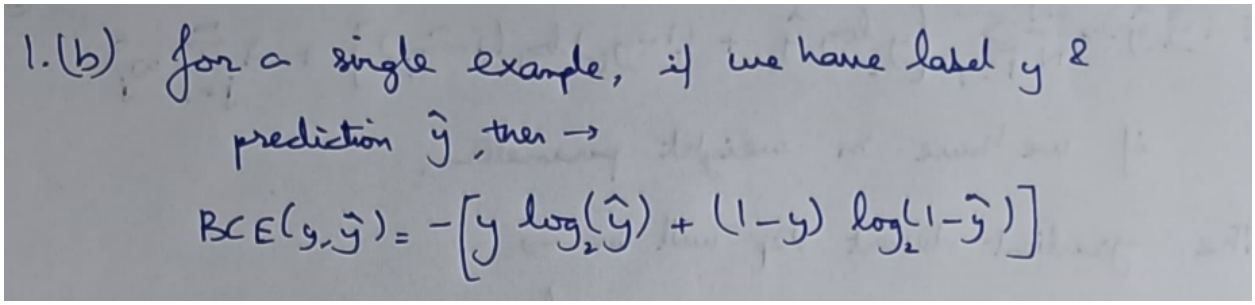
CSE344: Computer Vision Assignment 1 - Report

Name: Prakhar Gupta
Roll No: 2021550

Q1. Theory

1. (a)
   Classification of the sweetness of the papaya skin as sweet or not sweet is a binary classification task with two labels. Using mean square error (MSE) loss is not a good choice for a binary classification task. MSE loss is used for regression tasks, where we have to predict a continuous value and try to minimise the gap between the predicted and the actual continuous values. In binary classification, the output is categorical, and the values can be either 0 or 1, so the maximum value of MSE loss can be 1.
   So, the MSE loss function does not penalise the mistakes since if all the examples are wrongly classified, we get an MSE value of 1, and all correct classifications result in a value of 0. So, MSE loss does not penalise the mistakes enough, as the maximum value it will go for a binary classification task is 1. Instead, using binary cross-entropy loss for binary classification tasks makes more sense.

   (b) Binary cross entropy loss is better than MSE loss for binary classification tasks.



(c)

1.(c) for $\hat{y} = 0.9$, $y = 0$

$$BCE(y, \hat{y}) = -[0 \log_2(0.9) + (1-0)\log_2(1-0.9)]$$

$$= [0 \log_2(0.9) + 1 \log_2(0.1)]$$

$$= -[0 + \log_2(0.1)] = -\log_2(0.1) = +3.322$$

$$BCE(0, 0.9) = 3.322$$

(d)

1. (d) $Y = \{y_1, y_2, y_3\} = \{1, 0, 0\}$, $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3\} = \{0.1, 0.2, 0.7\}$

$$BCE = \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

$$\Rightarrow \frac{-1}{3}\begin{bmatrix} 1\log 0.1 + 0\log 0.9 \\ +0\log 0.2 + 1\log 0.8 \\ +0\log 0.7 + 1\log 0.3 \end{bmatrix} = -\frac{1}{3}\left[\log 0.1 + \log 0.8 + \log 0.3\right]$$

$$= -\frac{1}{3}\log\left(10^{-1} \times 8\times10^{-1} \times 3\times10^{-1}\right)$$

$$= -\frac{1}{3}\log(24\times10^{-3}) = -\frac{1}{3}\log_2(0.024) = 1.7935$$

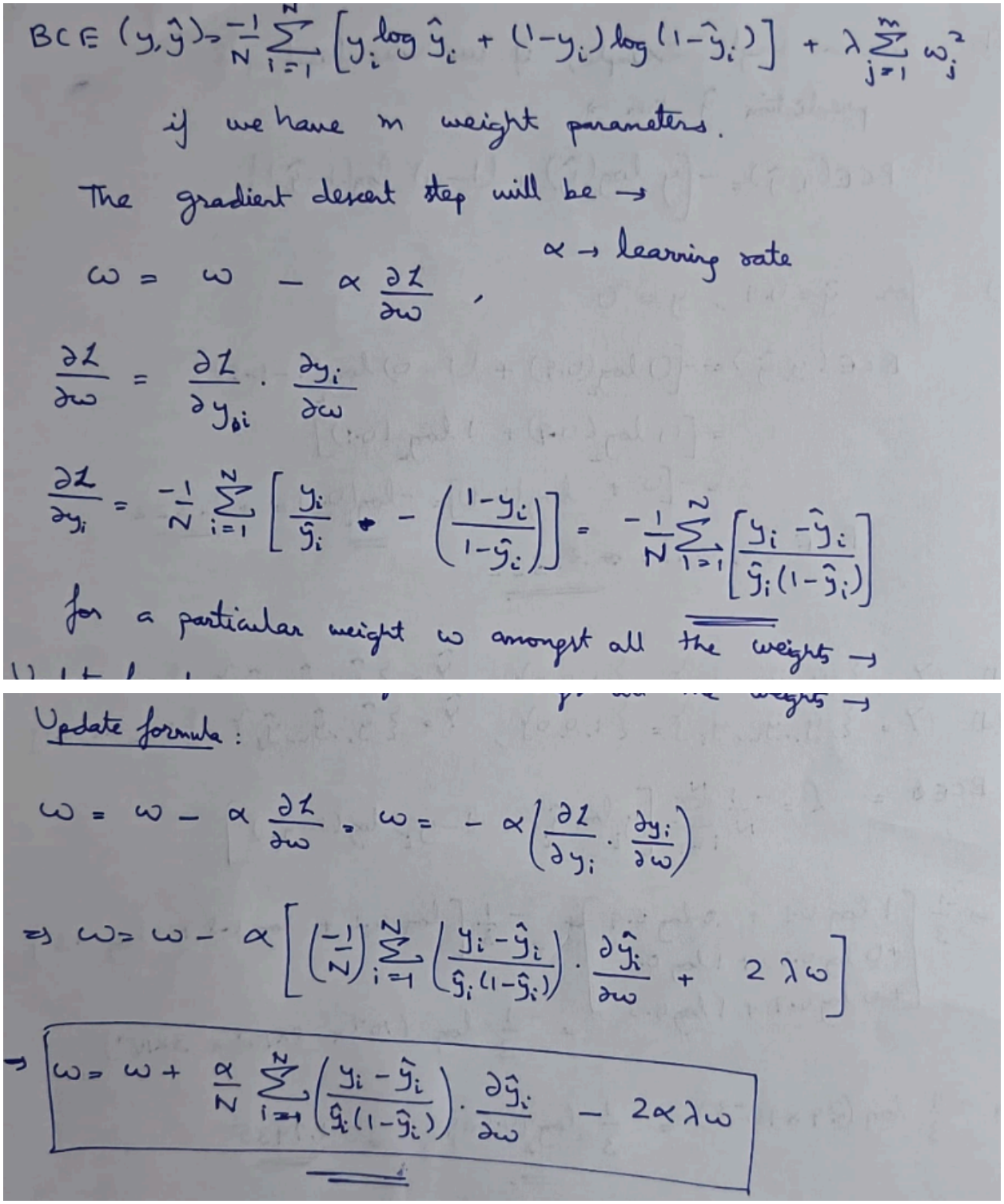
Total loss $\rightarrow$ $\mathcal{L} = 1.7935$

(e)

1.(e) $$BCE(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

And L2 Regularized BCE loss is $\rightarrow$

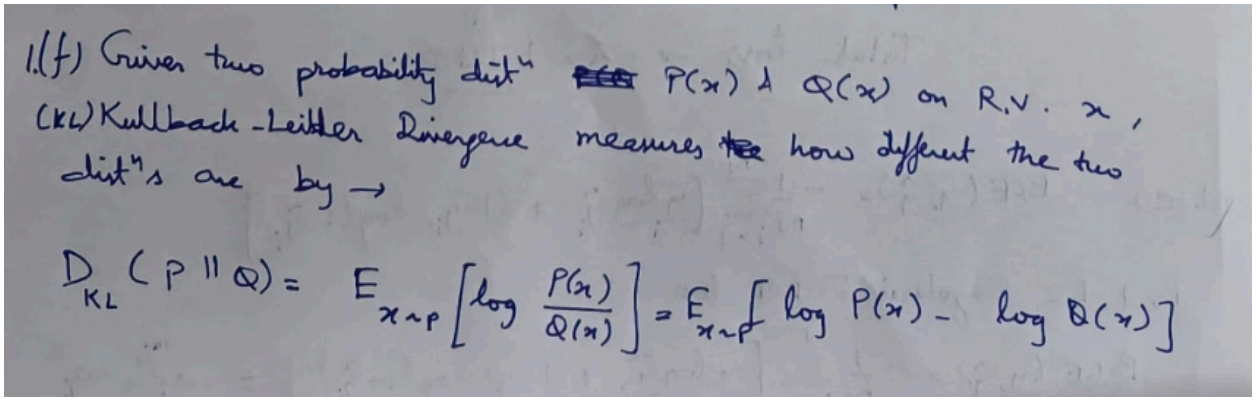
$$BCE(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right] + \lambda\sum_{j=1}^{m} w_j^2$$

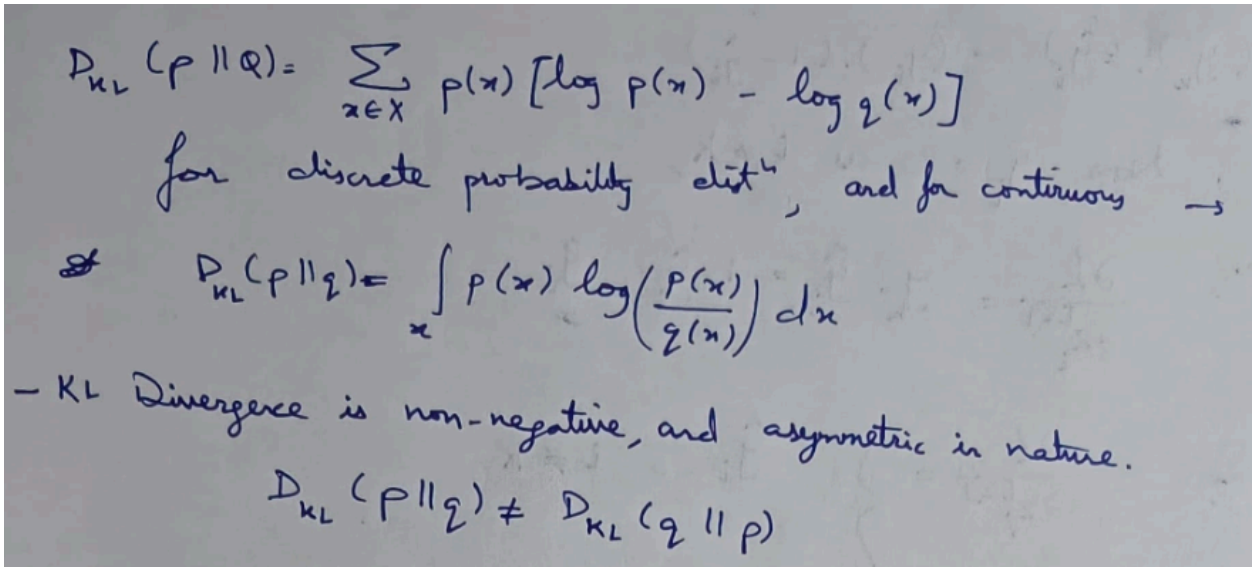


Model A with L2 regularisation will have weights that are closer to 0 as compared to model B’s weights. The absolute value/magnitude of model A's weights is smaller than model B's. This is because L2 regularisation forces the

model's weights to be pushed to a smaller value since model A is regularised. In contrast, model B isn't, so its weights will be smaller in magnitude as they are forced into lower values during backpropagation.

(f)





Cross Entropy:

The average no of bits required to represent one distribution using another distⁿ. Cross entropy is given by $H(P, Q)$

$$H(P, Q) = -E_{x \sim P}[\log q(x)] \quad -(1)$$

We know, $D_{KL}(P \| q) = E_{x \sim P}[\log p(x) - \log q(x)]$

$$\Rightarrow E_{x \sim P}[\log p(x)] - E_{x \sim P}[\log q(x)] \quad -(2)$$

That is, $D_{KL}(P \| q) = E_{x \sim P}[\log p(x)] + H(P, Q)$

So, $H(P, Q) = D_{KL}(P \| q) - E_{x \sim P}[\log p(x)]$

& we know, entropy of a distⁿ is given as –

$$H(P) = -E_{x \sim P}[\log p(x)] \quad -(3)$$

So, $\boxed{H(P, Q) = D_{KL}(P \| q) + H(P)}$

2.

Q2. 2-layer neural network for K-class classification → ①

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

$$a^{[1]} = \text{LeakyReLU}(z^{[1]}, \alpha = 0.01)$$

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

$$\hat{y} = \text{softmax}(z^{[2]})$$

Loss is given as → $$\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i)$$

Input shape of $x \rightarrow D_x \times 1$, One-hot encoded label form → $y \in \{0,1\}^*$

If the hidden layer has D_a nodes, then

$z^{[1]} \rightarrow D_a \times 1$, $a^{[1]} \rightarrow D_a \times 1$, $W^{[1]} \rightarrow D_a \times D_x$,

$b^{[1]} \rightarrow D_a \times 1$, $W^{[2]} \rightarrow D_K \times D_a$, $b^{[2]} \rightarrow D_K \times 1$,

$z^{[2]} \rightarrow D_K \times 1$, $\hat{y} \rightarrow D_K \times 1$

(a)

(a) Shape of $W^{[2]} \rightarrow D_K \times D_a$

Shape of $b^{[2]} \rightarrow D_K \times 1$

If we have vectorized over a batch of m samples, then $X \in \mathbb{R}^{D_x \times m}$, and so, the shape of the hidden layer →

$$(D_a \times D_x) \times (D_x \times m) \rightarrow D_a \times m$$

Shape of $z^{[1]}$ & $a^{[1]} \rightarrow D_a \times m$

(b)

(b) $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}}$, where z_k denotes the k^{th} element of vector z.

So, $\frac{\partial}{\partial z_k^{[2]}}\left(\hat{y}_k\right)$ & $\hat{y} = \text{softmax}\left(z^{[2]}\right)$

So, $\hat{y}_k = k^{th}$ element of softmax $= \frac{\exp(z_k^{[2]})}{\sum_{i=1}^{K} \exp(z_i^{[2]})}$ ②

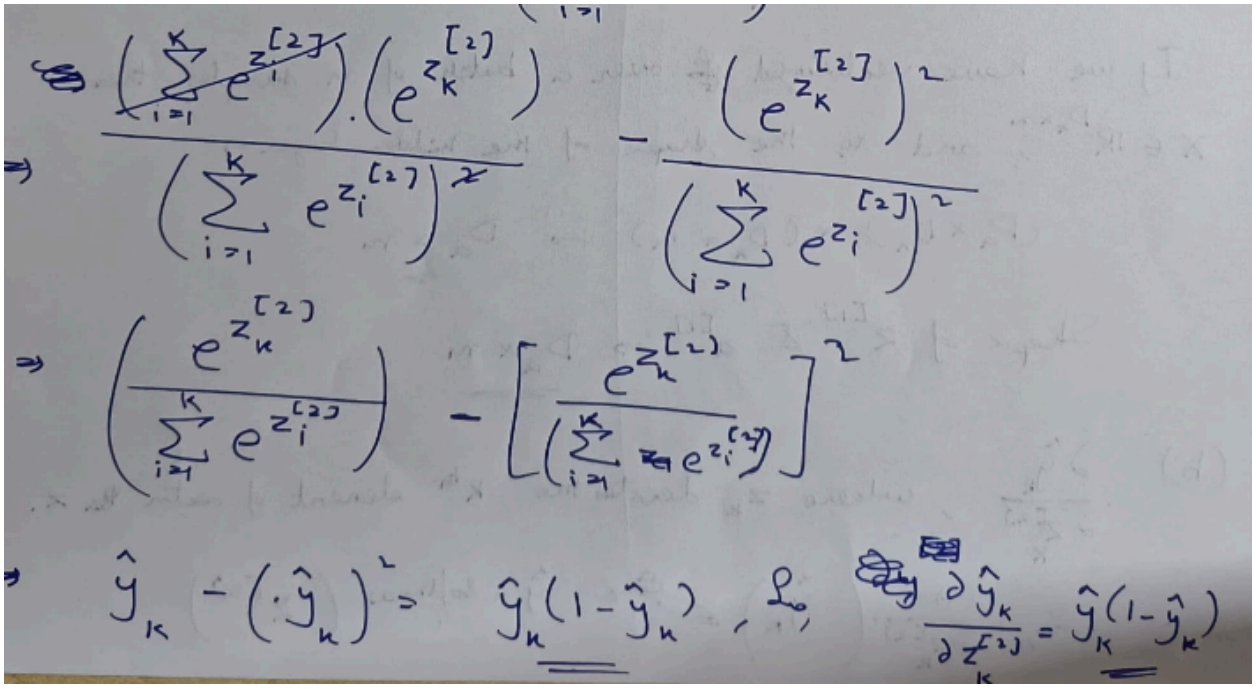
So, $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \frac{\partial}{\partial z_k^{[2]}}\left(\frac{e^{z_k^{[2]}}}{\sum_{i=1}^{K} e^{z_i^{[2]}}}\right)$

$$\Rightarrow \frac{\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)\left(\frac{\partial}{\partial z_k^{[2]}} e^{z_k^{[2]}}\right) - \left(e^{z_k^{[2]}}\right)\cdot\frac{\partial}{\partial z_k^{[2]}}\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)}{\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)^2}$$

Now, $\frac{\partial\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)}{\partial z_k^{[2]}} = \frac{\partial}{\partial z_k^{[2]}}\left(e^{z_1^{[2]}} + e^{z_2^{[2]}} + \ldots + e^{z_k^{[2]}} + \ldots + e^{z_K^{[2]}}\right)$

$\Rightarrow 0 + 0 + \ldots + 0 + e^{z_k^{[2]}} + \ldots + 0 = e^{z_k^{[2]}}$

So, $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \frac{\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)\left(e^{z_k^{[2]}}\right) - \left(e^{z_k^{[2]}}\right)\left(e^{z_k^{[2]}}\right)}{\left(\sum_{i=1}^{K} e^{z_i^{[2]}}\right)^2}$



(c)

(c) $\frac{\partial \hat{y}_k}{\partial z_i^{[2]}}$, for $i \neq k$ → $\hat{y}_k = \frac{e^{z_k^{[2]}}}{\sum_{j=1}^{K} e^{z_j^{[2]}}}$

$$\Rightarrow \frac{\partial}{\partial z_i^{[2]}} \left(\frac{e^{z_k^{[2]}}}{\sum_{j=1}^{K} e^{z_j^{[2]}}} \right)$$

$$\Rightarrow \frac{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)\left(\frac{\partial}{\partial z_i^{[2]}} e^{z_k^{[2]}}\right) - \left(e^{z_k^{[2]}}\right)\cdot\left(\frac{\partial}{\partial z_i^{[2]}}\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)\right)}{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)^2}$$

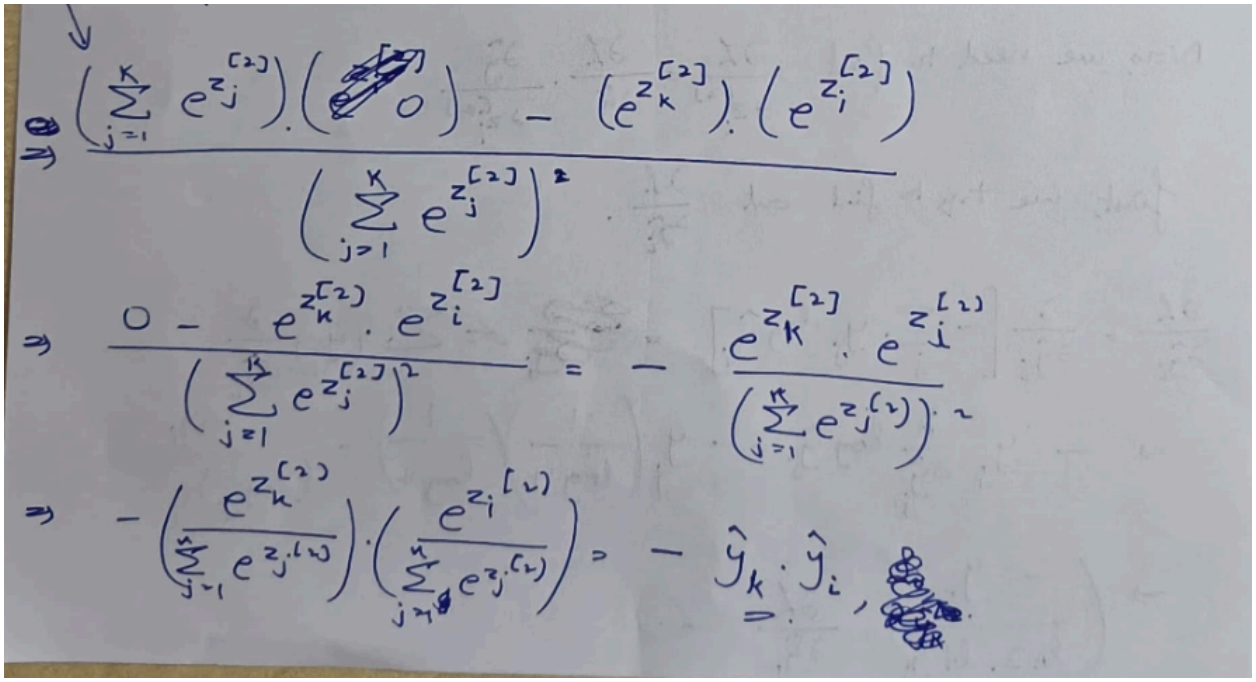
Now, $\frac{\partial}{\partial z_i^{[2]}} \cdot e^{z_k^{[2]}} = 0$ as $i \neq k$.

