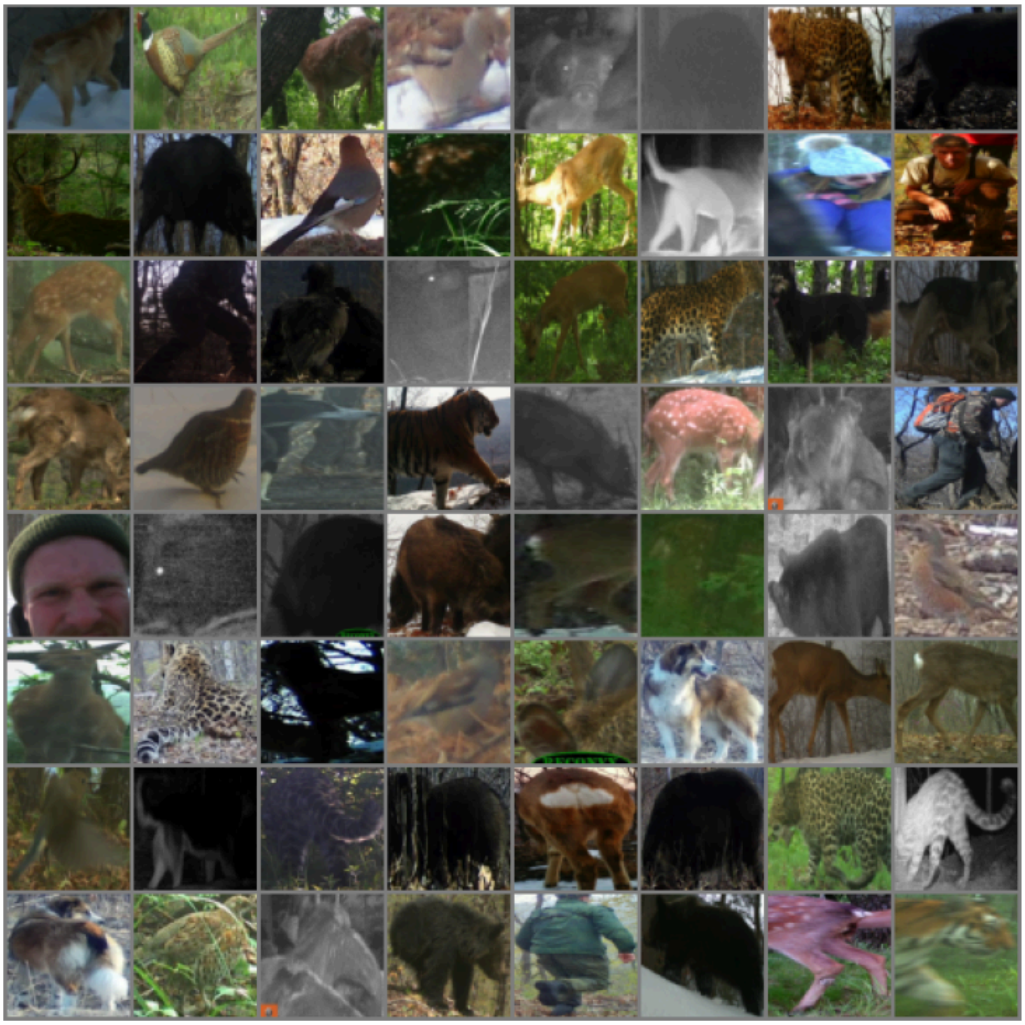
2.1

A) Class Distribution for training and validation datasets:



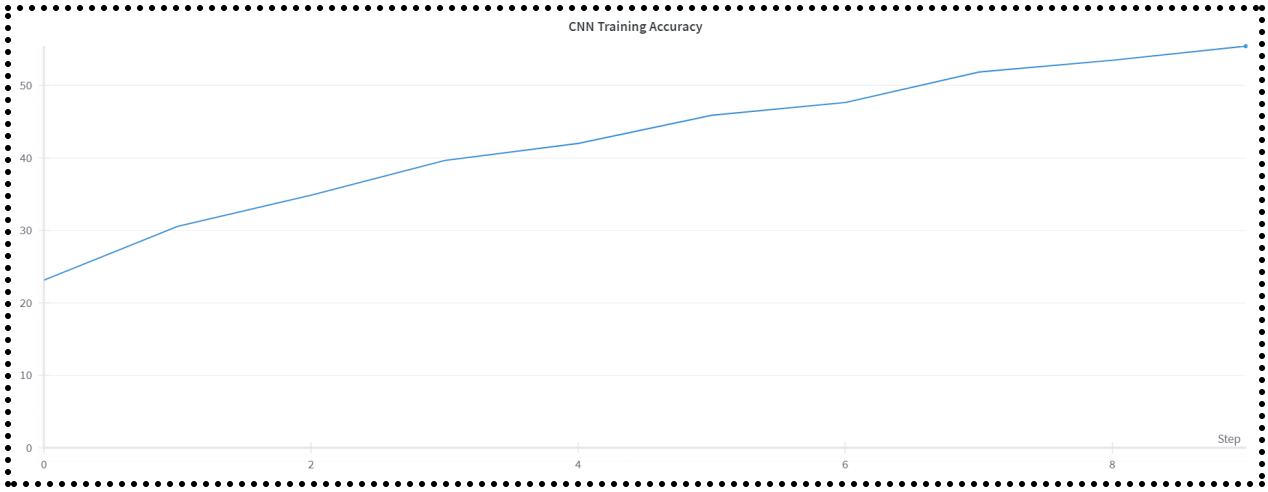


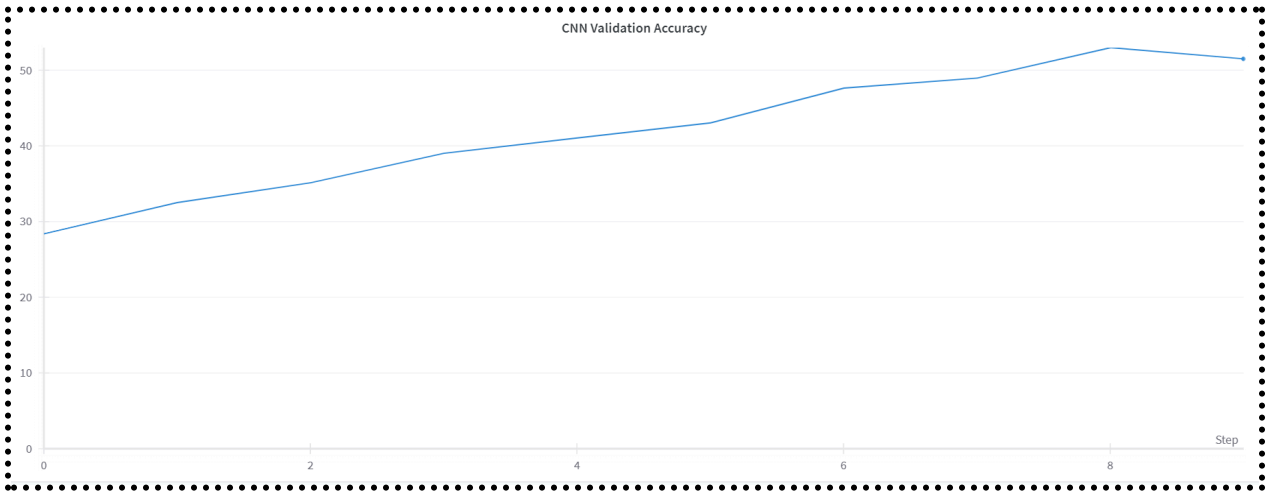
B) A batch in the training data
(Batch size-64, Number of Channels = 3, Pixels = 64 X 64)

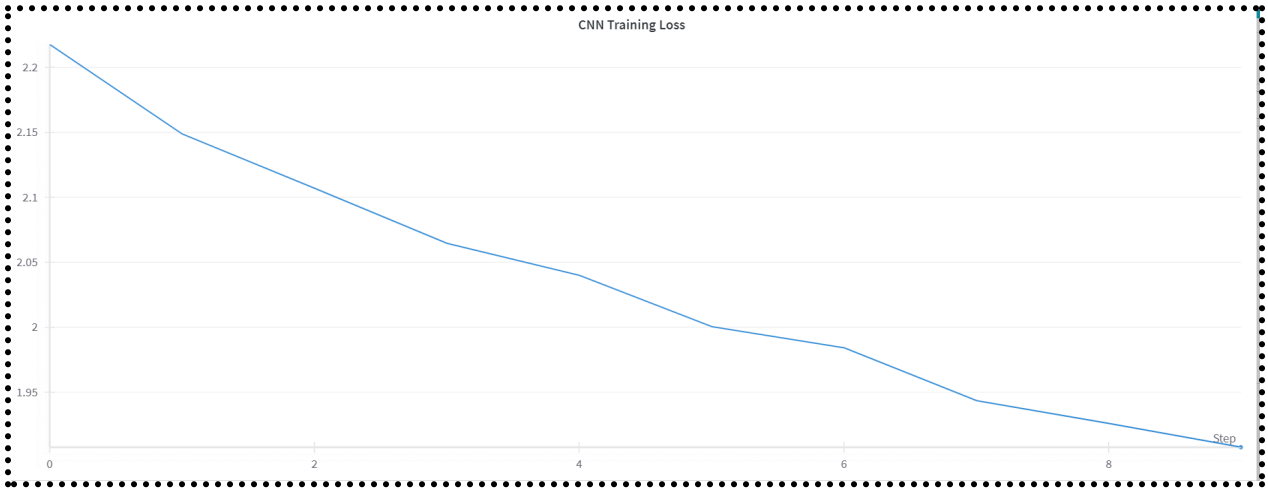


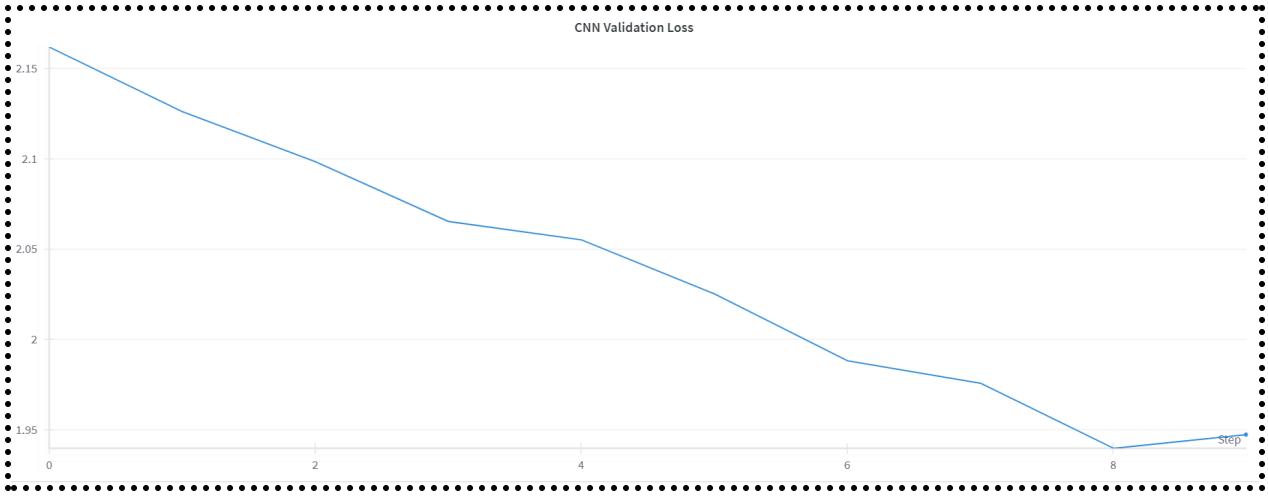
2.2

A) CNN Model Training and Validation Plots


CNN Training Accuracy
50
40
30
20
10
0
0
2
4
6
8
Step


CNN Validation Accuracy
50
40
30
20
10
0
0
2
4
6
8
Step


CNN Training Loss
2.2
2.15
2.1
2.05
2
1.95
0
2
4
6
8
Step

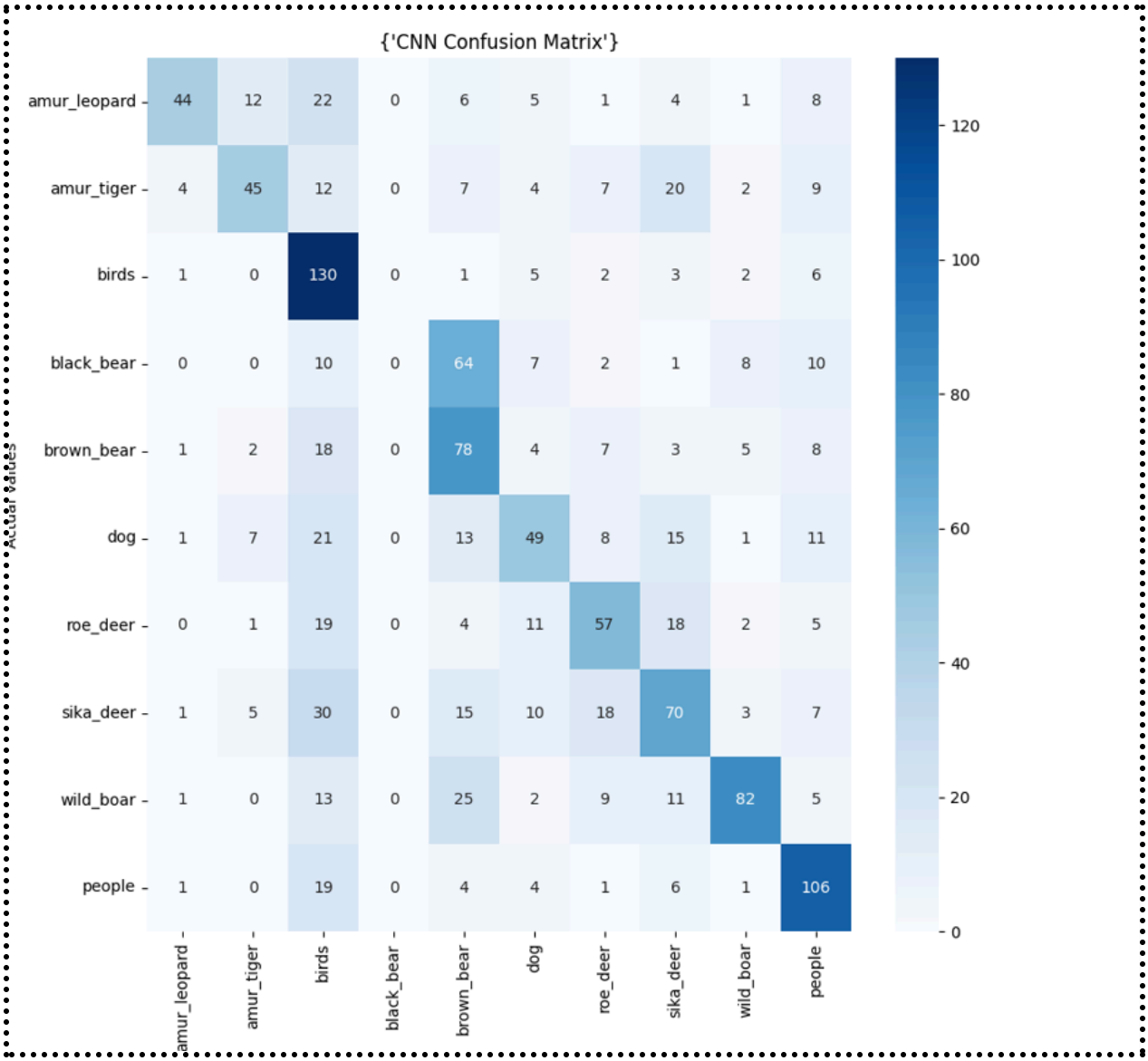


Yes, the model is overfitting. We can observe that at epoch 8, the validation accuracy is the highest, after that as the training accuracy