&

$$\frac{\partial}{\partial z_i^{[2]}}\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right) = \frac{\partial}{\partial z_i^{[2]}}\left(e^{z_1^{[2]}} + e^{z_2^{[2]}} + \dots + e^{z_i^{[2]}} + \dots + e^{z_K^{[2]}} \right)$$

$$\Rightarrow 0 + 0 + \dots + e^{z_i^{[2]}} + \dots + 0 = e^{z_i^{[2]}}$$

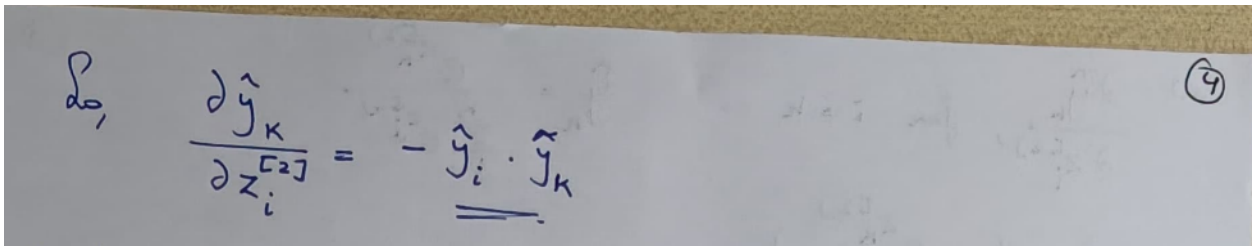
$$\Rightarrow \frac{\partial}{\partial z_i^{[2]}}\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right) = e^{z_i^{[2]}}$$



$$\frac{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)\left(0\right) - \left(e^{z_k^{[2]}}\right)\left(e^{z_i^{[2]}}\right)}{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)^2}$$

$$\Rightarrow \frac{0 - e^{z_k^{[2]}} \cdot e^{z_i^{[2]}}}{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)^2} = - \frac{e^{z_k^{[2]}} \cdot e^{z_i^{[2]}}}{\left(\sum_{j=1}^{K} e^{z_j^{[2]}}\right)^2}$$

$$\Rightarrow - \left(\frac{e^{z_k^{[2]}}}{\sum_{j=1}^{K} e^{z_j^{[2]}}}\right)\left(\frac{e^{z_i^{[2]}}}{\sum_{j=1}^{K} e^{z_j^{[2]}}}\right) = - \hat{y}_k \cdot \hat{y}_i$$



$$\text{So, } \frac{\partial \hat{y}_k}{\partial z_i^{[2]}} = - \hat{y}_i \cdot \hat{y}_k$$

(4)

(d)

(d) for ~~$\frac{\partial \mathcal{L}}{\partial z_k^{[2]}}$~~ y s.t. $y_k = 1$ & $y_j = 0$ for $j \neq k$.

$\therefore$ $y = \{0, 0, \ldots, 1, 0, \ldots, 0\}$

Here, $\mathcal{L} = -\sum_{j=1}^{K} y_j \log_e(\hat{y}_j)$

Now, $\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_i^{[2]}}$

We know, $\frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \hat{y}_k \cdot (1 - \hat{y}_k)$

And, $\frac{\partial \hat{y}_k}{\partial z_i^{[2]}} = -\hat{y}_i \cdot \hat{y}_k$ for $i \neq k$

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

Now, we need to find $\frac{\partial \mathcal{L}}{\partial z_i^{[2]}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_i^{[2]}}$

first, we try to find out $\frac{\partial \mathcal{L}}{\partial \hat{y}_i}$.

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}\left[-\sum_j y_j \log \hat{y}_j\right] =$$

$$\Rightarrow -y_j \frac{\partial}{\partial \hat{y}_j} \log \hat{y}_j = -y_j \left(\frac{1}{\log y_j}\right)\left(\frac{1}{\log 2}\right) =$$

$$\Rightarrow \left(\frac{-y_j}{\ln 2 \cdot \ln \hat{y}_j}\right) = \frac{\partial \mathcal{L}}{\partial \hat{y}_j} \quad -(1)$$

Now, when $i = k$,

$$\frac{\partial L}{\partial z_k^{[2]}} = \frac{\partial L}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_k^{[2]}} = \left(\frac{y_k}{\ln 2 \cdot \hat{y}_k}\right) \cdot \frac{\partial}{\partial z_k^{[2]}}$$

$$\Rightarrow \sum_{j=1}^{K} \frac{\partial L}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial z_k^{[2]}} \Rightarrow \sum_{j=1}^{K-1} \left(\frac{-y_j}{\hat{y}_j}\right) \cdot \left(\frac{\partial \hat{y}_j}{\partial z_k^{[2]}}\right) + \left(\frac{-y_k}{\hat{y}_k}\right) \cdot \hat{y}_k \cdot (1 - \hat{y}_k)$$

$$\Rightarrow \underbrace{\sum_{j=1}^{K} \left(\frac{-y_j}{\hat{y}_j}\right) \cdot (-\hat{y}_j \cdot \hat{y}_k)}_{0} + \left(\frac{-y_k}{\hat{y}_k}\right)(\hat{y}_k)(1 - \hat{y}_k)$$

& $y_j = 0$ for $j = 1, \ldots, (K-1)$, $y_k = 1$.

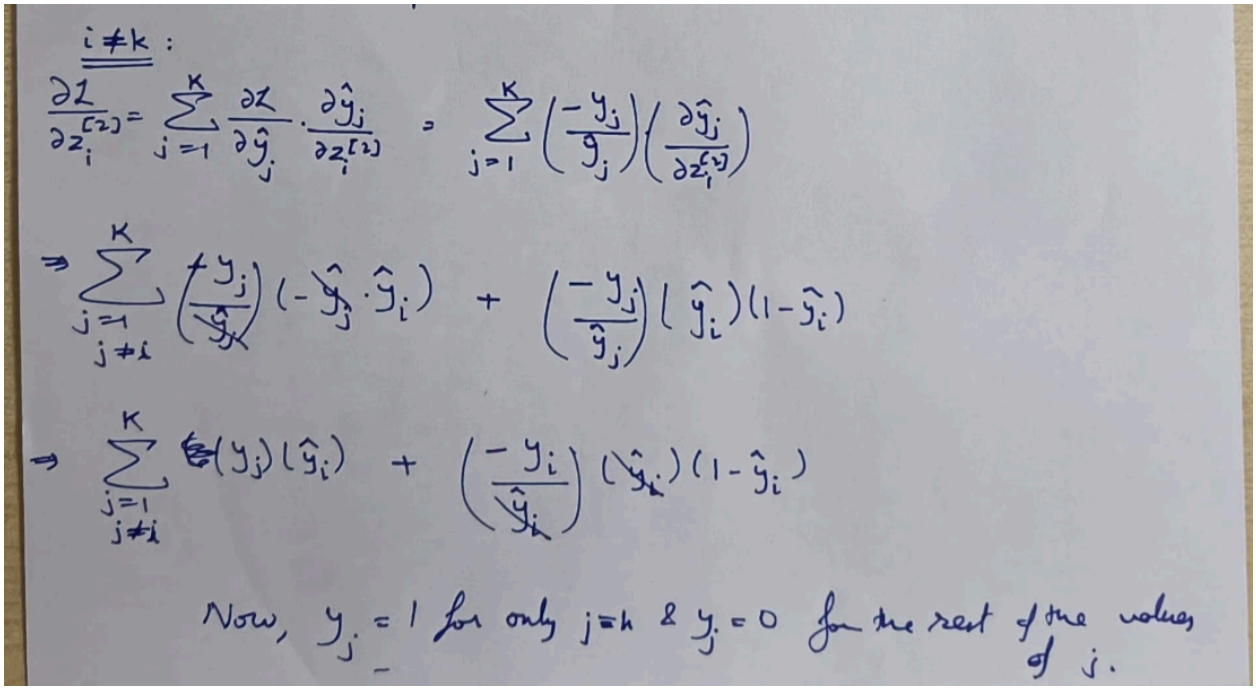
So,

$$\Rightarrow 0 + -(y_k) \cdot (1 - \hat{y}_k) = (-1)(1 - \hat{y}_k)$$

$$= \hat{y}_k - 1 \Rightarrow$$

So $\frac{\partial L}{\partial z_k^{[2]}} = \hat{y}_k - 1$

or, $\frac{\partial L}{\partial z_i^{[2]}} = \hat{y}_k - 1$, when $i = k$



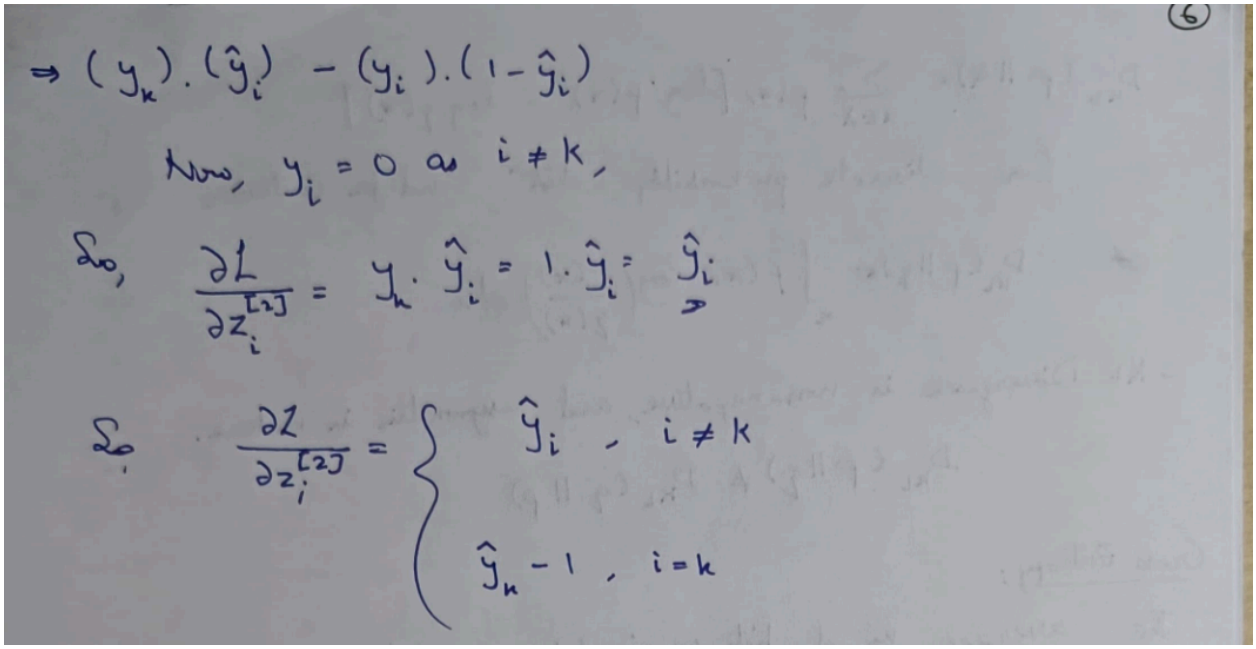
$i \neq k$:

$$\frac{\partial L}{\partial z_i^{[2]}} = \sum_{j=1}^{K} \frac{\partial L}{\partial \hat{y}_j} \cdot \frac{\partial \hat{y}_j}{\partial z_i^{[2]}} = \sum_{j=1}^{K} \left(\frac{-y_j}{\hat{y}_j}\right)\left(\frac{\partial \hat{y}_j}{\partial z_i^{[2]}}\right)$$

$$\Rightarrow \sum_{\substack{j=1 \\ j \neq i}}^{K} \left(\frac{-y_j}{\hat{y}_j}\right)(-\hat{y}_j \cdot \hat{y}_i) + \left(\frac{-y_i}{\hat{y}_i}\right)(\hat{y}_i)(1-\hat{y}_i)$$

$$\Rightarrow \sum_{\substack{j=1 \\ j \neq i}}^{K} (y_j)(\hat{y}_i) + \left(\frac{-y_i}{\hat{y}_i}\right)(\hat{y}_i)(1-\hat{y}_i)$$

Now, $y_j = 1$ for only $j = k$ & $y_j = 0$ for the rest of the values of j.



(6)

$$\Rightarrow (y_k).(\hat{y}_i) - (y_i).(1-\hat{y}_i)$$

Now, $y_i = 0$ as $i \neq k$,

So, $\frac{\partial L}{\partial z_i^{[2]}} = y_k \cdot \hat{y}_i = 1 \cdot \hat{y}_i = \hat{y}_i$

So, $$\frac{\partial L}{\partial z_i^{[2]}} = \begin{cases} \hat{y}_i, & i \neq k \\ \hat{y}_k - 1, & i = k \end{cases}$$

(e) Since the softmax function is a function containing exponentiation and the exponential function is implemented multiple times in it, it is possible that there can be cases of numerical overflow when one of the values of the softmax function is very large such that it cannot be represented in a computer. Suppose one of the values of the layer is 1000. The softmax function will have to compute e^{1000}, which will exceed the highest number that can be represented on a

computer, leading to numerical overflow. Due to this numerical overflow, softmax will return NAN values (language agnostic), and this causes numerical instability. For the below original softmax function-

$$sm(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$$

If the highest input value is c, then we will subtract it from all the inputs and then compute softmax again -

$$sm(x_i) = \frac{e^{x_i - c}}{\sum_{j=1}^{d} e^{x_j - c}}$$

This shifting of all the inputs causes underflow in some cases. Still, underflow is easy to handle since the values just tend to 0 for an underflow, but then get returned to NAN for overflow cases. Using a series of steps, we can also prove that the softmax with the c value subtracted returns the same value theoretically as the original softmax function-

$$sm(x_i) = \frac{e^{x_i - c}}{\sum_{j=1}^{d} e^{x_j - c}}$$

$$sm(x_i) = \frac{e^{x_i} e^{-c}}{\sum_{j=1}^{d} e^{x_j} e^{-c}}$$

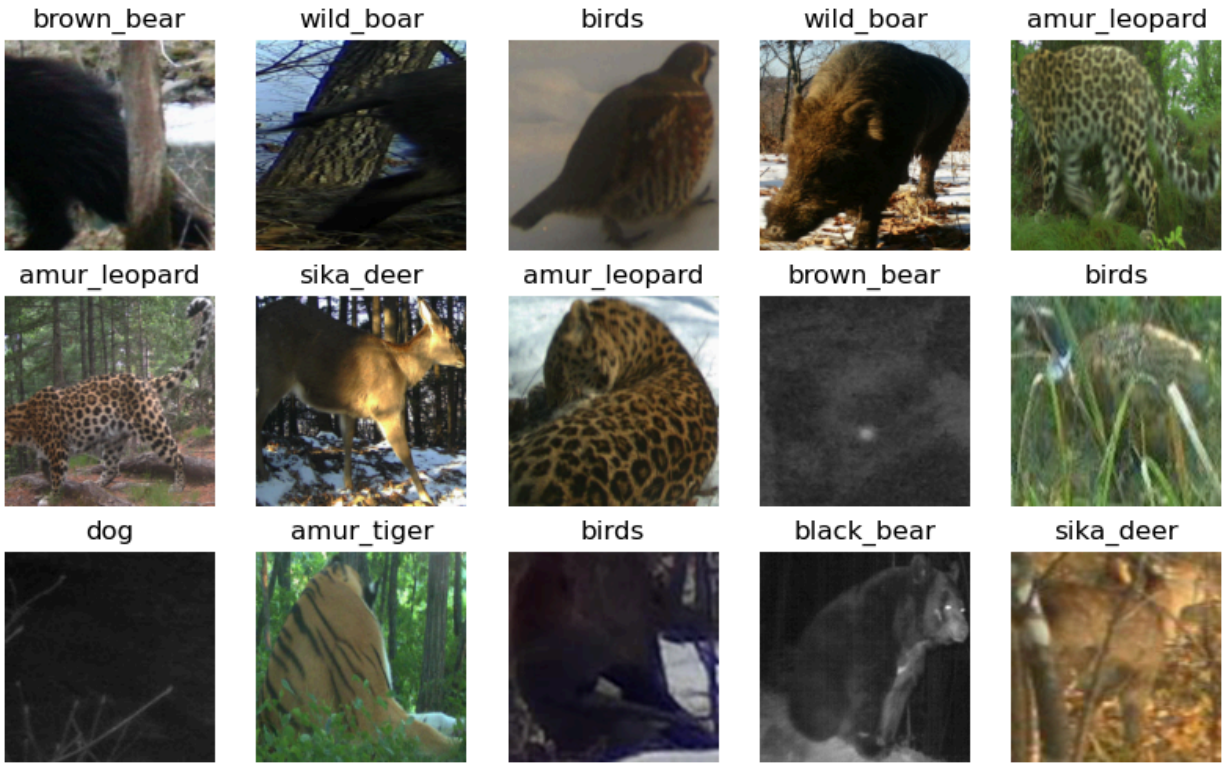
$$sm(x_i) = \frac{e^{x_i} e^{-c}}{e^{-c} \sum_{j=1}^{d} e^{x_j}}$$

$$sm(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$$

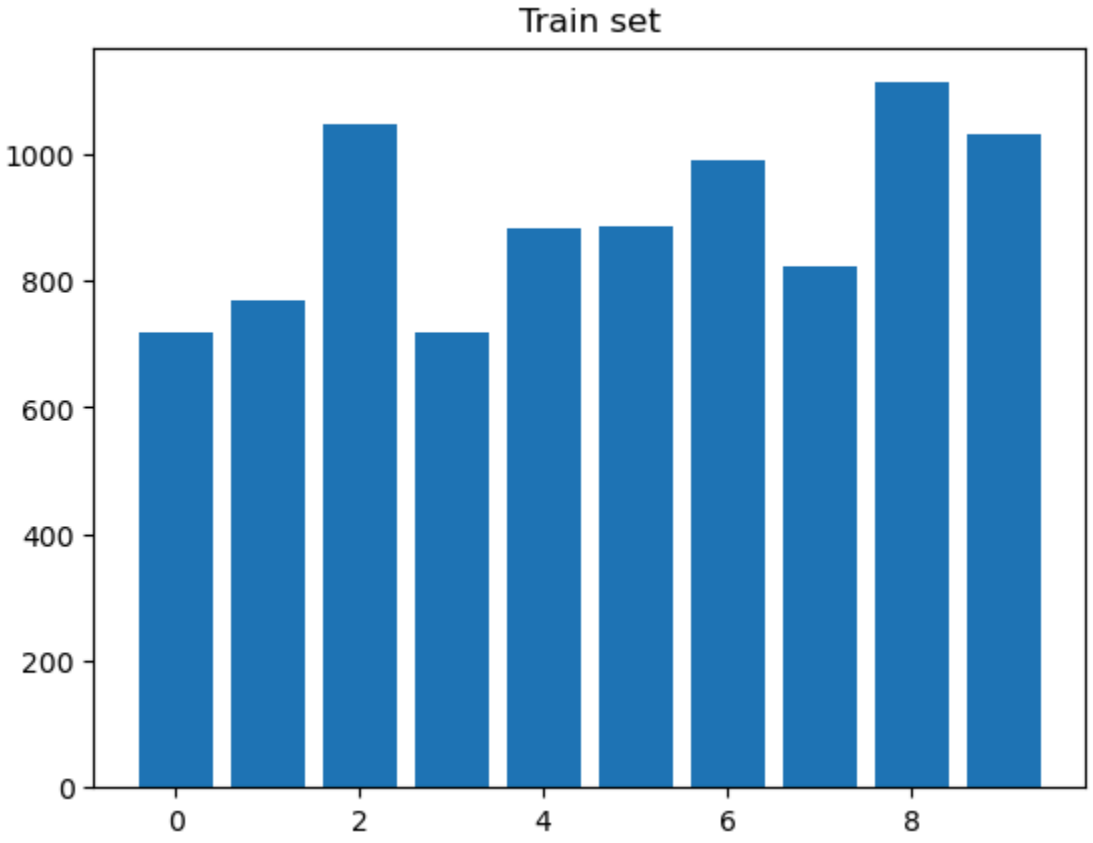
Citation:
https://ogunlao.github.io/2020/04/26/you_dont_really_know_softmax.html