continues to increase, validation accuracy decreases. Similarly, the validation loss is lowest at epoch 8 and increases after that while the training loss keeps decreasing. This shows that the model is overfitting. At the end, there is a large difference between training accuracy and validation accuracy, further justifying this conclusion.

B) CNN Model Testing Output

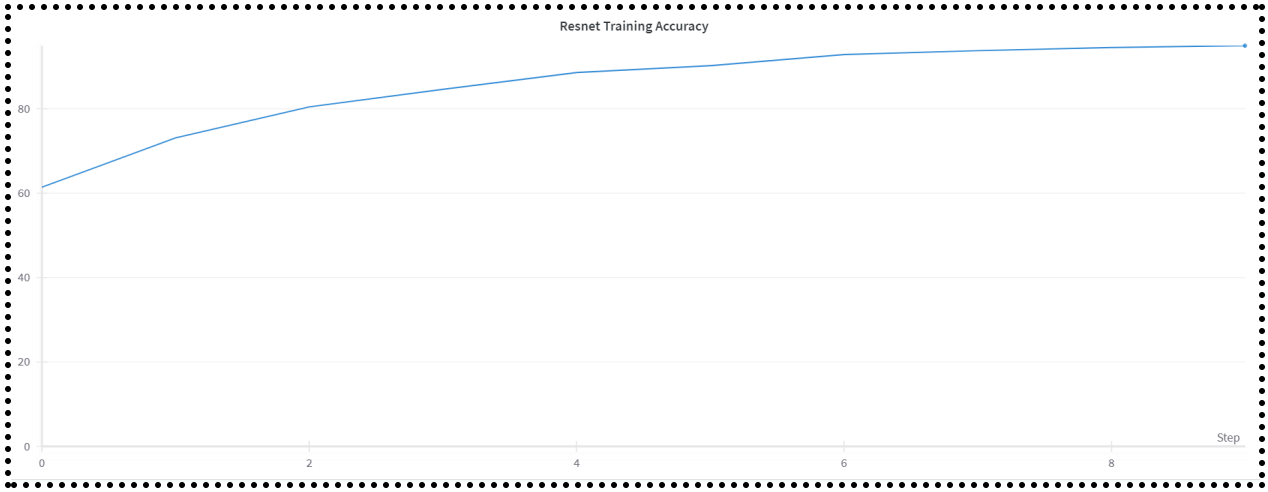
CNN Test Accuracy = 51.519875292283714
CNN Test F1 Score = 0.49249347703619045