Q2. Image Classification
2.1 (c) Following are some of the images of the different classes from the training data split of the Russian Wildlife Dataset:

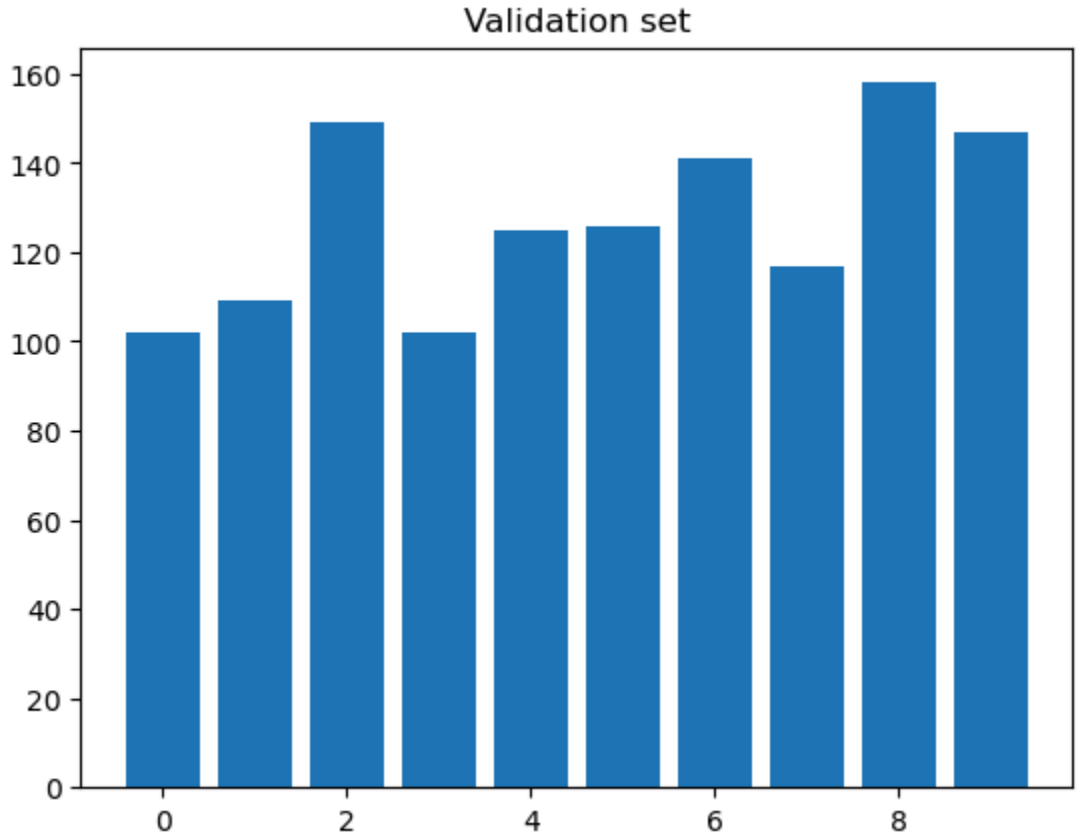
Following are some of the images of the different classes from the validation data split of the Russian Wildlife Dataset:



The distribution of the training data across classes is as follows -

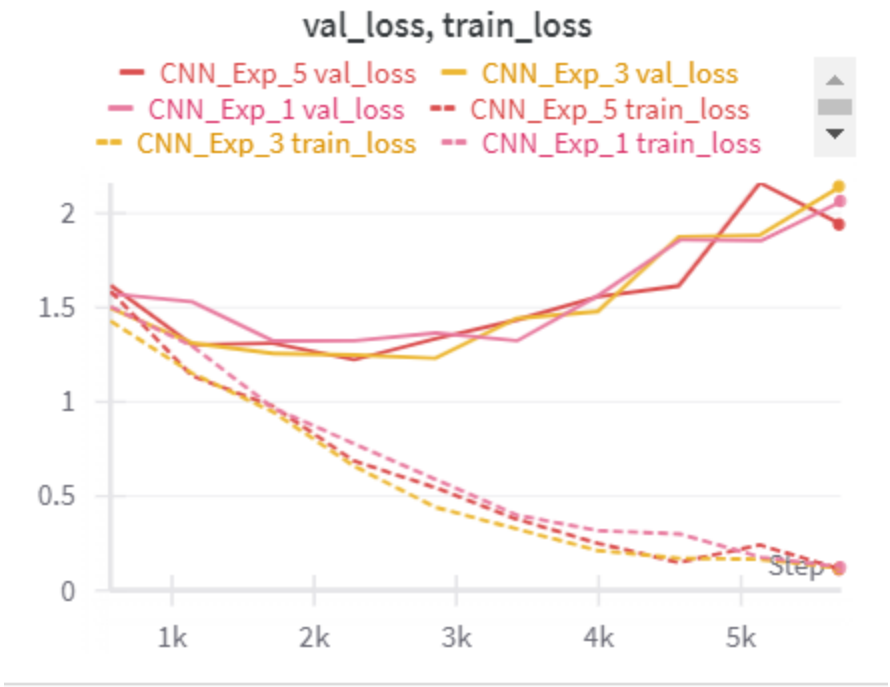


The distribution of the validation split across all classes is as follows-

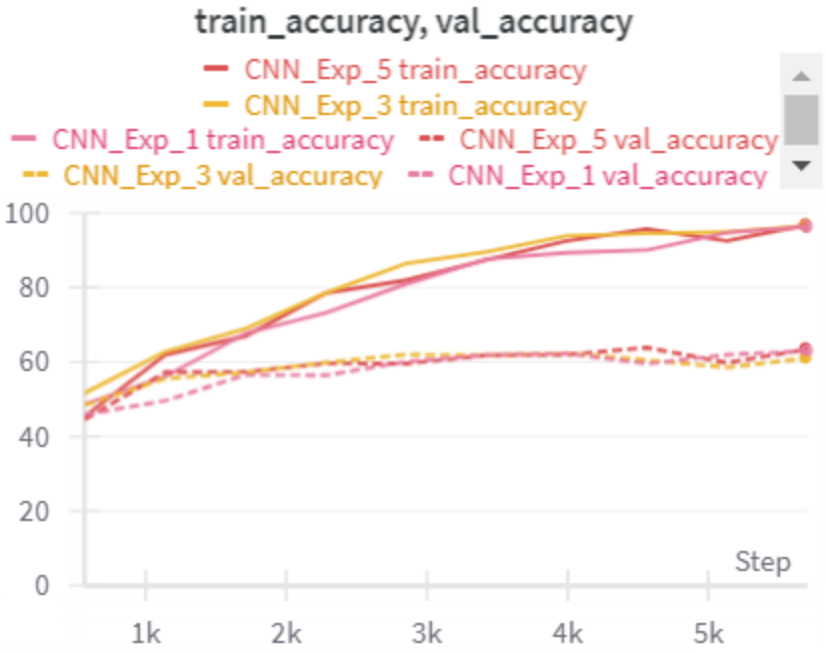


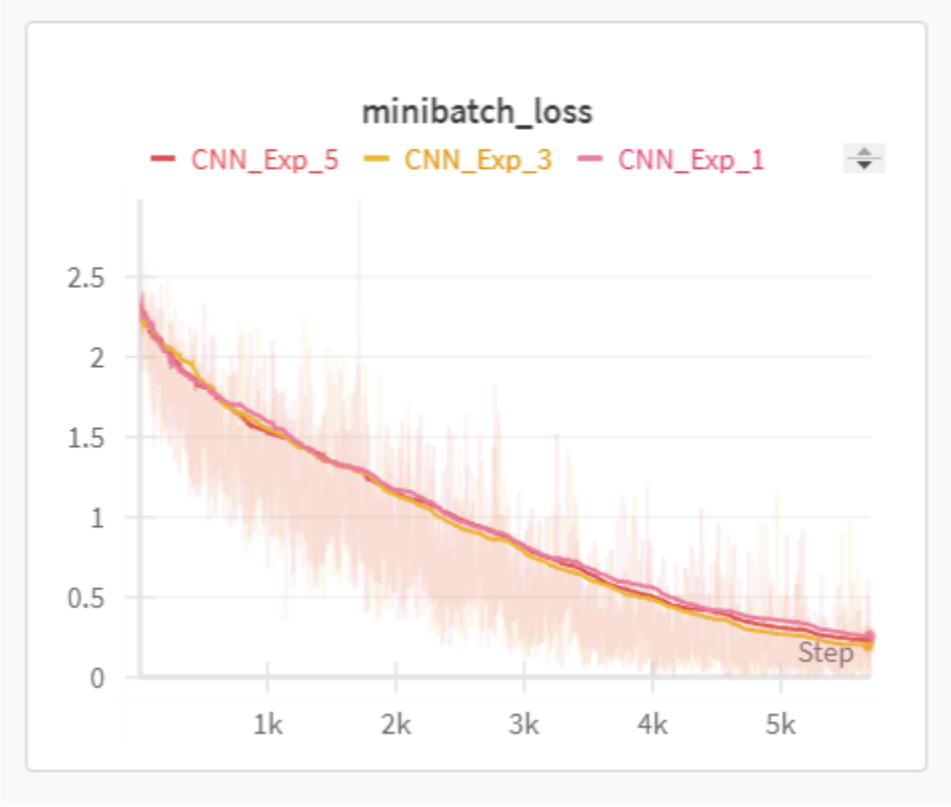
As can be observed, the stratified split is applied to the dataset during the splitting of the dataset and hence, the ratio of each class is maintained across the different splits, which is important for learning the data distribution.

2.2.(c). Observing the training and validation data, it can be seen that the training data achieves an accuracy of **96.93%** at best. In comparison, the validation set achieves an accuracy of **63.61%**, and the significant gap between the training and validation accuracy suggests that the model has indeed overfitted to a large extent. The training and validation loss values also suggest the same, as the training loss is **0.011**, while the validation loss is **2.06**, suggesting that the model has overfitted to a large extent.



Experimented over several runs, it is observed that the model overfits and there is a significant gap between the performance on the training and validation datasets.



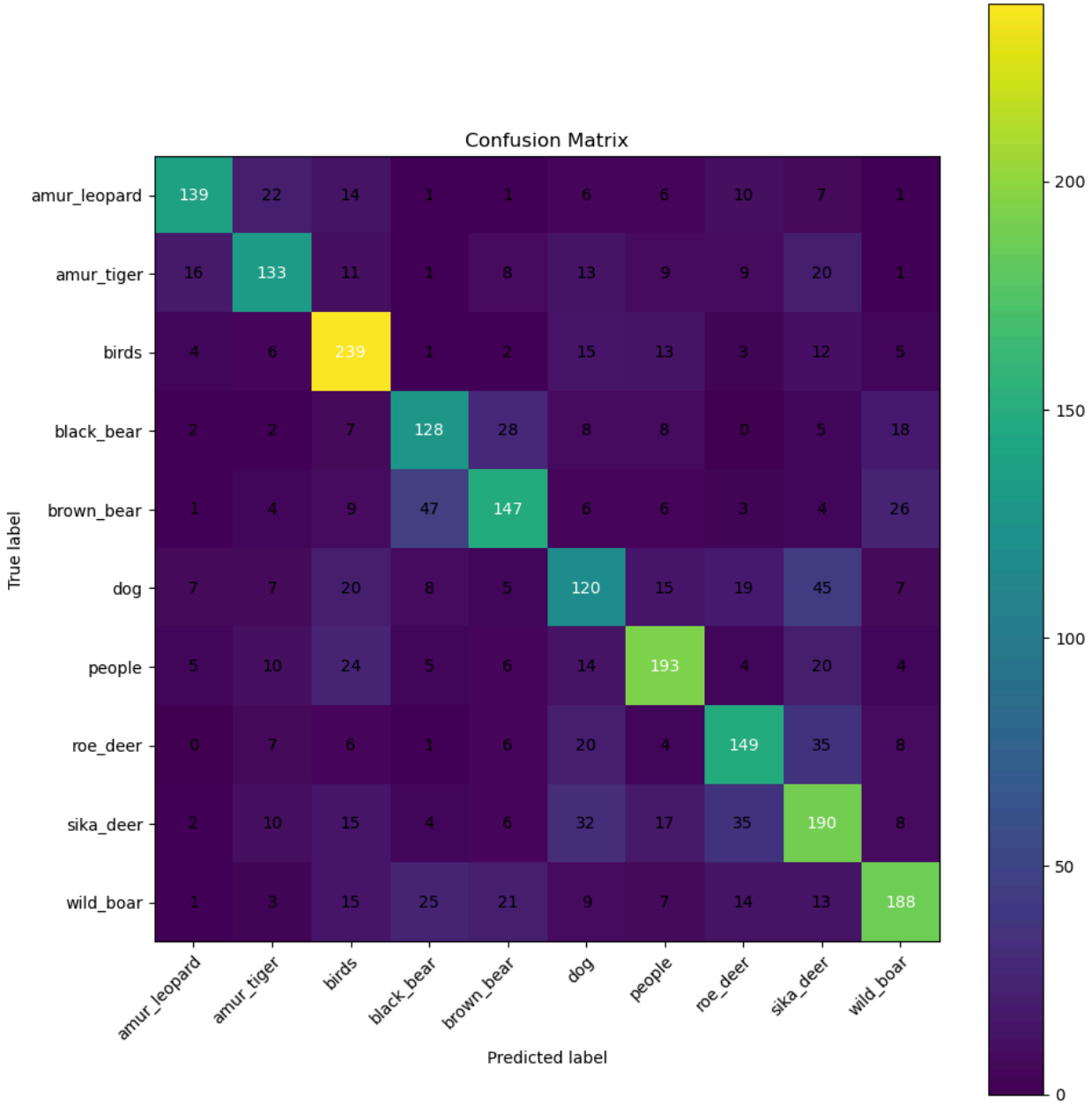


The CNN architecture achieves a test accuracy of 65.93% at best.

2.2.(d). Results on the test dataset are observed as follows:

Test Accuracy: **65.9375%**
F1 Score: 0.6585133121155642
Precision: 0.6597975267778609
Recall: 0.659375

The **confusion matrix** for the test set:



It can be noted that there are many mistakes between the black and brown bear and the roe and sika deer. Since the difference between the two sets of classes might be very small, hence the model is confusing between them.

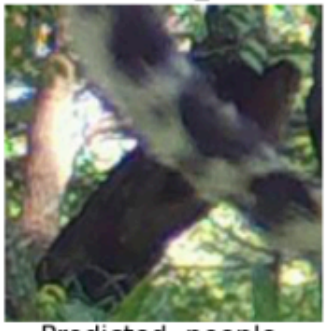
2.2. (e) The misclassification plotting suggests that misclassification of images can occur due to several different reasons -

- There are certain images which are hazy/shaken and the image is not clear. Hence it makes sense that the model cannot detect these images.
- Observing the confusion matrix and misclassification plots, it can be analysed that there are a lot of misclassifications between black and brown bears, and between roe and sika deer. The difference between the species is probably not significant enough to be labelled differently, or under certain lighting conditions, it is difficult to tell apart a brown and black bear from each other.
- There are a few images that have been either cut out and the entire class is not present in the image, or the label class has been occluded by disturbances.

For the above cases, data augmentation of the data will help the model to learn different nuances of the dataset, which will make the model more robust. Along with this, we can explore class-balancing techniques to make all the classes have the same number of elements in the training dataset. Additionally, we can try to learn the model for a longer period so that it can capture more nuances of the dataset.