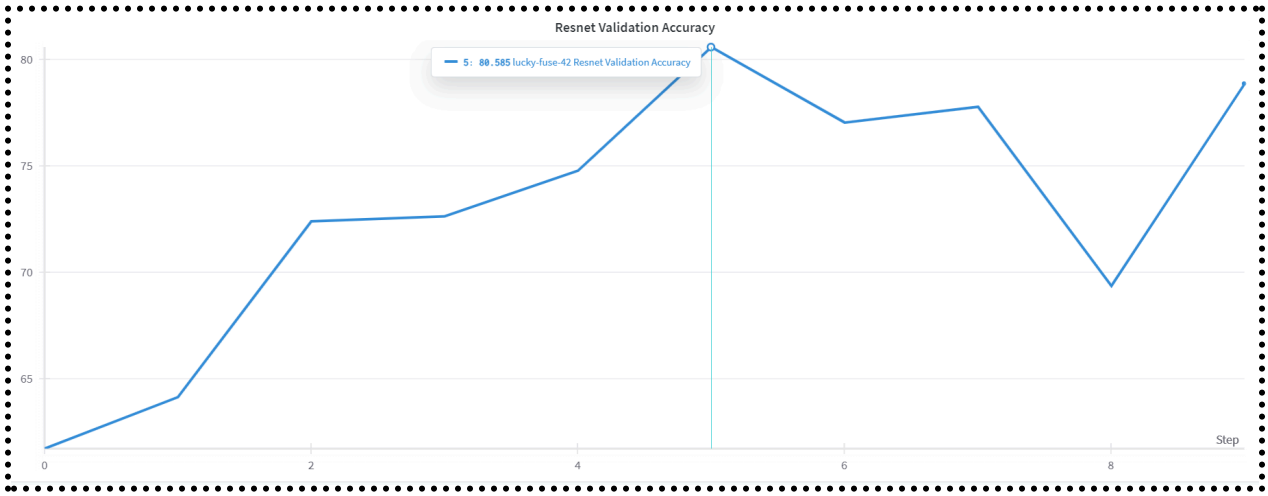
C) CNN Model confusion matrix

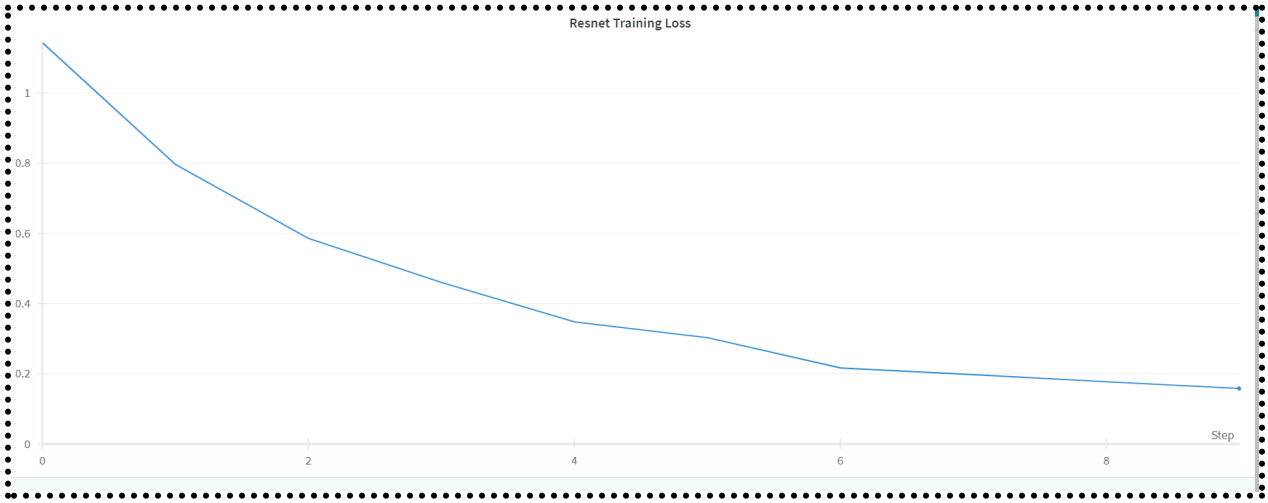


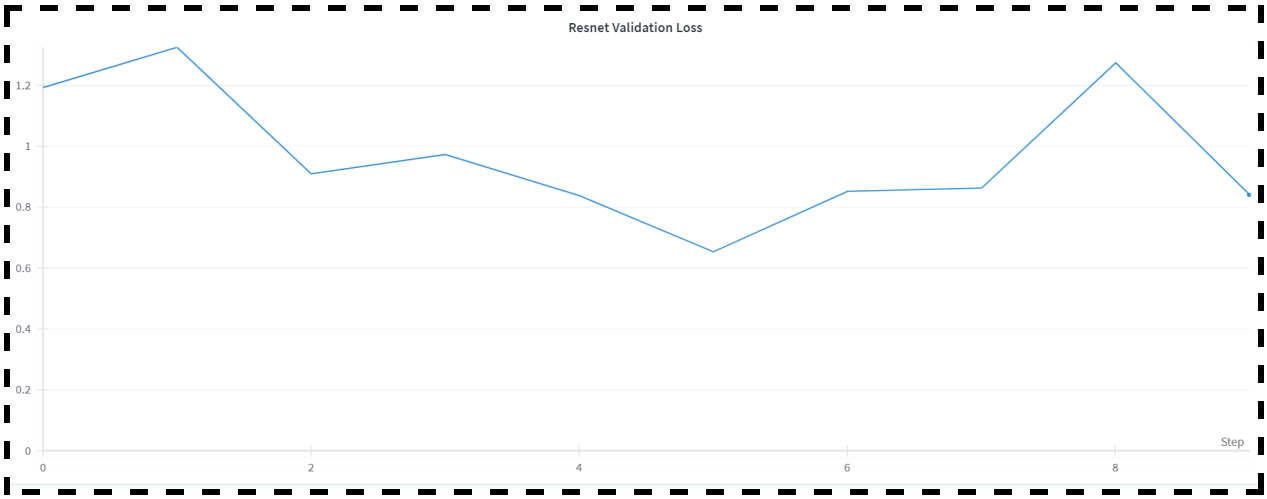
2.3

A) ResNet Model Training and Validation Plots


Resnet Training Accuracy
80
60
40
20
0
0
2
4
6
8
Step


Resnet Validation Accuracy
5: 80.585 lucky-fuse-42 Resnet Validation Accuracy
80
75
70
65
0
2
4
6
8
Step


Resnet Training Loss
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
Step

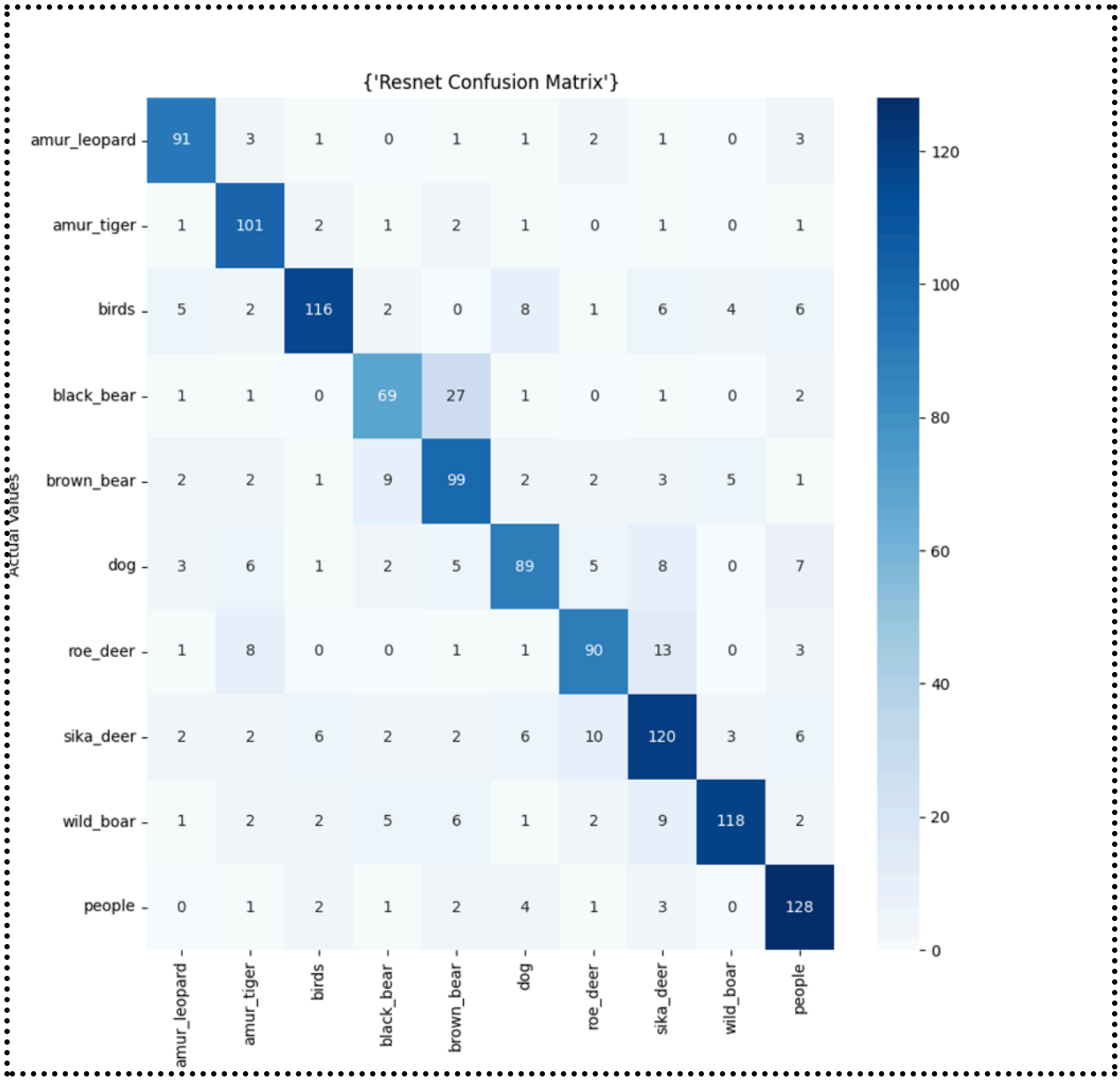


Yes, the model is overfitting. We can observe that at epoch 5, the validation accuracy is the highest, after that as the training accuracy continues to increase, validation accuracy decreases. Similarly, the validation loss is lowest at epoch 5 and increases after that while the training loss keeps decreasing. This shows that the model is overfitting. Another observation is the erratic and uneven behavior of the validation plots after epoch 5, indicating that the model is getting overfitted. There is also a large difference between the training and validation accuracy at the end, which also indicates overfitting.

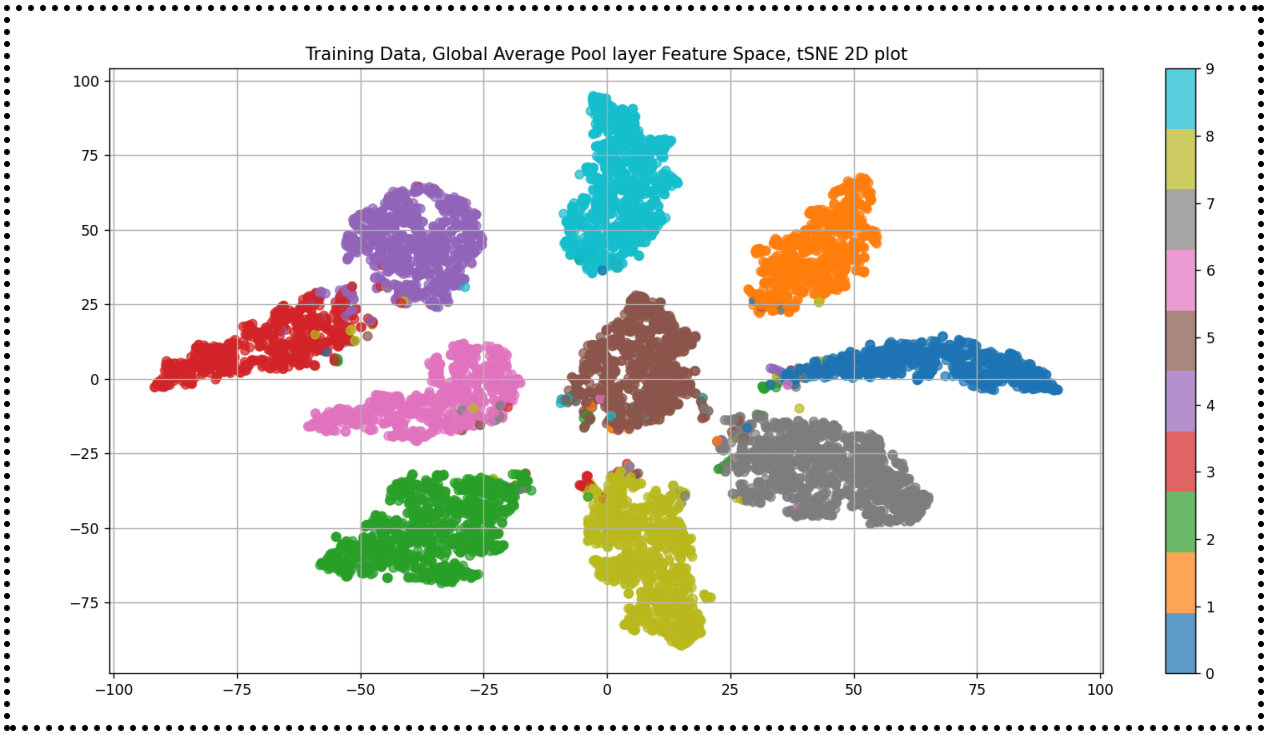
The overfitting however is less than what we observed in the case of CNN.