Predicted: roe_deer,
Actual: amur_leopard

Predicted: birds,
Actual: amur_leopard



Predicted: sika_deer,
Actual: amur_leopard

Predicted: birds,
Actual: amur_tiger

Predicted: people,
Actual: amur_tiger

Predicted: amur_leopard,
Actual: amur_tiger

Predicted: brown_bear,
Actual: birds

Predicted: brown_bear,
Actual: birds

Predicted: roe_deer,
Actual: birds

Predicted: wild_boar,
Actual: black_bear

Predicted: brown_bear,
Actual: black_bear

Predicted: wild_boar,
Actual: black_bear

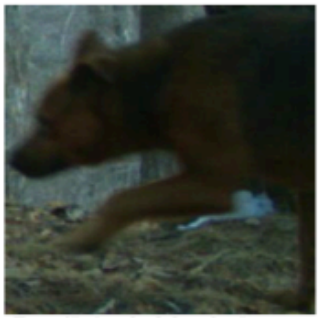
Predicted: sika_deer,
Actual: brown_bear

Predicted: black_bear,
Actual: brown_bear

Predicted: sika_deer,
Actual: brown_bear

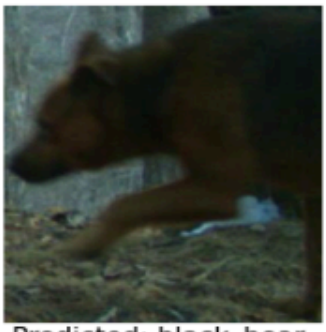
Predicted: birds,
Actual: dog

Predicted: amur_tiger,
Actual: dog



Predicted: sika_deer,
Actual: dog

Predicted: dog,
Actual: people



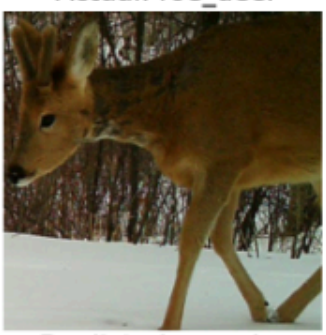
Predicted: black_bear,
Actual: people

Predicted: dog,
Actual: people

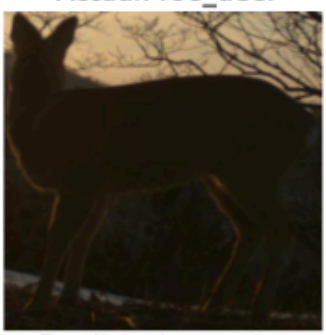
Predicted: sika_deer,
Actual: roe_deer

Predicted: dog,
Actual: roe_deer

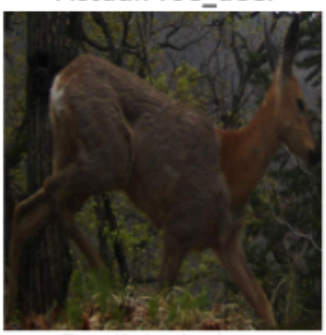
Predicted: sika_deer,
Actual: roe_deer



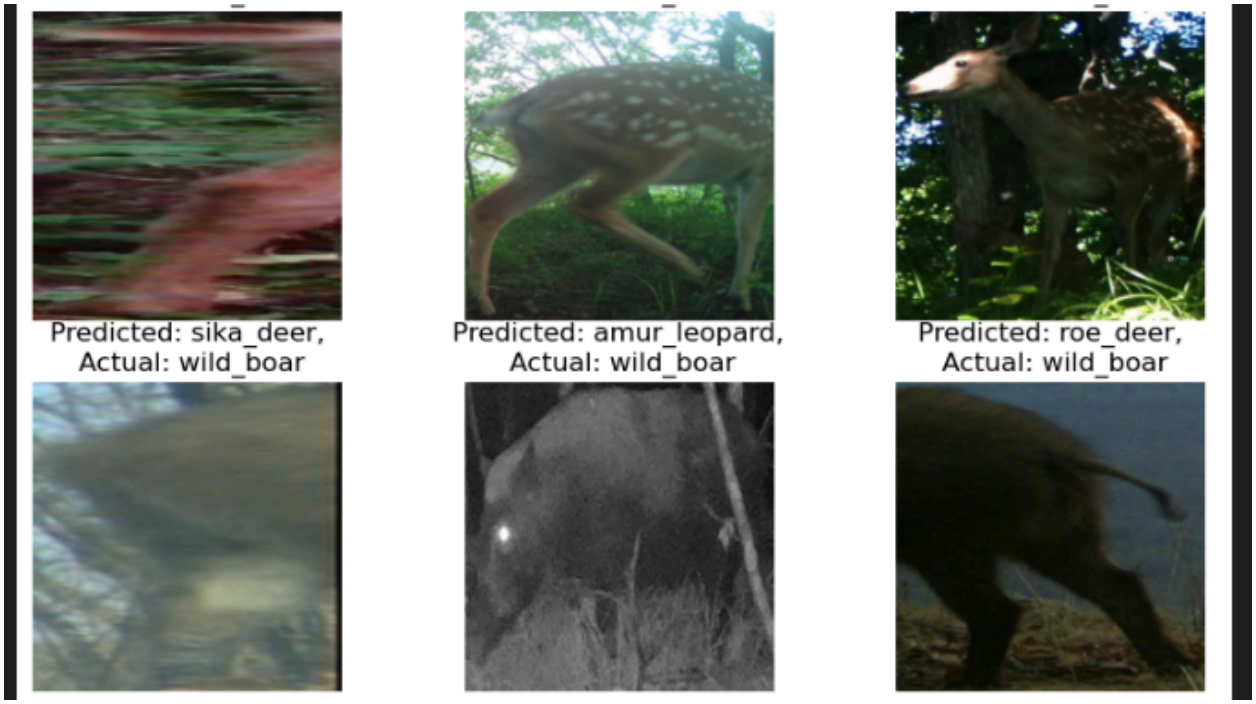
Predicted: people,
Actual: sika_deer



Predicted: people,
Actual: sika_deer

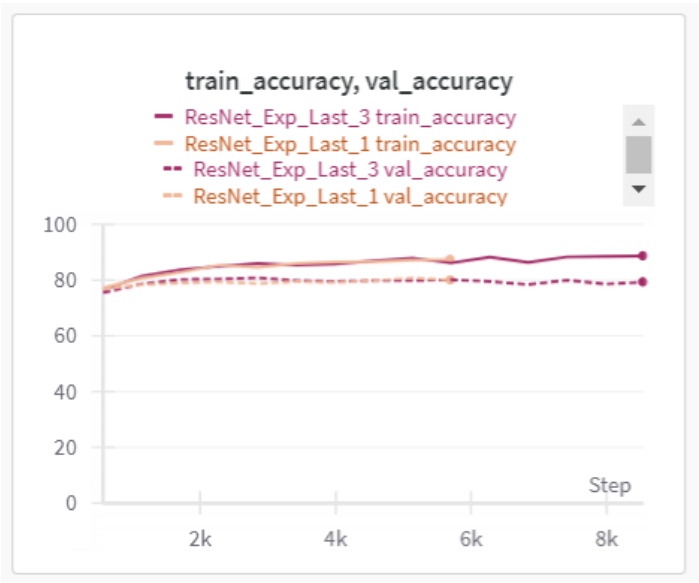


Predicted: dog,
Actual: sika_deer

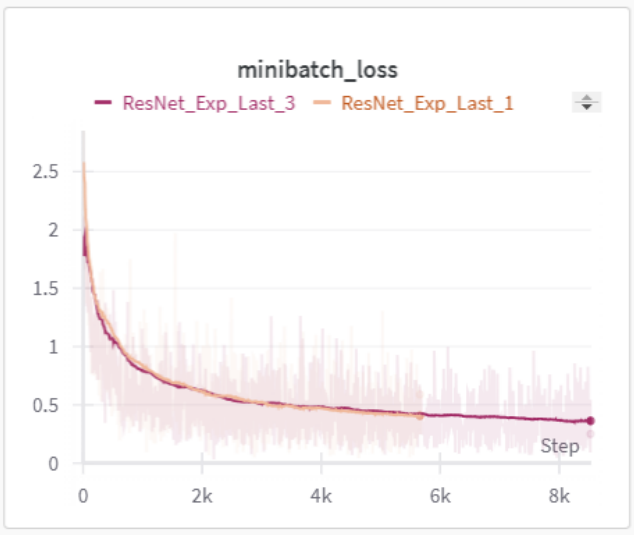


Q3.

3.(b). Training and validation loss plots suggest that the model partially overfits the training data, as we obtain an 88.75% training accuracy and 79.38% validation accuracy using the pre-trained ResNet18 model. The backbone of the model was frozen and only the final fully connected layer of the ResNet architecture was trained. The training loss was 0.3295, while the validation loss was 0.5971, suggesting some amount of overfitting on the training data.

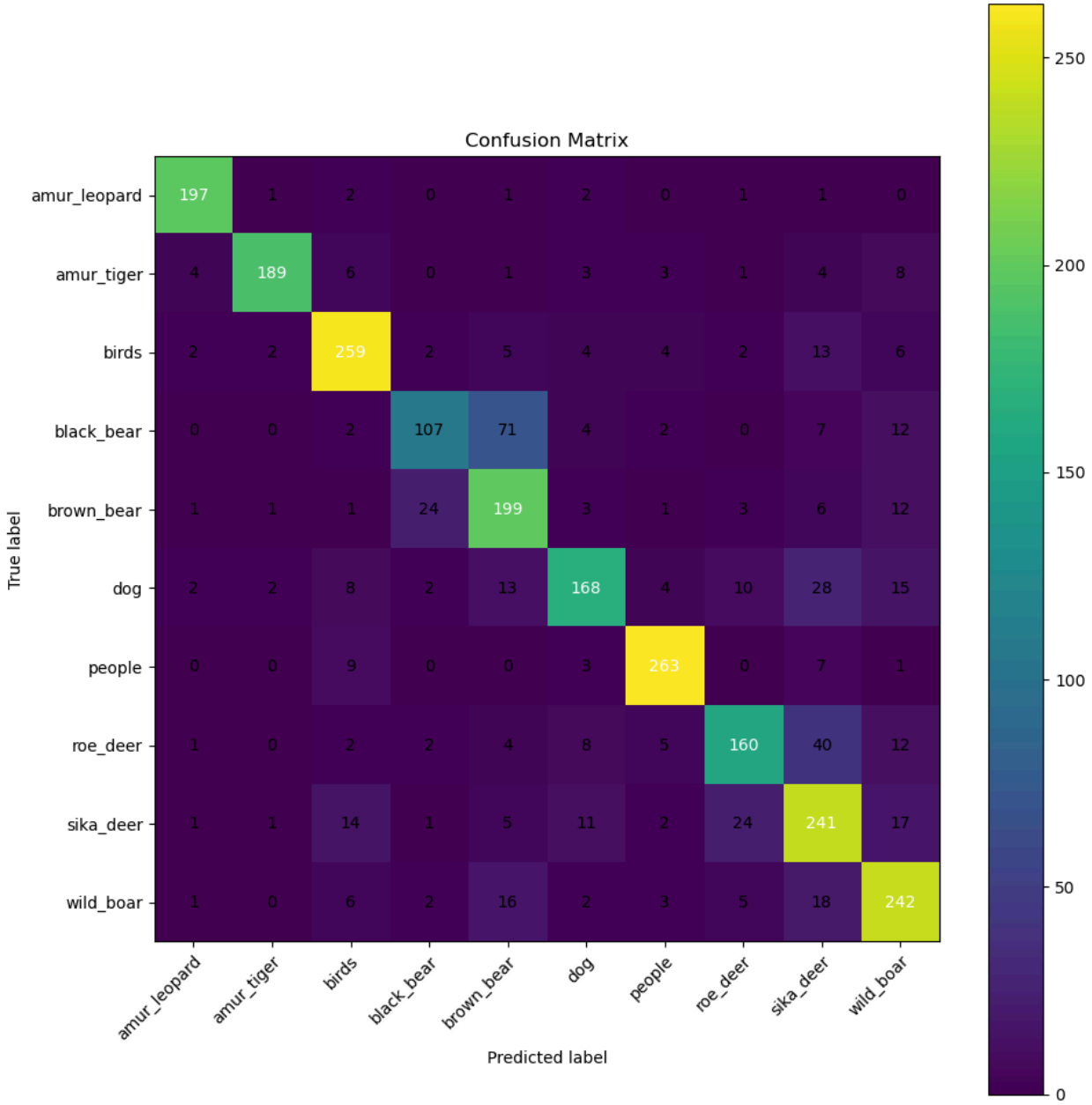

train_accuracy, val_accuracy
ResNet_Exp_Last_3 train_accuracy
ResNet_Exp_Last_1 train_accuracy
ResNet_Exp_Last_3 val_accuracy
ResNet_Exp_Last_1 val_accuracy
100
80
60
40
20
0
Step
2k
4k
6k
8k


val_loss, train_loss
ResNet_Exp_Last_3 val_loss
ResNet_Exp_Last_1 val_loss
ResNet_Exp_Last_3 train_loss
ResNet_Exp_Last_1 train_loss
0.6
0.4
0.2
0
Step
2k
4k
6k
8k

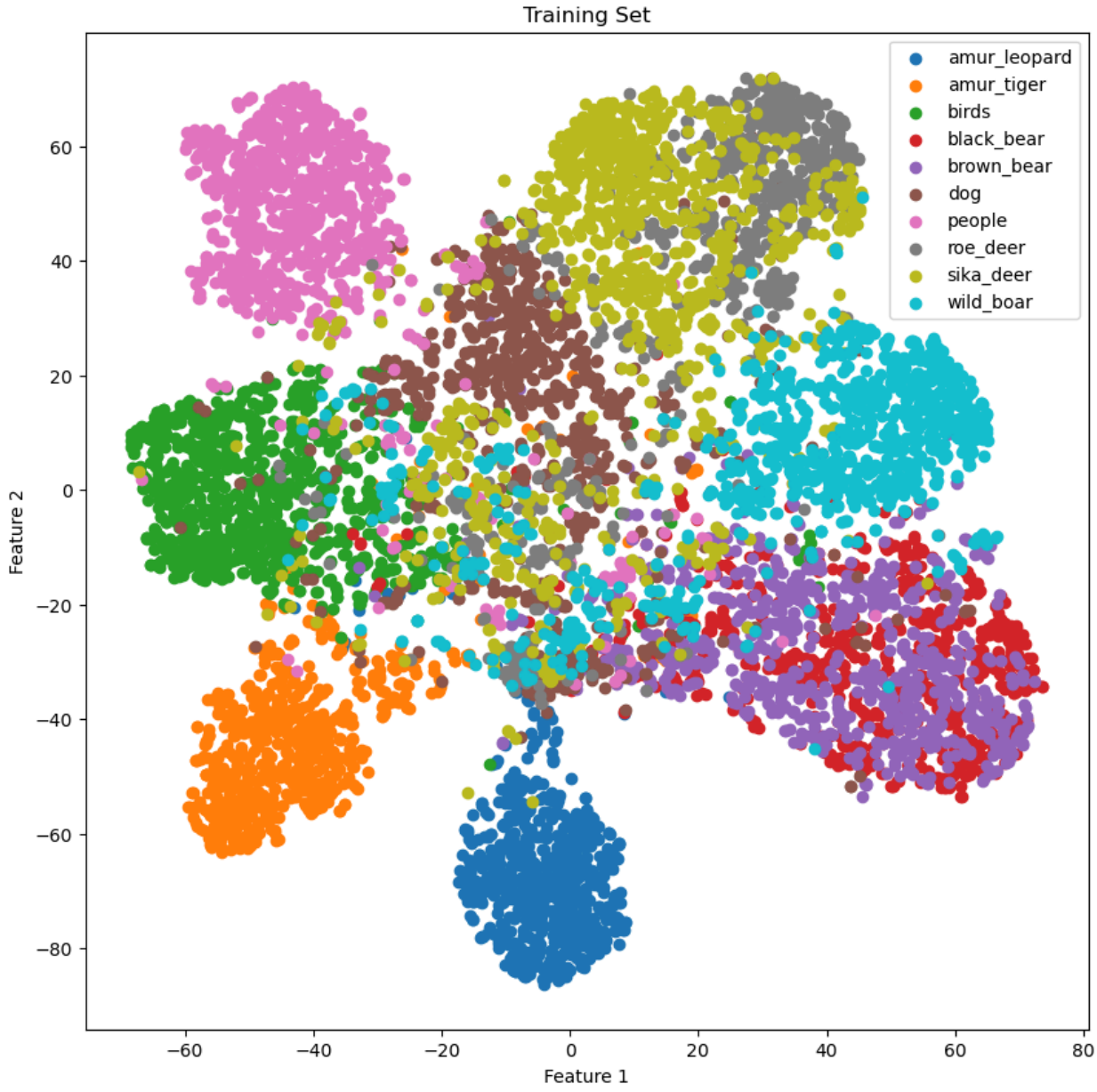

minibatch_loss
ResNet_Exp_Last_3
ResNet_Exp_Last_1
2.5
2
1.5
1
0.5
0
Step
0
2k
4k
6k
8k

3. (c). The model achieves the following result by training on the ResNet18 backbone pre-trained on the ImageNet dataset-
Test Accuracy: **79.1015625%**
F1 Score: 0.7901184833394301
Precision: 0.7983836335072794
Recall: 0.791015625

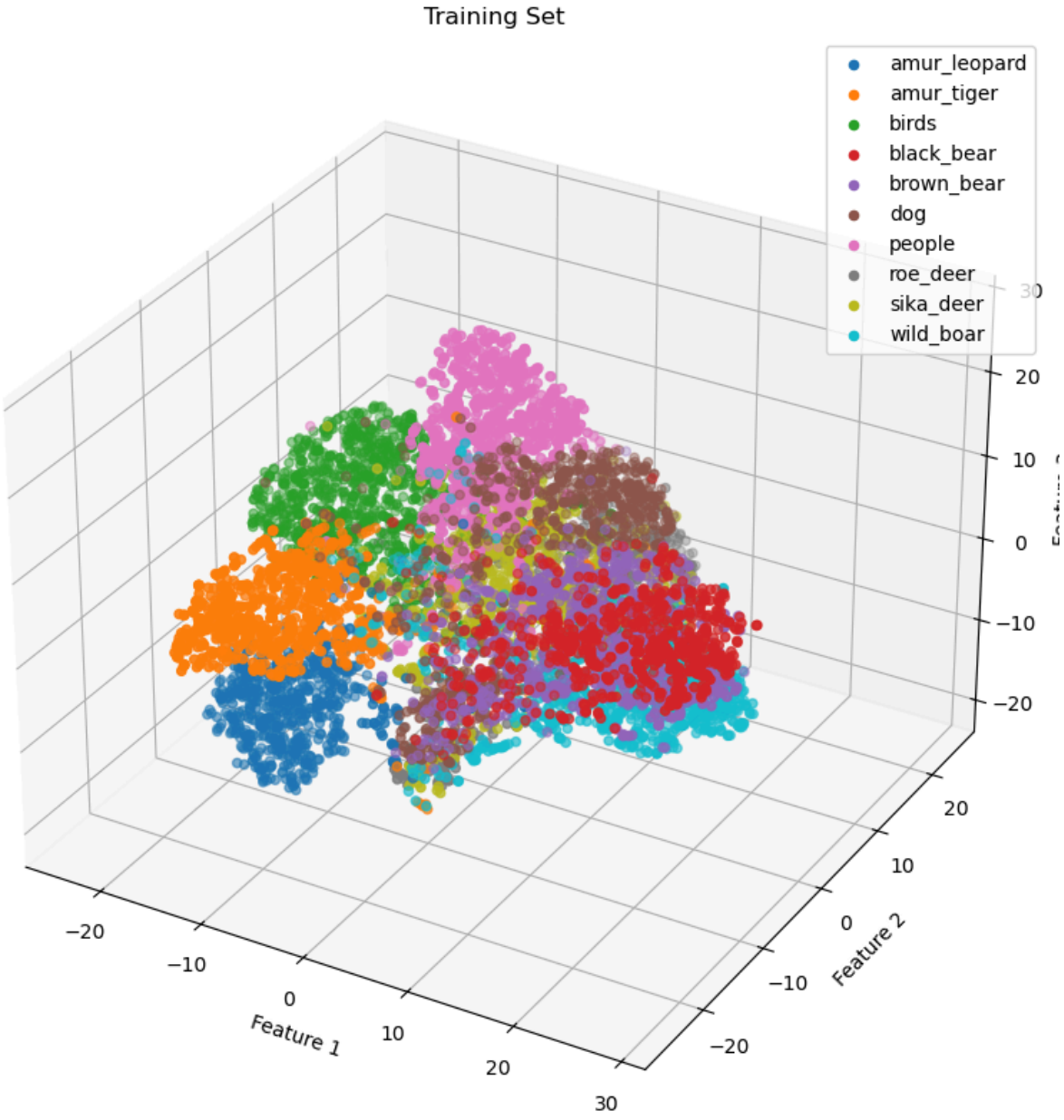
The confusion matrix is as follows -



3. (d) For visualization of the images as feature vectors is a lower dimension, we make use of the ResNet18 backbone, along with TSNE. The images are passed through the forward propagation through the network, where the images are reduced to a 10-dimensional feature vector. Subsequently, the 10-dimensional feature vector space is learnt by TSNE, and we plot the data distribution in 2-dimensions and 3-dimensions. We can observe a larger overlap between the classes for the 2-dimensional plot, which is due to the larger information loss through the dimensionality reduction, whereas the class-wise images are separated better through the 3-dimensional plot.
2-dimensional TSNE plot of the training dataset-