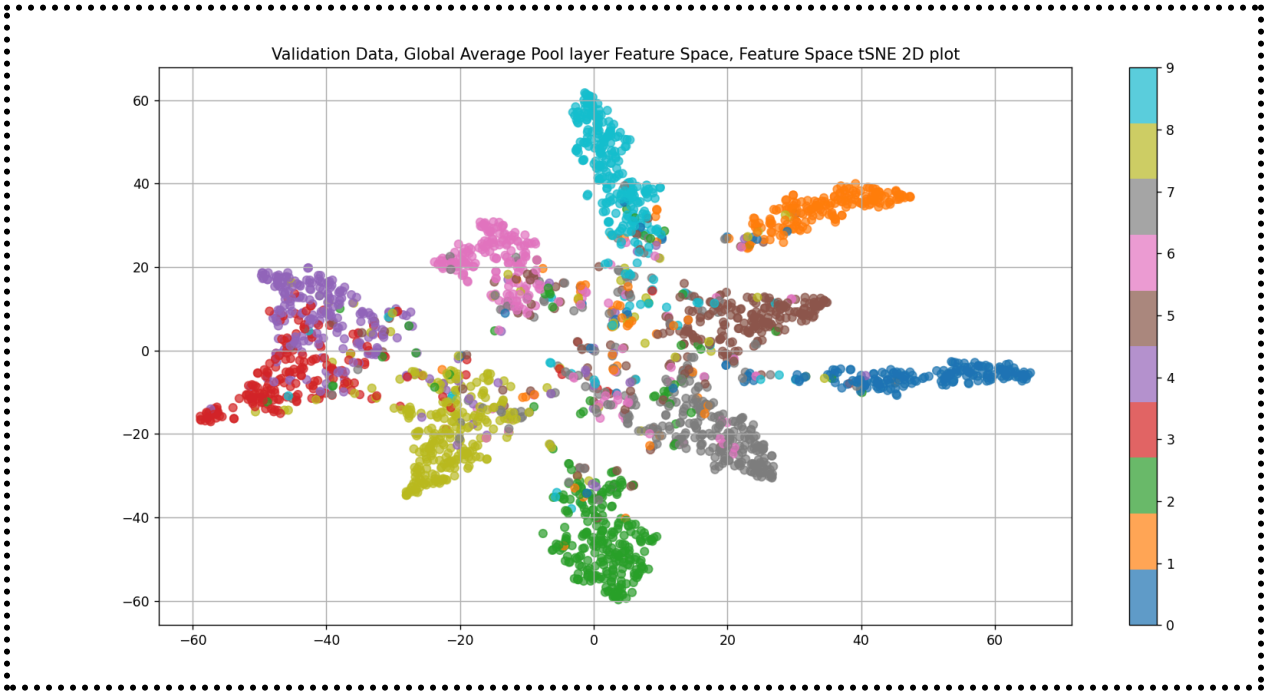
B) ResNet Model Testing Output
Resnet Test Accuracy = 79.57911145752144
Resnet Test F1 Score = 0.7961408665935681

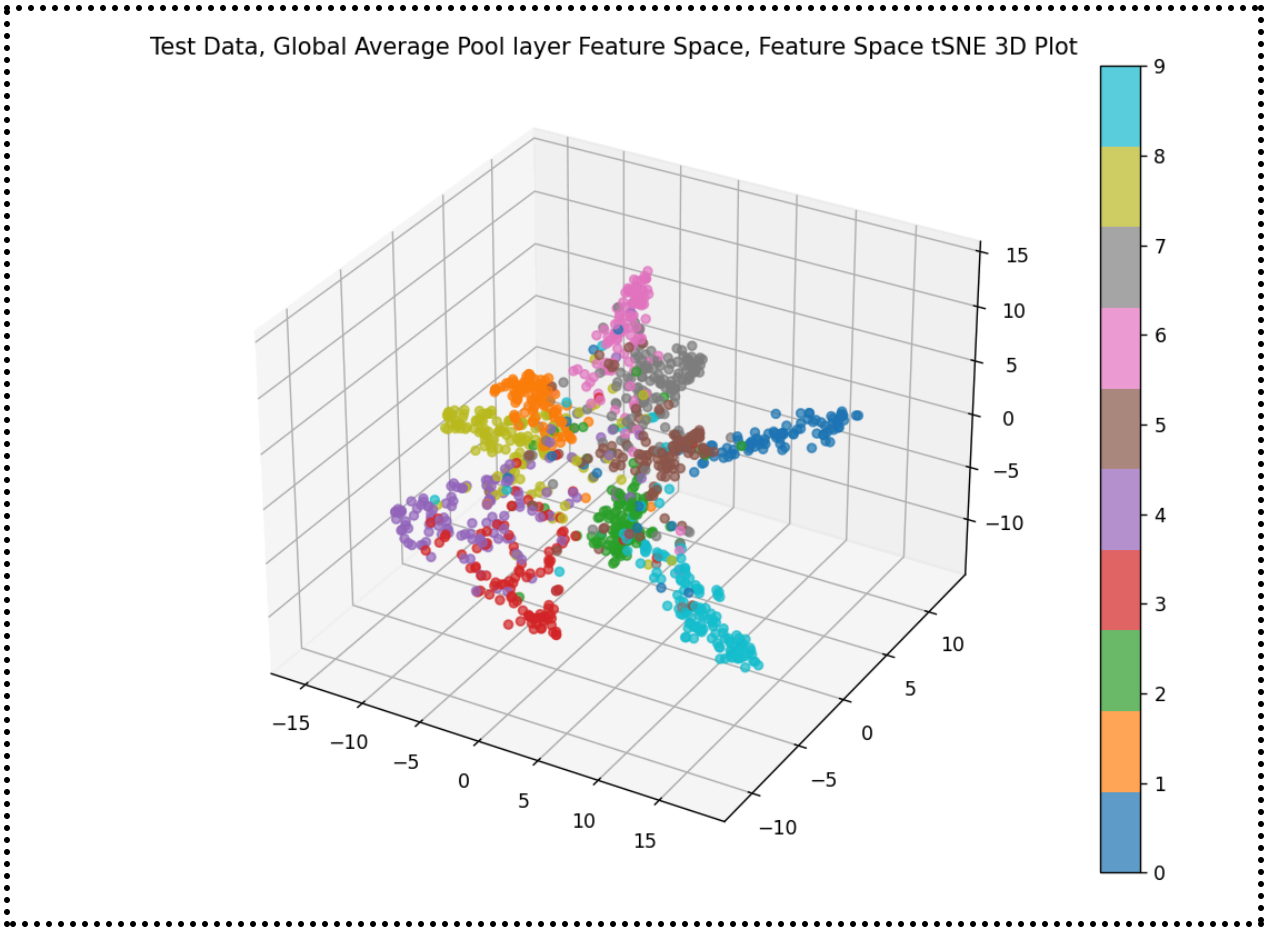
C) ResNet Model confusion matrix


{'Resnet Confusion Matrix'}
Actual Values
amur_leopard
amur_tiger
birds
black_bear
brown_bear
dog
roe_deer
sika_deer
wild_boar
people
91 3 1 0 1 1 2 1 0 3
1 101 2 1 2 1 0 1 0 1
5 2 116 2 0 8 1 6 4 6
1 1 0 69 27 1 0 1 0 2
2 2 1 9 99 2 2 3 5 1
3 6 1 2 5 89 5 8 0 7
1 8 0 0 1 1 90 13 0 3
2 2 6 2 2 6 10 120 3 6
1 2 2 5 6 1 2 9 118 2
0 1 2 1 2 4 1 3 0 128
120
100
80
60
40
20
0

D) TSNE Plots

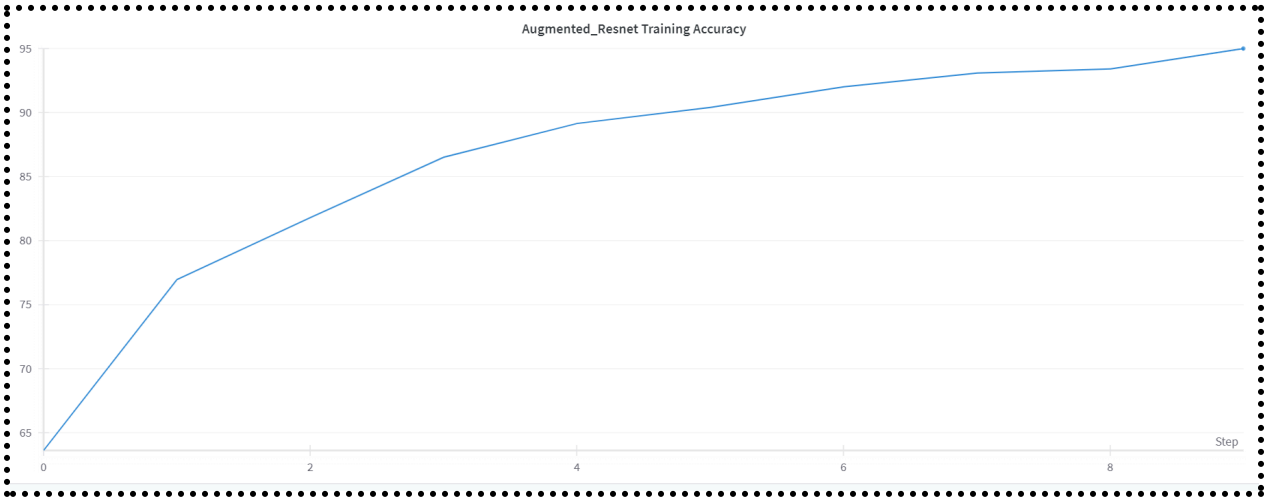

Training Data, Global Average Pool layer Feature Space, tSNE 2D plot
100
75
50
25
0
−25
−50
−75
−100
−75
−50
−25
0
25
50
75
100
9
8
7
6
5
4
3
2
1
0

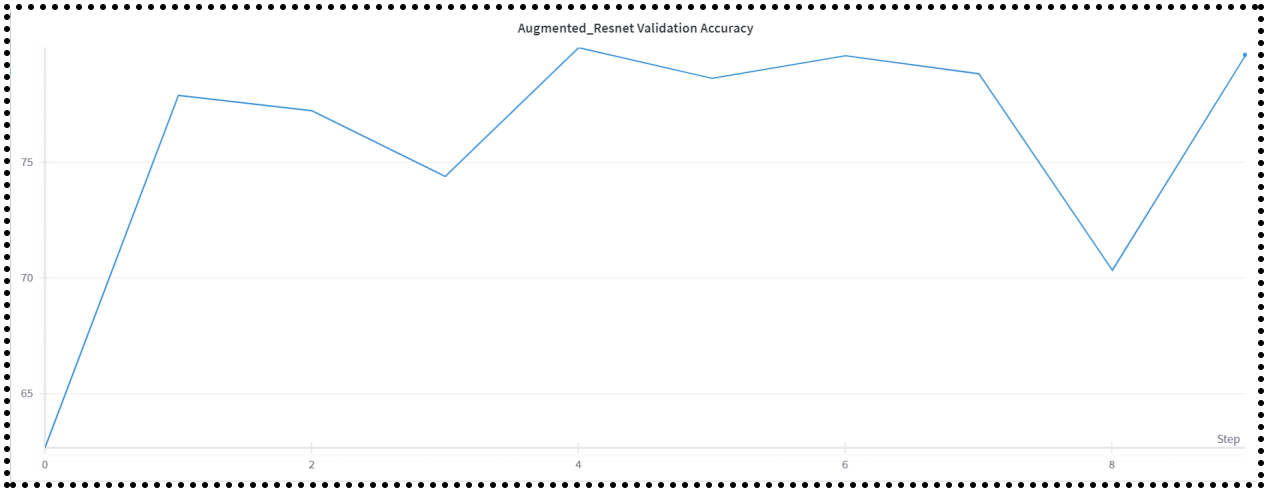

Validation Data, Global Average Pool layer Feature Space, Feature Space tSNE 2D plot
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
9
8
7
6
5
4
3
2
1
0