3-dimensional plot of the validation data distribution is as follows -

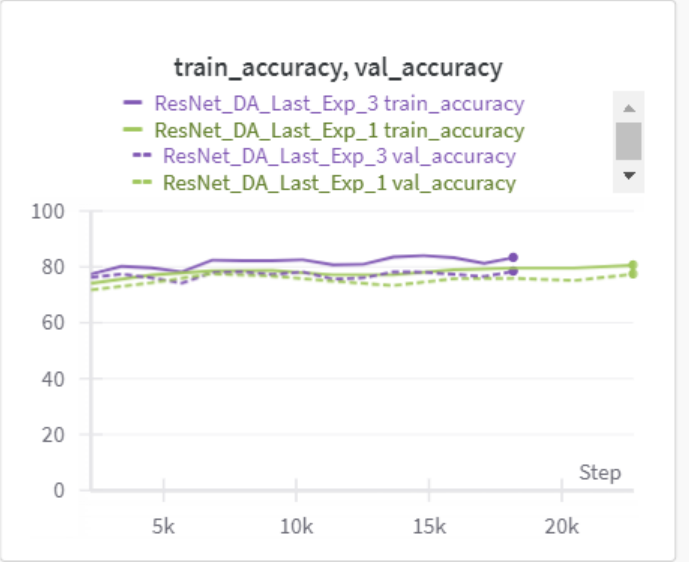


4. (a). Data augmentation techniques such as RandomCropping, RandomHorizontalFlip, and RandomRotation are used for augmenting the data with slight variations of the training dataset. Firstly, the images are resized to 256 x 256, where we RandomCrop a section of 224 x 224 of the image, followed by a RandomHorizontalFlip of the image with a probability of 0.5, and the RandomRotation of 30 degrees. The RandomHorizontalFlip makes the model invariant to lateral mirror-images of the original image, while the RandomRotation makes the model invariant to side-wise rotation of the image, helping the model

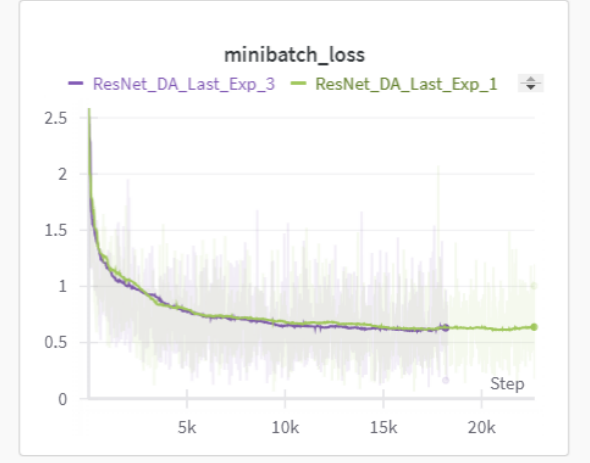
to learn better feature representations. RandomCrop helps the model to focus on random-but few features of the model, helping the model to not focus on only a few features, but rather generalizing to many different features.

4. (c). After data augmentation, we can certainly say that the problem of overfitting has been solved, as the gap between the training and validation plots is closer, with a training accuracy of 83.63% and validation accuracy of 77.50%, and a training loss of 0.57, validation loss of 0.6671, which suggests that the gap between the training and validation performance is less, and the model has learnt the feature representation of the data distribution well.

4. (d) The metrics obtained after training the pre-trained ResNet backbone with data augmented training data are as follows -
Test Accuracy: 78.1640625%
F1 Score: 0.7806122857258431
Precision: 0.7874506418194003
Recall: 0.781640625

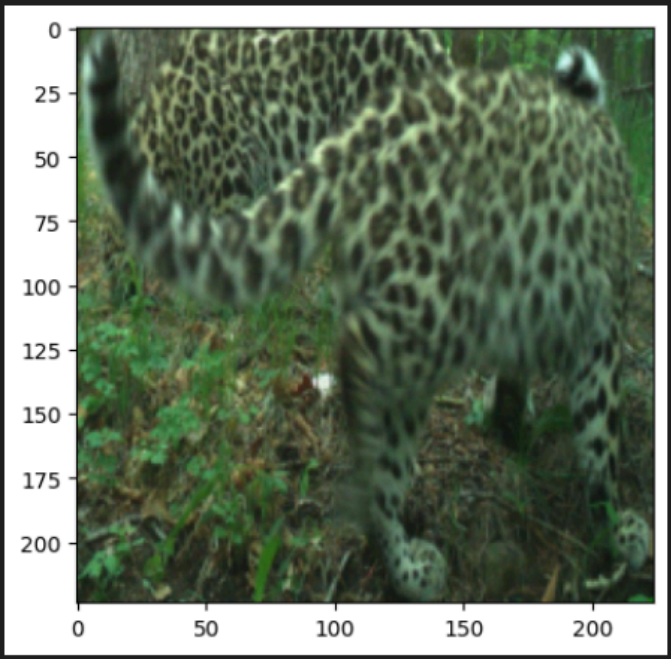

train_accuracy, val_accuracy
ResNet_DA_Last_Exp_3 train_accuracy
ResNet_DA_Last_Exp_1 train_accuracy
ResNet_DA_Last_Exp_3 val_accuracy
ResNet_DA_Last_Exp_1 val_accuracy
100
80
60
40
20
0
Step
5k
10k
15k
20k


val_loss, train_loss
ResNet_DA_Last_Exp_3 val_loss
ResNet_DA_Last_Exp_1 val_loss
ResNet_DA_Last_Exp_3 train_loss
ResNet_DA_Last_Exp_1 train_loss
0.6
0.4
0.2
0
Step
5k
10k
15k
20k

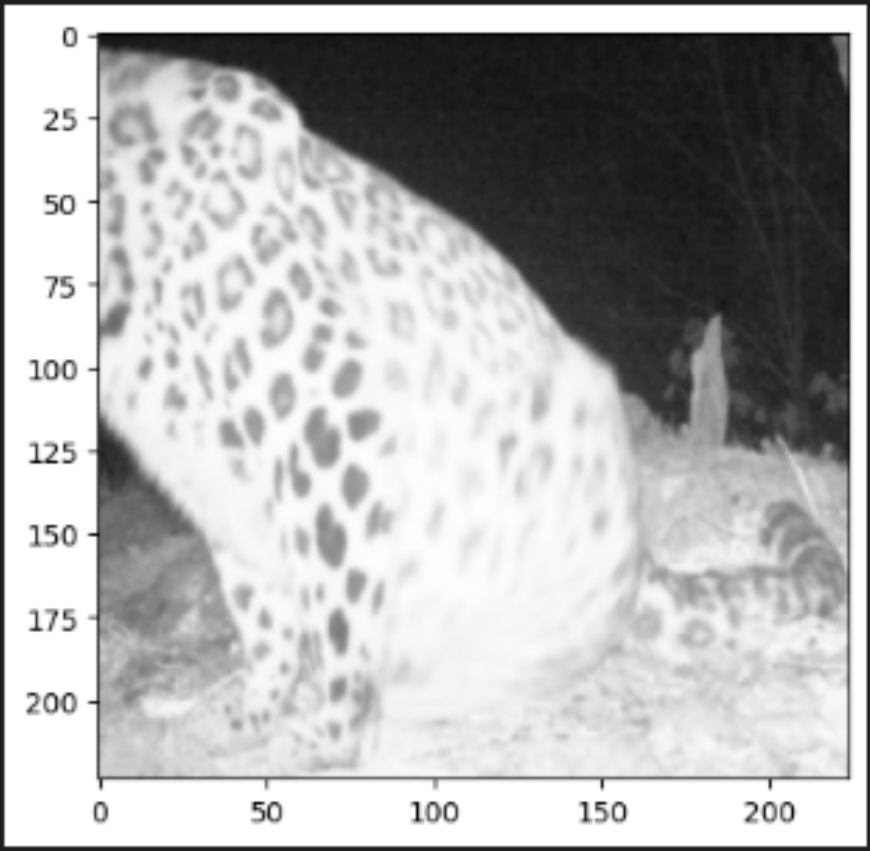

minibatch_loss
ResNet_DA_Last_Exp_3
ResNet_DA_Last_Exp_1
2.5
2
1.5
1
0.5
0
Step
5k
10k
15k
20k

5. (BONUS)
Approach: For each class, we find 3 images which were misclassified by the CNN architecture and cache it. Then, we find the 10-dimensional feature representation of all the images in the training dataset, and then find the centroid of each class on the 10-dimensional feature representation and store it. Now, for each image, we go over the misclassified images and pass it through the ResNet18 and ResNet18 architecture with Data augmentation, and predict the outputs in each case. We observe that in general, the ResNet models give either a correct classification, or a shorter distance from the 10-dimensional space. This suggests that the ResNet trained models have a better 10-dimensional representation of the data distribution, and hence is able to better model the data. It is observed that the ResNet18 and data-augmented ResNet18 gives close results, while the data-augmented ResNet18 gives better/closer distances on more challenging images, such as those which have occlusions, or are hazy.

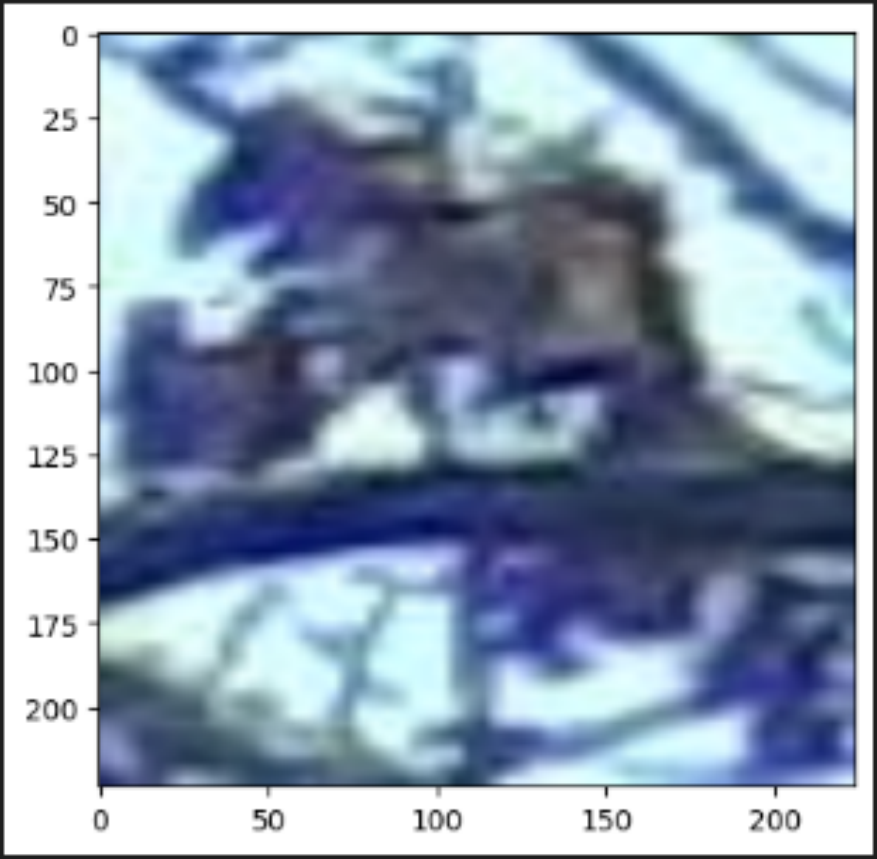
```
CNN
Predicted:  birds , Actual:  amur_leopard
Distance from ground truth:  24.12
Distance from prediction:  13.64
ResNet18
Predicted:  sika_deer , Actual:  amur_leopard
Distance from ground truth:  40.99
Distance from prediction:  47.85
ResNet18 DAG
Predicted:  dog , Actual:  amur_leopard
Distance from ground truth:  34.43
Distance from prediction:  39.48
```



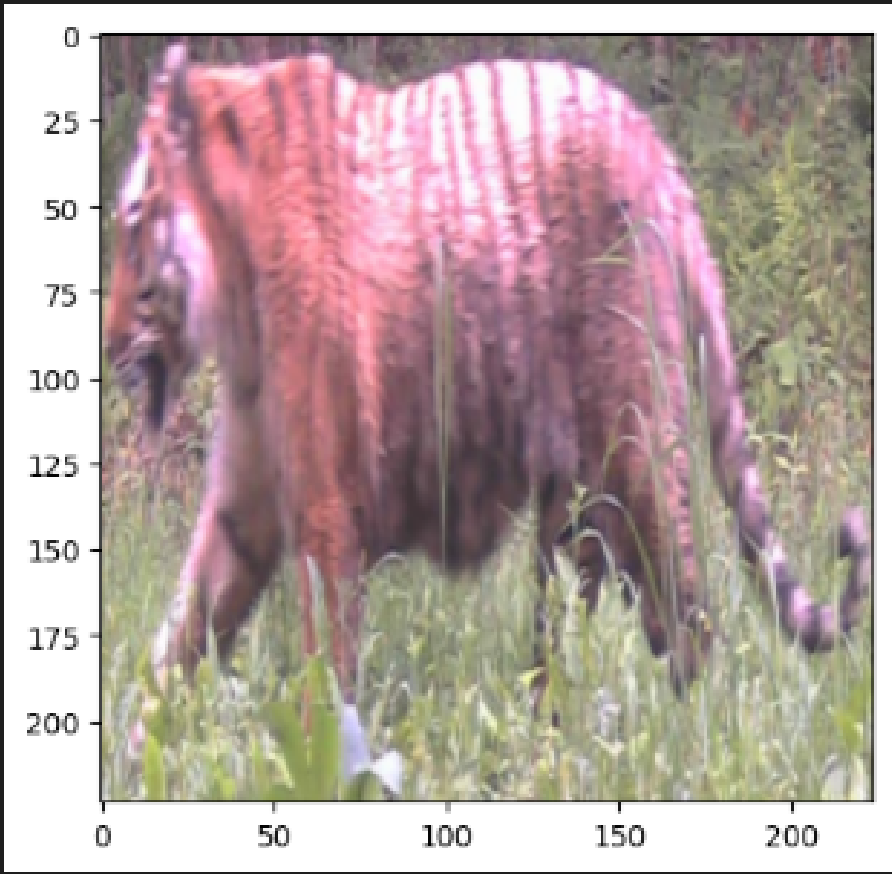
```
CNN
Predicted:  dog , Actual:  amur_leopard
Distance from ground truth:  20.47
Distance from prediction:  15.2
ResNet18
Predicted:  sika_deer , Actual:  amur_leopard
Distance from ground truth:  40.93
Distance from prediction:  38.83
ResNet18 DAG
Predicted:  dog , Actual:  amur_leopard
Distance from ground truth:  34.32
Distance from prediction:  30.34
```



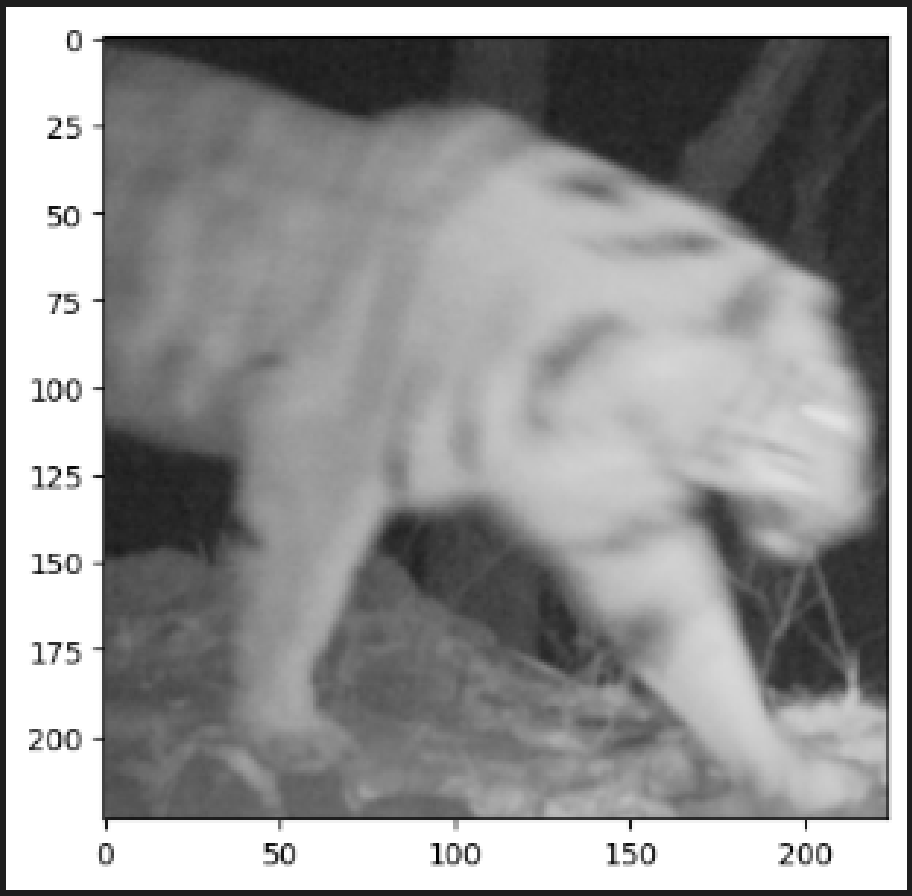
```
CNN
Predicted:  birds , Actual:  amur_leopard
Distance from ground truth:  58.07
Distance from prediction:  49.22
ResNet18
Predicted:  sika_deer , Actual:  amur_leopard
Distance from ground truth:  40.98
Distance from prediction:  47.87
ResNet18 DAG
Predicted:  dog , Actual:  amur_leopard
Distance from ground truth:  34.63
Distance from prediction:  39.68
```



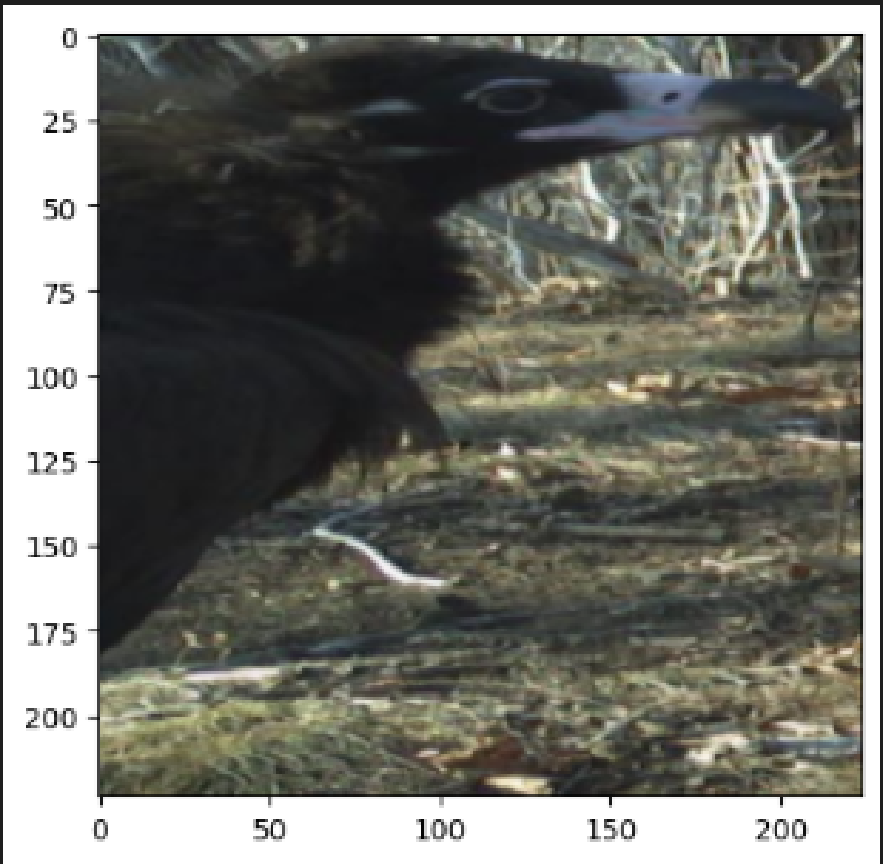
```
CNN
Predicted:  birds , Actual:  amur_tiger
Distance from ground truth:  30.76
Distance from prediction:  18.81
ResNet18
Predicted:  sika_deer , Actual:  amur_tiger
Distance from ground truth:  38.95
Distance from prediction:  47.86
ResNet18 DAG
Predicted:  dog , Actual:  amur_tiger
Distance from ground truth:  32.35
Distance from prediction:  39.58
```