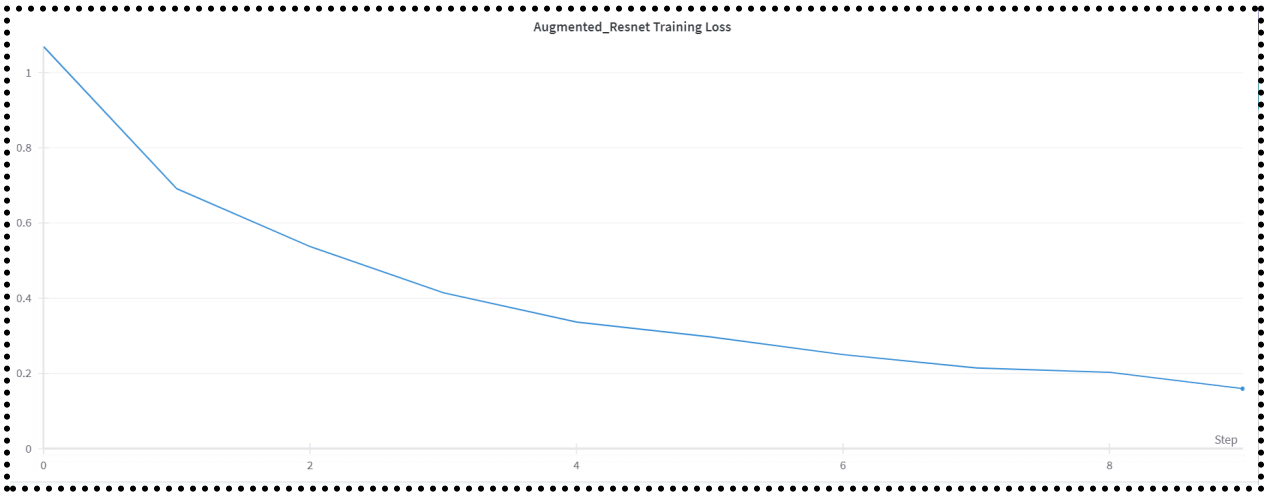


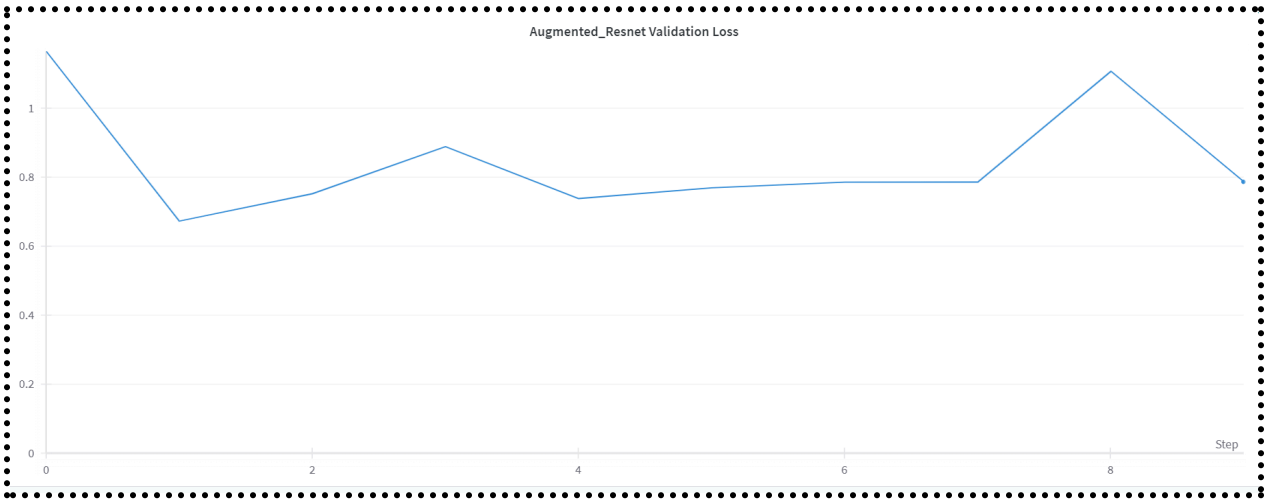
2.4

A) ResNet Augmented Model Training and Validation Plots









This model is also overfitting. This can be observed with the decrease in validation accuracy after the 8th epoch.

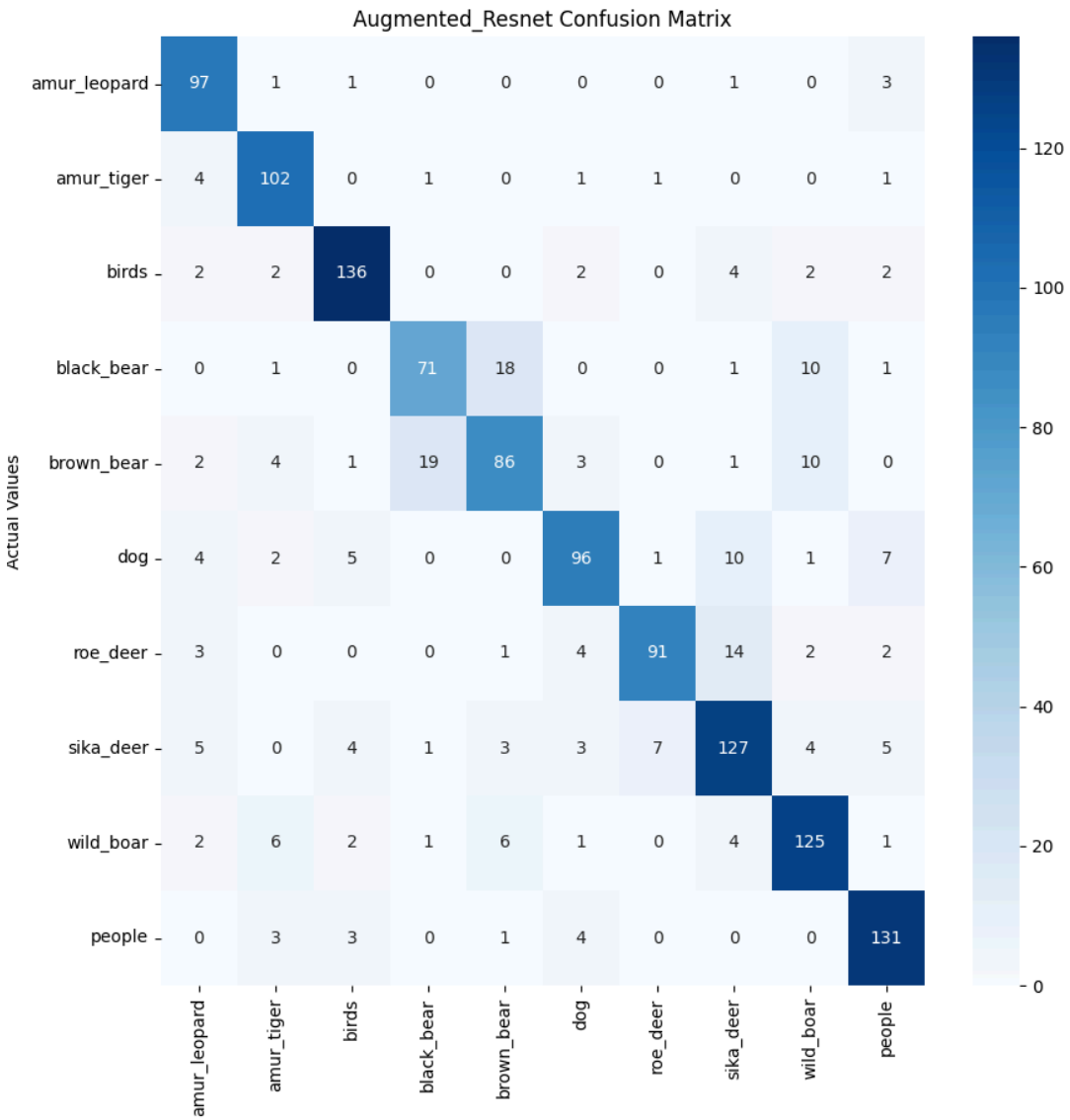
B) ResNet Augmented Model Testing Output:

C)

Augmented_Resnet Test Accuracy = 82.77474668745128

Augmented_Resnet Test F1 Score = 0.826054064753971

D) Resnet Augmented Model Confusion Matrix



2.6 All three models experience overfitting. However, overfitting is most severe in the CNN model, much less severe for the ResNet18 model and the least for the augmented ResNet model. This is based on the training and validation plots.

The test accuracies follow the same order. The test accuracy is highest for the augmented ResNet model, showing its high generalizability. It is slightly lower for the normal ResNet model and significantly lower for the CNN model.

We have the same observation for the F1 Scores as well. Where The augmented ResNet model has the highest F1 score, the normal ResNet model has a slightly lower F1 score and the CNN model has a significantly lower ResNet score.

The 3 confusion matrices also support the accuracy results as we can see significantly higher concentration at the diagonals for the Augmented ResNet model and the normal ResNet model as compared to the CNN model, indicating it's high accuracy and low false predictions.

Based on the 3 confusion matrices that we have, the CNN model faces difficulties in differentiating between black bears and brown bears, which is why it wrongly classifies all bears (both brown and black) as brown bears. The situation is significantly better in the case of the two ResNet models, where there are such cases of two classes being indistinguishable.

Question 3: Image Segmentation

Relevant File: Image_Segmentation.py
Code Overview: In this question, only the dataset has been downloaded, and processed and a dataloader has been created.