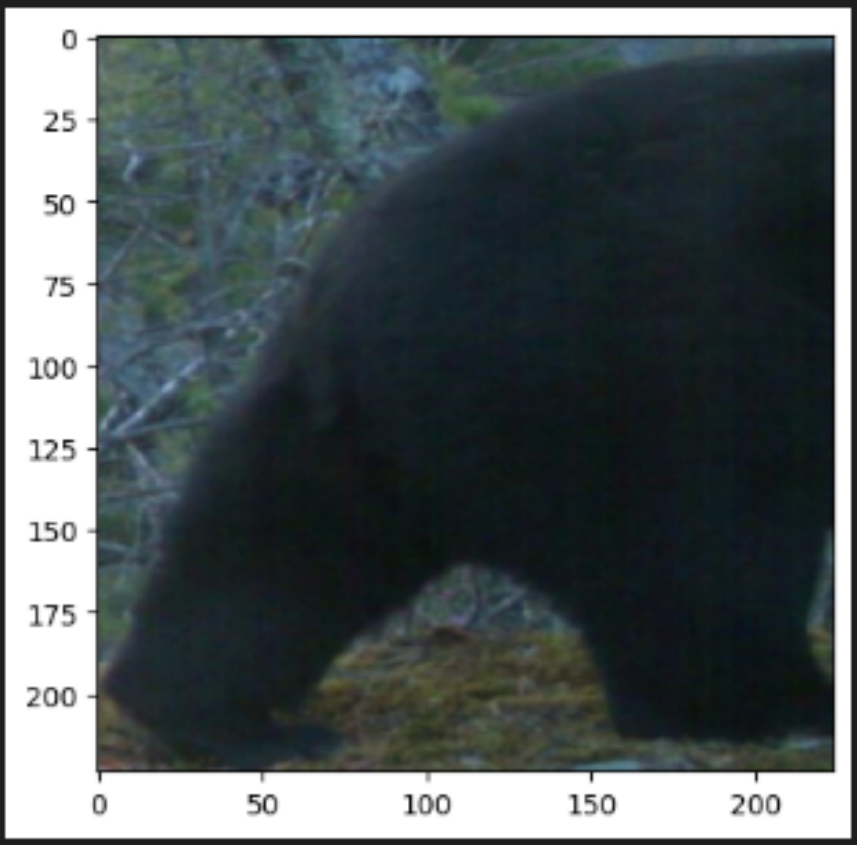
```
CNN
Predicted:  dog , Actual:  amur_tiger
Distance from ground truth:  20.39
Distance from prediction:  21.16
ResNet18
Predicted:  sika_deer , Actual:  amur_tiger
Distance from ground truth:  38.73
Distance from prediction:  38.8
ResNet18 DAG
Predicted:  dog , Actual:  amur_tiger
Distance from ground truth:  32.45
Distance from prediction:  30.3
```



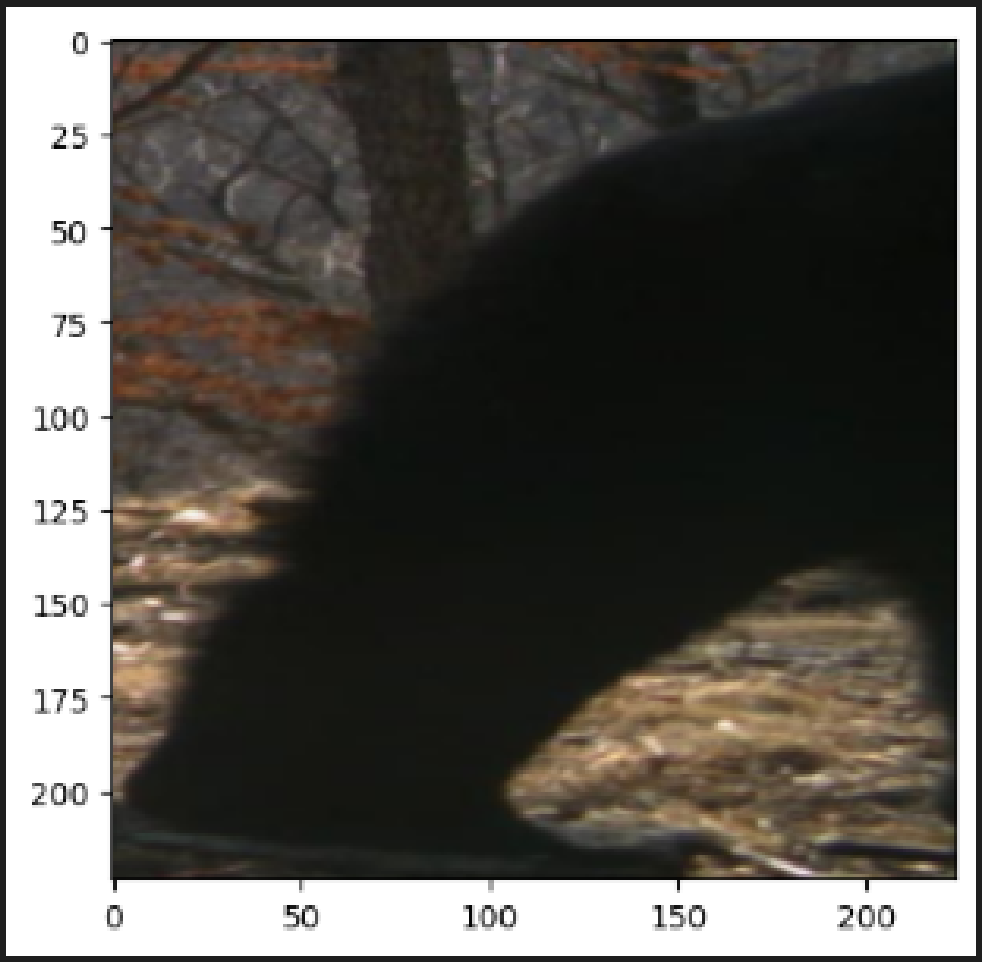
```
CNN
Predicted:  people , Actual:  birds
Distance from ground truth:  41.43
Distance from prediction:  19.68
ResNet18
Predicted:  sika_deer , Actual:  birds
Distance from ground truth:  47.74
Distance from prediction:  73.45
ResNet18 DAG
Predicted:  dog , Actual:  birds
Distance from ground truth:  39.58
Distance from prediction:  64.16
```



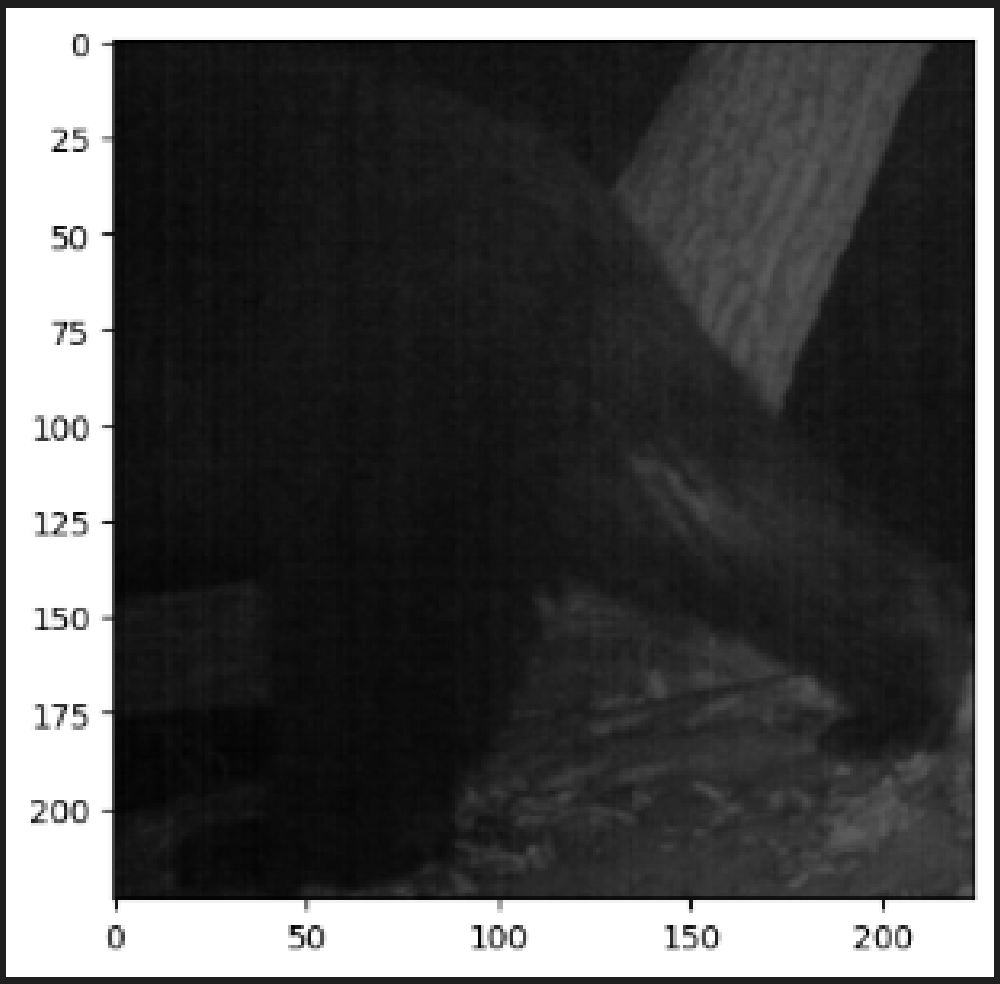
```
CNN
Predicted:  brown_bear , Actual:  black_bear
Distance from ground truth:  25.24
Distance from prediction:  19.88
ResNet18
Predicted:  sika_deer , Actual:  black_bear
Distance from ground truth:  34.87
Distance from prediction:  29.65
ResNet18 DAG
Predicted:  dog , Actual:  black_bear
Distance from ground truth:  26.87
Distance from prediction:  21.71
```



```
CNN
Predicted:  wild_boar , Actual:  black_bear
Distance from ground truth:  18.76
Distance from prediction:  10.77
ResNet18
Predicted:  sika_deer , Actual:  black_bear
Distance from ground truth:  34.74
Distance from prediction:  31.76
ResNet18 DAG
Predicted:  dog , Actual:  black_bear
Distance from ground truth:  26.97
Distance from prediction:  23.91
```



```
CNN
Predicted:  amur_tiger , Actual:  brown_bear
Distance from ground truth:  28.59
Distance from prediction:  38.23
ResNet18
Predicted:  sika_deer , Actual:  brown_bear
Distance from ground truth:  29.55
Distance from prediction:  38.91
ResNet18 DAG
Predicted:  dog , Actual:  brown_bear
Distance from ground truth:  21.65
Distance from prediction:  32.5
```



2.6. We have trained CNN architecture, ResNet18 backbone pre-trained on the ImageNet dataset, and the ResNet18 backbone pre-trained on the ImageNet dataset with data augmentation.
The CNN architecture is observed to be significantly over-fitting on the training dataset, as we obtain the following results:

Training Accuracy: 96.93%
Validation Accuracy: 63.61%
Test Accuracy: **65.9375%**
Training Loss: 0.011
Validation Loss: 2.06
F1 Score: 0.6585133121155642
Precision: 0.6597975267778609
Recall: 0.659375

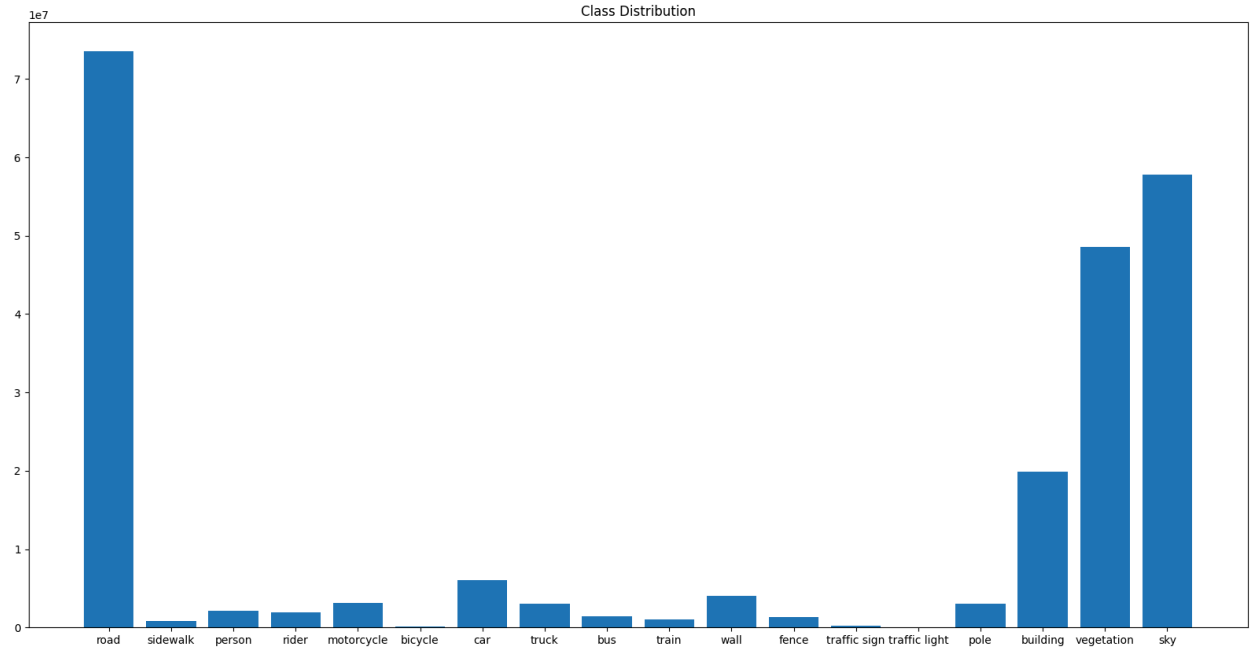
These results signify a major gap between the results obtained by the training and validation dataset, suggesting that the model has over-fit on the training dataset, as we obtain only a 65.93% accuracy on the test dataset.

The ResNet18 backbone was pre-trained on the ImageNet dataset and we only trained the last layer of the model, achieving the following results. These results signify some amount of overfitting, however it is much better than the CNN architecture results -
Training Accuracy: 88.75%
Validation Accuracy: 79.38%
Test Accuracy: **79.1015625%**
Training Loss: 0.3295
Validation Loss: 0.5971
F1 Score: 0.7901184833394301
Precision: 0.7983836335072794
Recall: 0.791015625

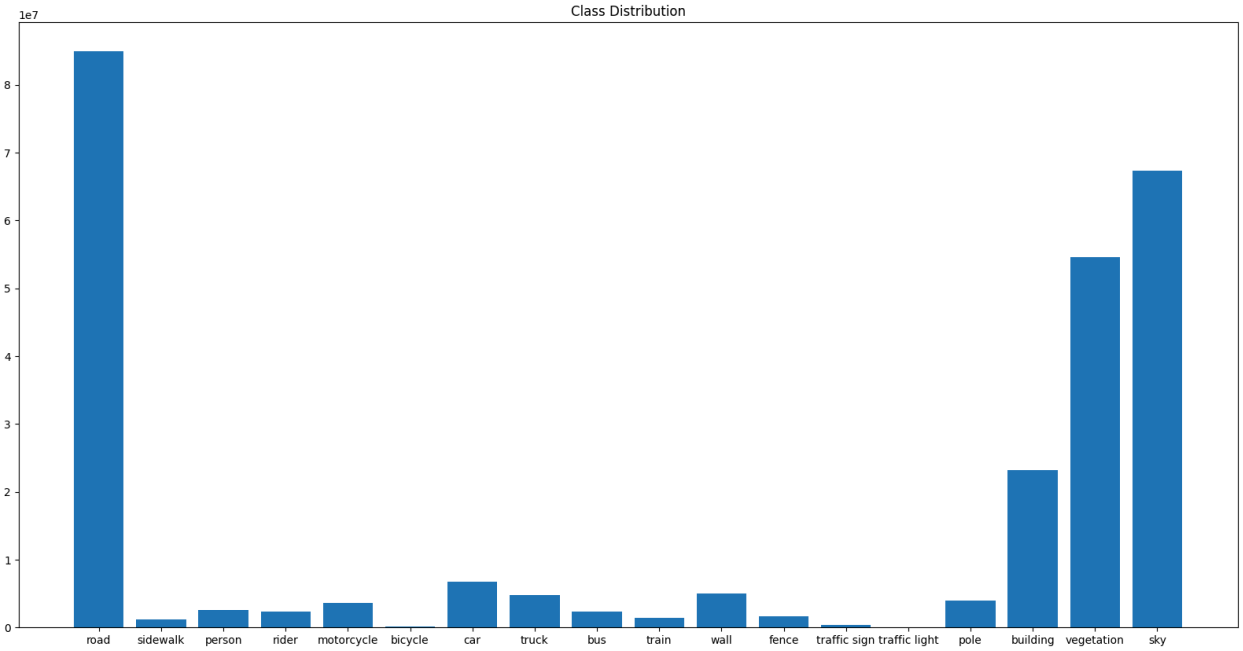
The ResNet18 backbone was pre-trained on the ImageNet dataset, with data augmentation, and we only trained the last layer of the model, achieving the following results. These results signify no overfitting, achieving competitive results compared to the above ResNet model-
Training Accuracy: 83.63%

Validation Accuracy: 77.50%
Test Accuracy: 78.1640625%
Training Loss: 0.570
Validation Loss: 0.6671
F1 Score: 0.7806122857258431
Precision: 0.7874506418194003
Recall: 0.781640625

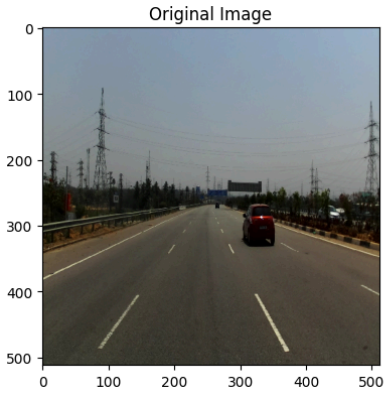
Q3. Image Segmentation
3.1.(b) The pixel-wise distribution of all the images on the training dataset for each class are as follows-

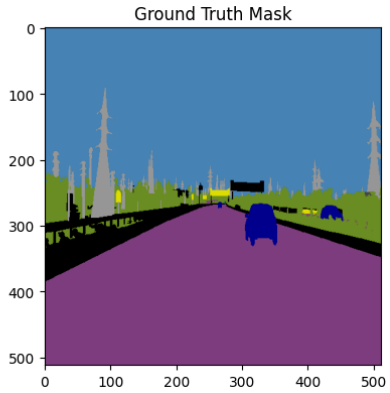


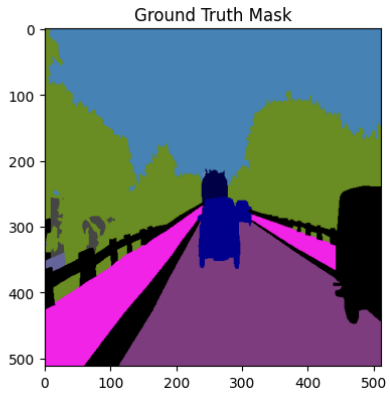
The distribution of the test dataset are as follows -

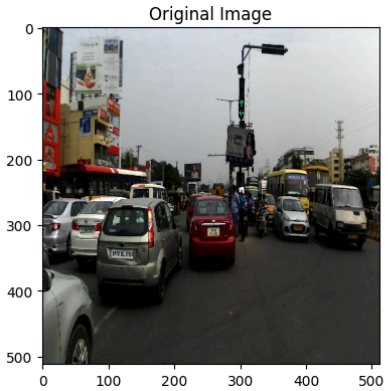


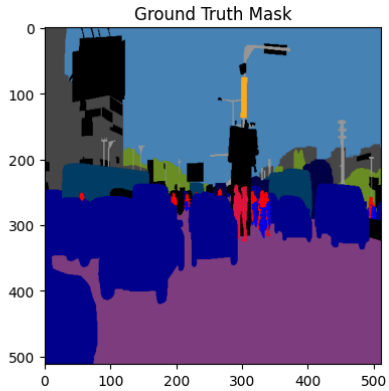
3.1.(c) We analyse images from the training dataset along with their masks as follows -

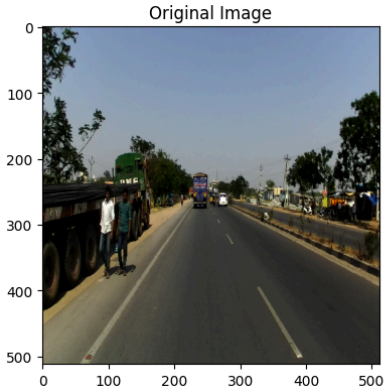


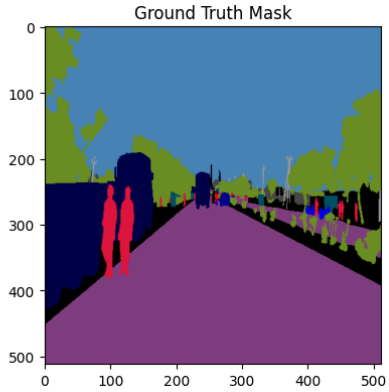












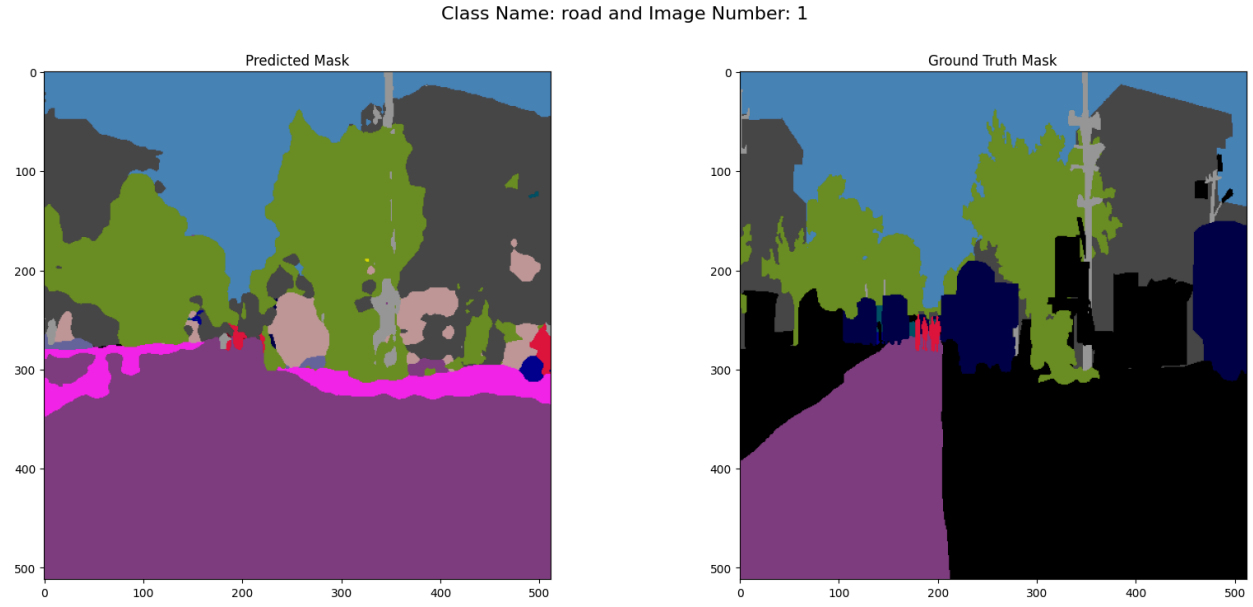
3.2.(b) Pixel-wise metrics are computed on the 30% split of the dataset (test data), where we have applied smoothing techniques to avoid a zero in the denominator and compute the IoU, accuracy, precision, recall, and dice coefficient.
Observations:

- Classes like "road," "vegetation," and "sky" have relatively higher IoU scores, suggesting that the model performs well in segmenting these classes.
- Classes like "train" and "traffic light" have IoU scores of 0, indicating that the model fails to detect or accurately segment these classes.
- Classes like "bus," "truck," "rider," and "person" have high accuracy, suggesting that the model performs well in correctly classifying these classes.

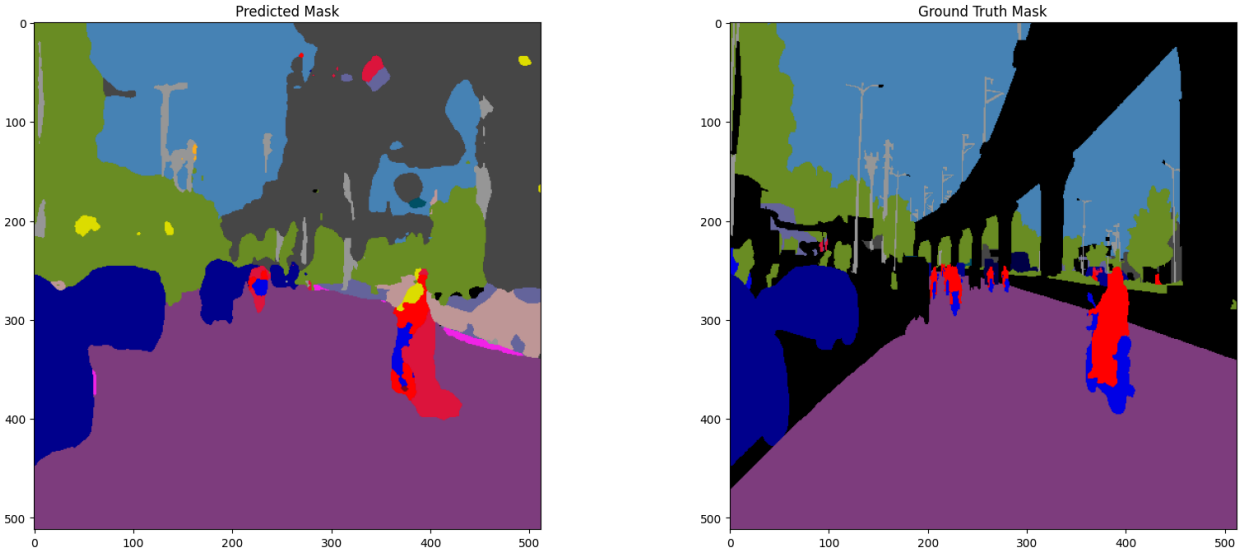
However, classes with fewer instances or smaller regions (e.g., "bicycle," "traffic sign") may have high accuracy but low IoU scores, indicating that the model struggles with precise segmentation for these classes.

Class Name	IoU	Precision	Recall	Dice Coefficient	Accuracy
road	0.72	0.74	0.95	0.82	0.9
vegetation	0.6	0.67	0.85	0.72	0.93
sky	0.54	0.95	0.54	0.65	0.9
car	0.28	0.3	0.58	0.35	0.97
building	0.24	0.27	0.56	0.33	0.86
pole	0.15	0.26	0.28	0.23	0.98
rider	0.11	0.26	0.15	0.16	0.99
person	0.1	0.16	0.17	0.14	0.99
motorcycle	0.09	0.31	0.12	0.14	0.99
sidewalk	0.05	0.06	0.11	0.07	0.97
wall	0.05	0.13	0.07	0.07	0.98
traffic sign	0.04	0.05	0.08	0.06	1
truck	0.02	0.1	0.02	0.03	0.99
bus	0.02	0.06	0.02	0.02	0.99
fence	0.02	0.04	0.04	0.03	0.99
bicycle	0.01	0.01	0.02	0.01	0.99
train	0	0	0	0	1
traffic light	0	0	0	0	1

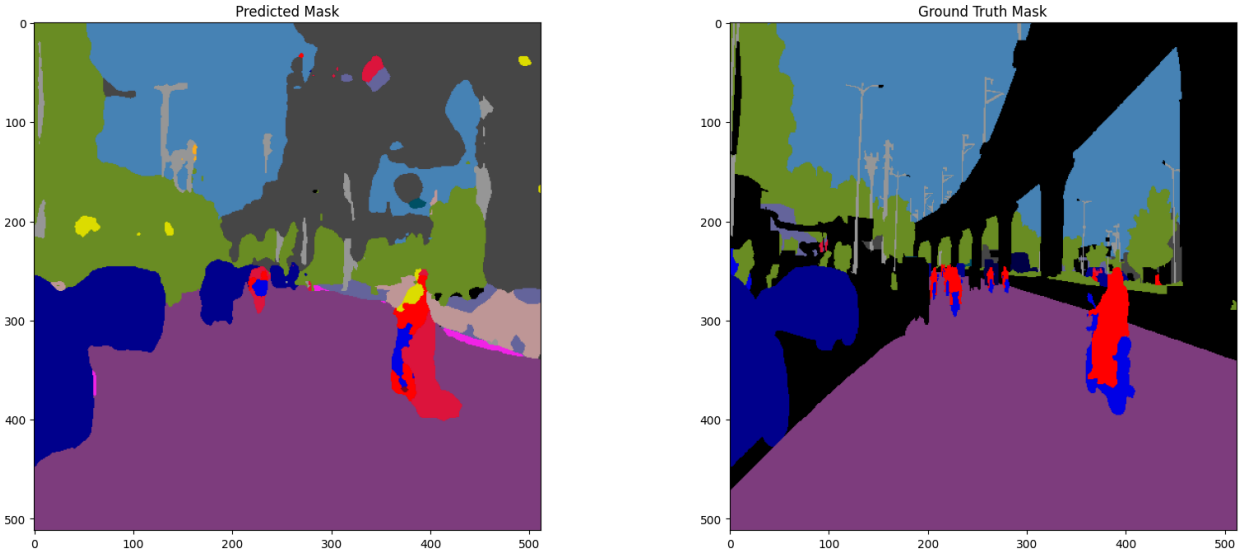
3.2.(c)



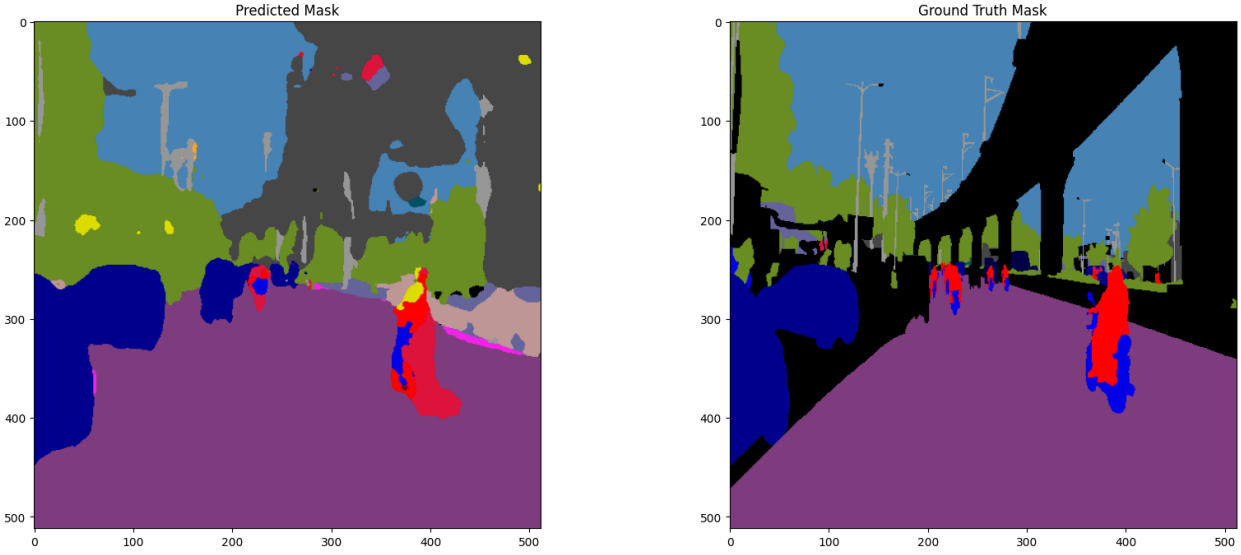
Class Name: sidewalk and Image Number: 1



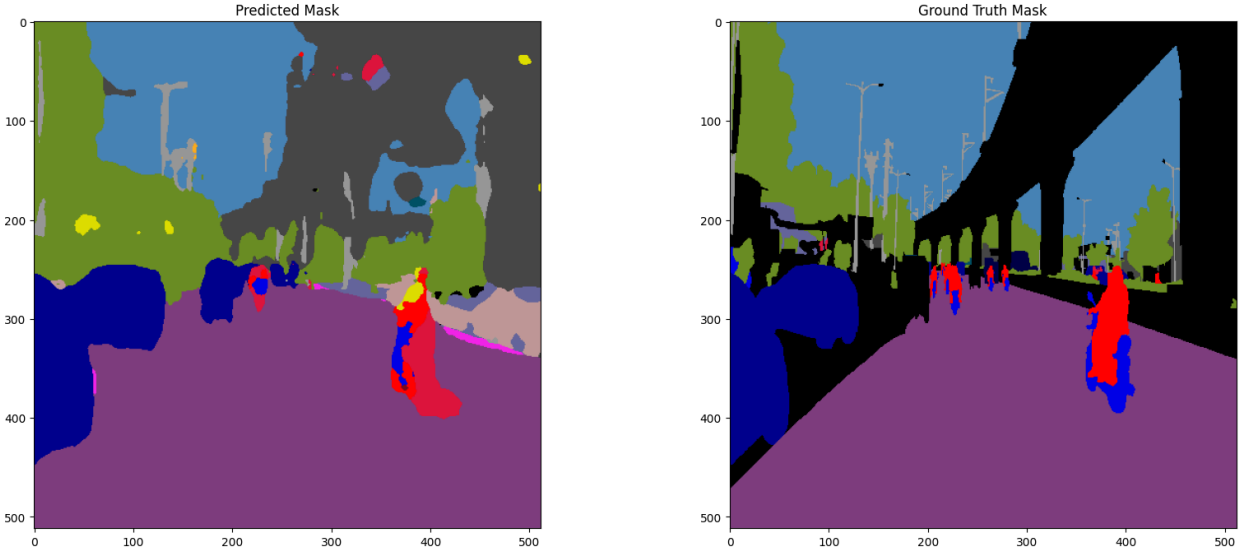
Class Name: person and Image Number: 1



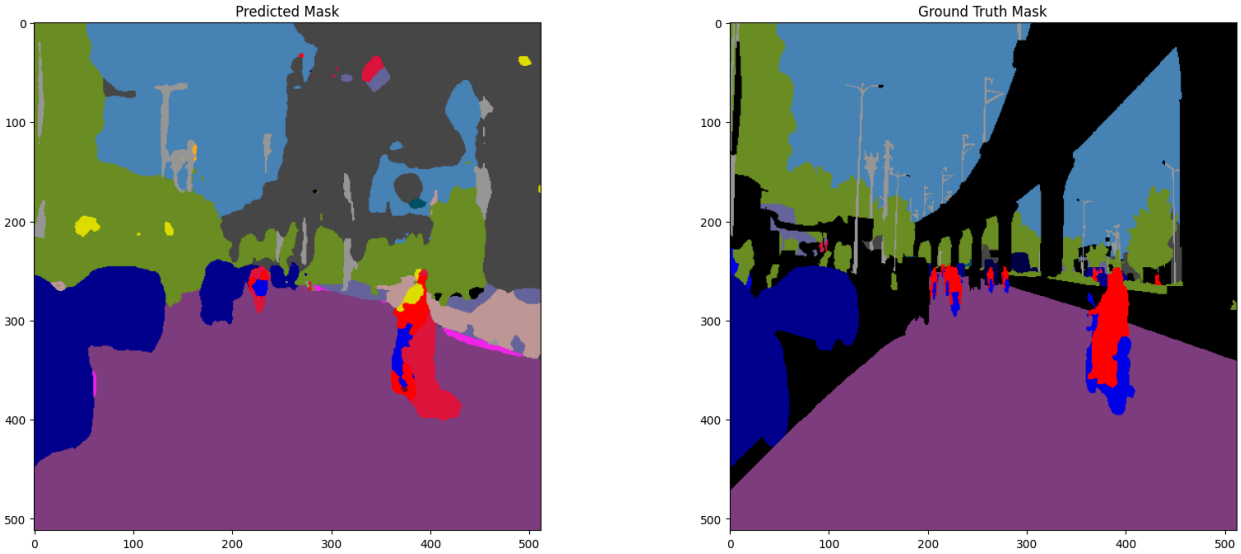
Class Name: rider and Image Number: 1



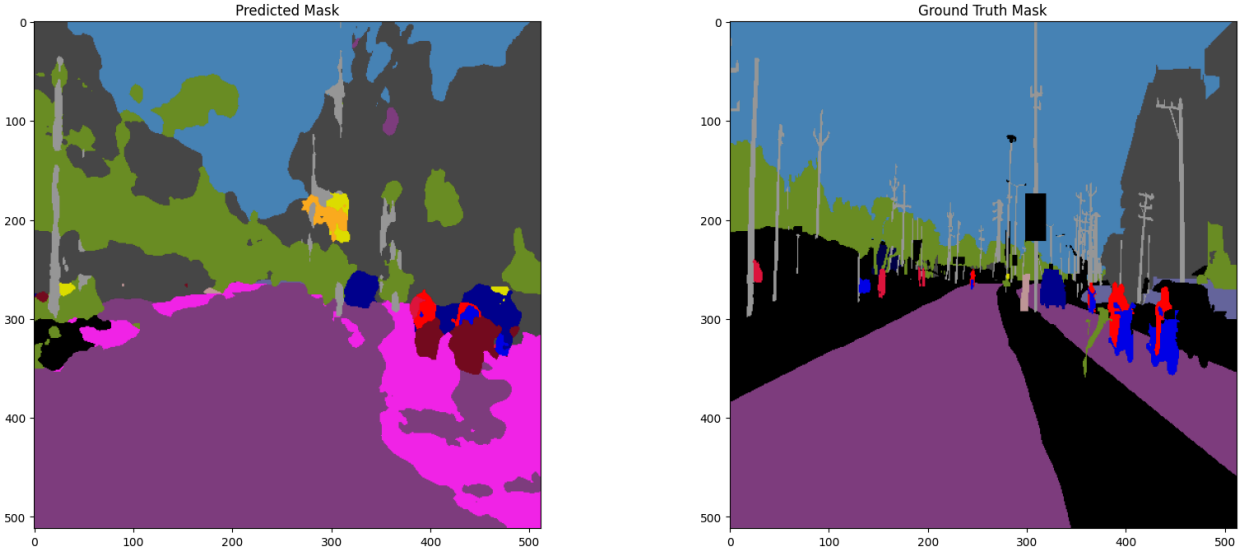
Class Name: motorcycle and Image Number: 1



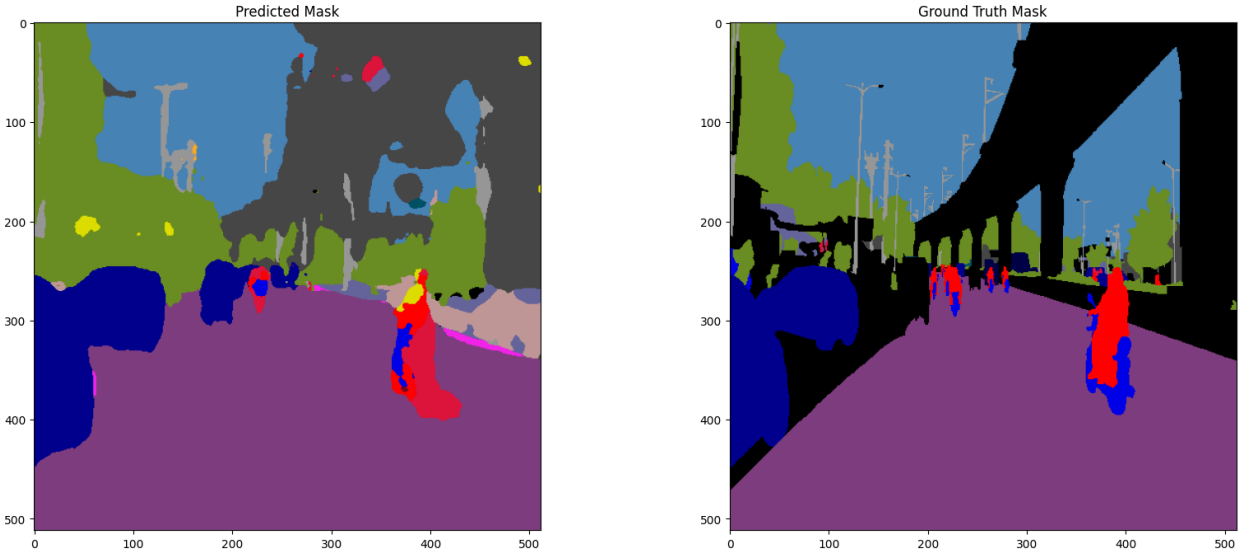
Class Name: bicycle and Image Number: 1



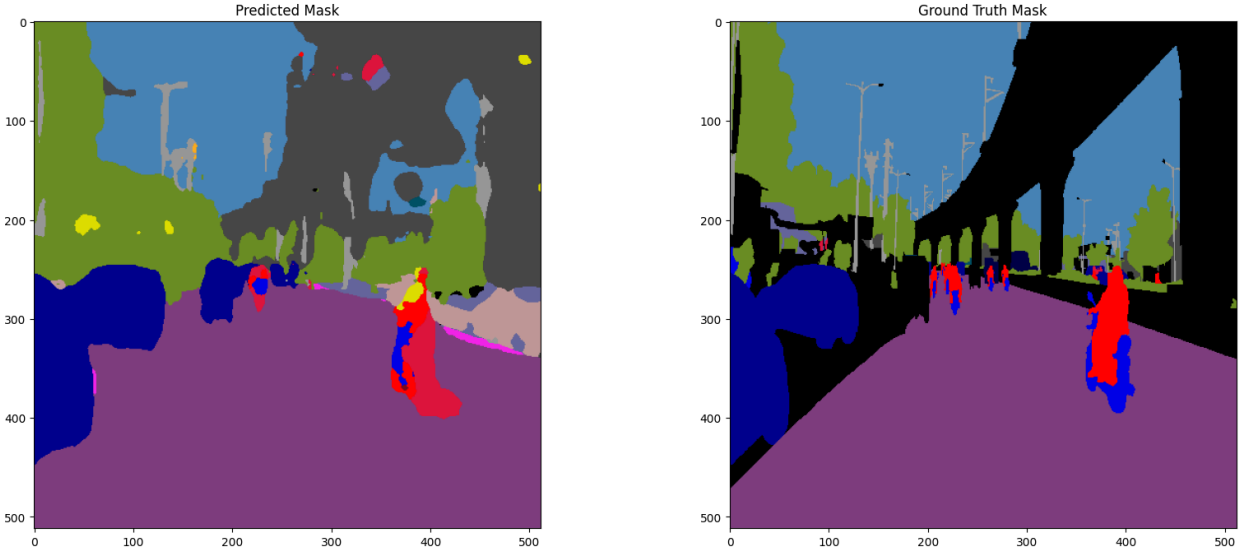
Class Name: car and Image Number: 1



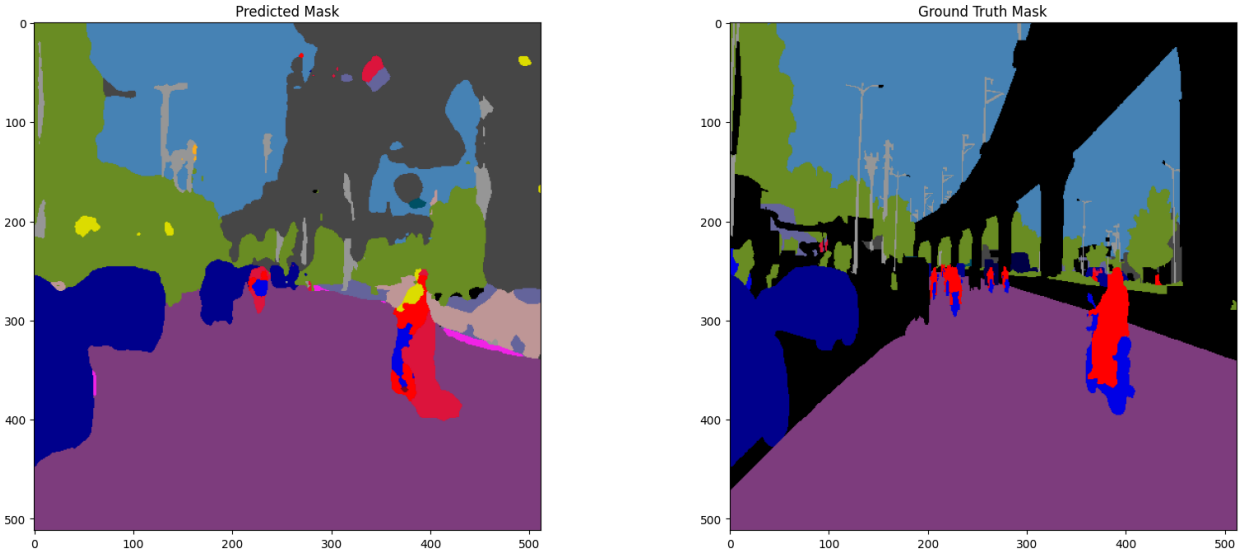
Class Name: truck and Image Number: 1

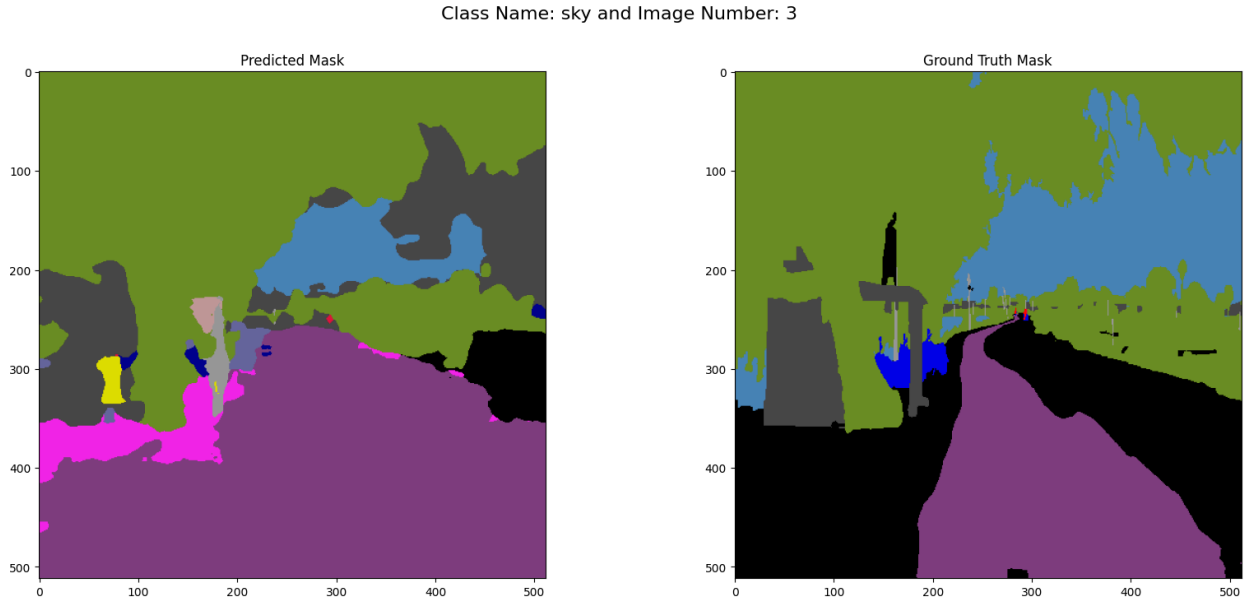


Class Name: bus and Image Number: 1



Class Name: train and Image Number: 1



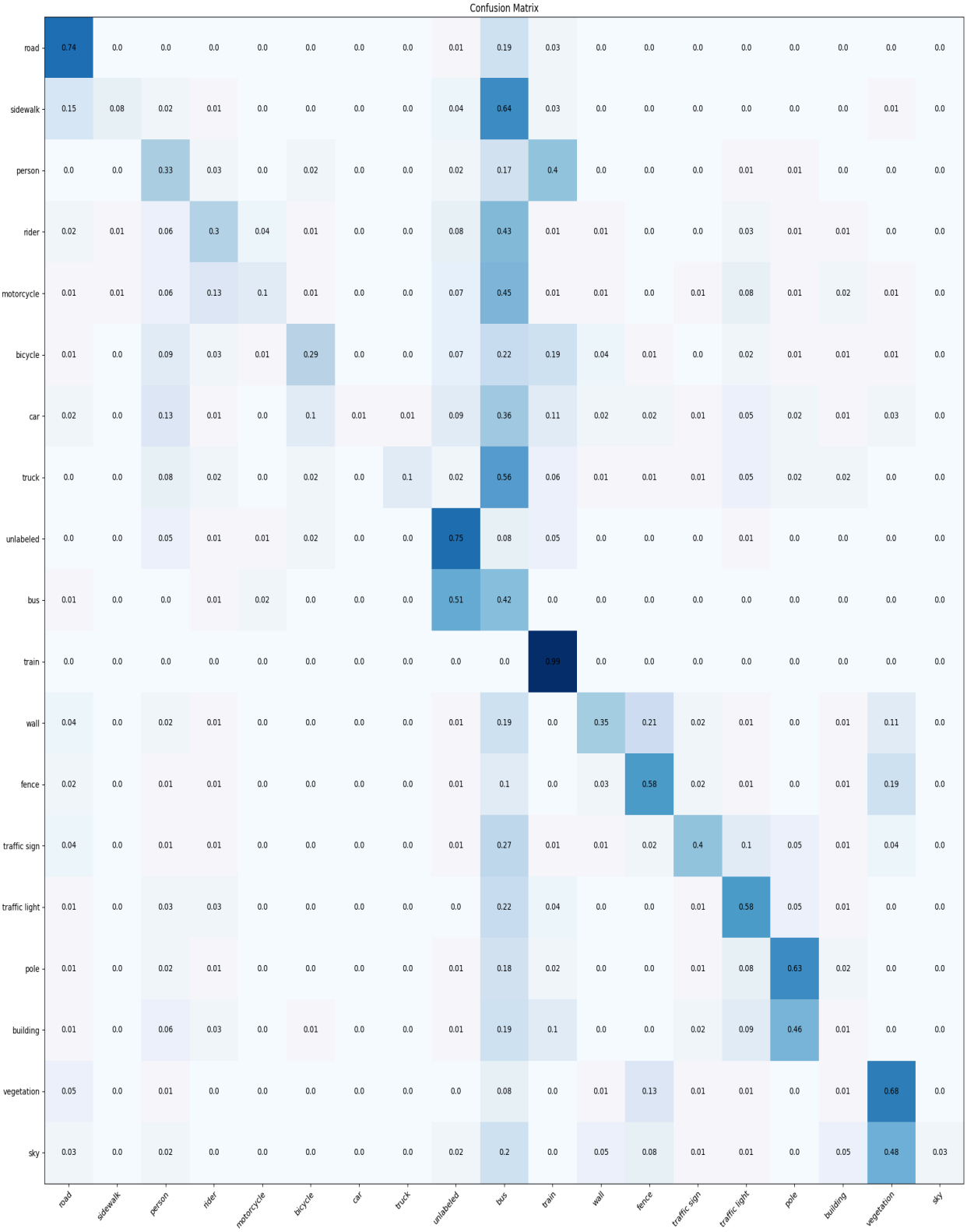


Through the above analysis, we can observe that there have been cases of occlusion of a class object, misclassificatoin and some other variations. For cases of object/class occlusion, occluded segments of the image may result in the struggle of the model to segment it accurately, leading to poor results. Misclassification of classes may also result in a lower IoU when the model assigns another class for the given class. For smaller classes, we can observe that smaller classes such as motorcycle, rider, person, etc. which take up less amount of pixels get easily misclassified or occluded, which result in a poor IoU value for these labels. Whereas, for labels which are present in a large quantity and widespread, such as vegetation, road, etc. the IoU values are significantly higher.

By analyzing these visualizations and considering contextual information, we can gain insights into why the model is failing in these specific cases and devise strategies to address these challenges, such as incorporating additional training data, adjusting model architecture, or implementing post-processing techniques.

3.3.(a) Confusion matrix is plotted as follows-

From the plotted confusion matrix, we can observe that road, vegetation, train, fence, and pole get significantly high values of accurate predictions, whereas labels like building, truck, car, motorcycle and sidewalk get lower values on the confusion matrix. It can also be observed that bus is the one of the most misclassified object, which is almost randomly being assigned to any label.


Confusion Matrix
road 0.74 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.01 0.19 0.03 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
sidewalk 0.15 0.08 0.02 0.01 0.0 0.0 0.0 0.0 0.04 0.64 0.03 0.0 0.0 0.0 0.0 0.0 0.0 0.01 0.0
person 0.0 0.0 0.33 0.03 0.0 0.02 0.0 0.0 0.02 0.17 0.4 0.0 0.0 0.0 0.01 0.01 0.0 0.0 0.0
rider 0.02 0.01 0.06 0.3 0.04 0.01 0.0 0.0 0.08 0.43 0.01 0.01 0.0 0.0 0.03 0.01 0.01 0.0 0.0
motorcycle 0.01 0.01 0.06 0.13 0.1 0.01 0.0 0.0 0.07 0.45 0.01 0.01 0.0 0.01 0.08 0.01 0.02 0.01 0.0
bicycle 0.01 0.0 0.09 0.03 0.01 0.29 0.0 0.0 0.07 0.22 0.19 0.04 0.01 0.0 0.02 0.01 0.01 0.01 0.0
car 0.02 0.0 0.13 0.01 0.0 0.1 0.01 0.01 0.09 0.36 0.11 0.02 0.02 0.01 0.05 0.02 0.01 0.03 0.0
truck 0.0 0.0 0.08 0.02 0.0 0.02 0.0 0.1 0.02 0.56 0.06 0.01 0.01 0.01 0.05 0.02 0.02 0.0 0.0
unlabeled 0.0 0.0 0.05 0.01 0.01 0.02 0.0 0.0 0.75 0.08 0.05 0.0 0.0 0.0 0.01 0.0 0.0 0.0 0.0
bus 0.01 0.0 0.0 0.01 0.02 0.0 0.0 0.0 0.51 0.42 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
train 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.99 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
wall 0.04 0.0 0.02 0.01 0.0 0.0 0.0 0.0 0.01 0.19 0.0 0.35 0.21 0.02 0.01 0.0 0.01 0.11 0.0
fence 0.02 0.0 0.01 0.01 0.0 0.0 0.0 0.0 0.01 0.1 0.0 0.03 0.58 0.02 0.01 0.0 0.01 0.19 0.0
traffic sign 0.04 0.0 0.01 0.01 0.0 0.0 0.0 0.0 0.01 0.27 0.01 0.01 0.02 0.4 0.1 0.05 0.01 0.04 0.0
traffic light 0.01 0.0 0.03 0.03 0.0 0.0 0.0 0.0 0.0 0.22 0.04 0.0 0.0 0.01 0.58 0.05 0.01 0.0 0.0
pole 0.01 0.0 0.02 0.01 0.0 0.0 0.0 0.0 0.01 0.18 0.02 0.0 0.0 0.01 0.08 0.63 0.02 0.0 0.0
building 0.01 0.0 0.06 0.03 0.0 0.01 0.0 0.0 0.01 0.19 0.1 0.0 0.0 0.02 0.09 0.46 0.01 0.0 0.0
vegetation 0.05 0.0 0.01 0.0 0.0 0.0 0.0 0.0 0.0 0.08 0.0 0.01 0.13 0.01 0.01 0.0 0.01 0.68 0.0
sky 0.03 0.0 0.02 0.0 0.0 0.0 0.0 0.0 0.02 0.2 0.0 0.05 0.08 0.01 0.01 0.0 0.05 0.48 0.03
road sidewalk person rider motorcycle bicycle car truck unlabeled bus train wall fence traffic sign traffic light pole building vegetation sky

3.3.(b)

Class Name	IoU	Precision	Recall	Dice Coefficient	Accuracy
road	0.72	0.74	0.95	0.82	0.9
vegetation	0.6	0.67	0.85	0.72	0.93
sky	0.54	0.95	0.54	0.65	0.9
car	0.28	0.3	0.58	0.35	0.97
building	0.24	0.27	0.56	0.33	0.86
pole	0.15	0.26	0.28	0.23	0.98
rider	0.11	0.26	0.15	0.16	0.99
person	0.1	0.16	0.17	0.14	0.99
motorcycle	0.09	0.31	0.12	0.14	0.99
sidewalk	0.05	0.06	0.11	0.07	0.97
wall	0.05	0.13	0.07	0.07	0.98
traffic sign	0.04	0.05	0.08	0.06	1
truck	0.02	0.1	0.02	0.03	0.99
bus	0.02	0.06	0.02	0.02	0.99
fence	0.02	0.04	0.04	0.03	0.99
bicycle	0.01	0.01	0.02	0.01	0.99
train	0	0	0	0	1
traffic light	0	0	0	0	1

Observations:

- Classes like "road," "vegetation," and "sky" have relatively higher IoU scores, suggesting that the model performs well in segmenting these classes.
- Classes like "train" and "traffic light" have IoU scores of 0, indicating that the model fails to detect or accurately segment these classes.
- Classes like "pole," "person," and "motorcycle" have higher precision values, indicating that the model makes fewer false positive predictions for these classes. However, these classes may have lower recall values, suggesting that the model misses some instances of these classes in the predictions.
- Classes like "car" and "building" have higher recall values, indicating that the model captures a higher fraction of true positive instances for these classes, but with lower precision.
- Classes like "road," "vegetation," and "sky" have relatively higher Dice coefficients, indicating better segmentation performance for these classes.

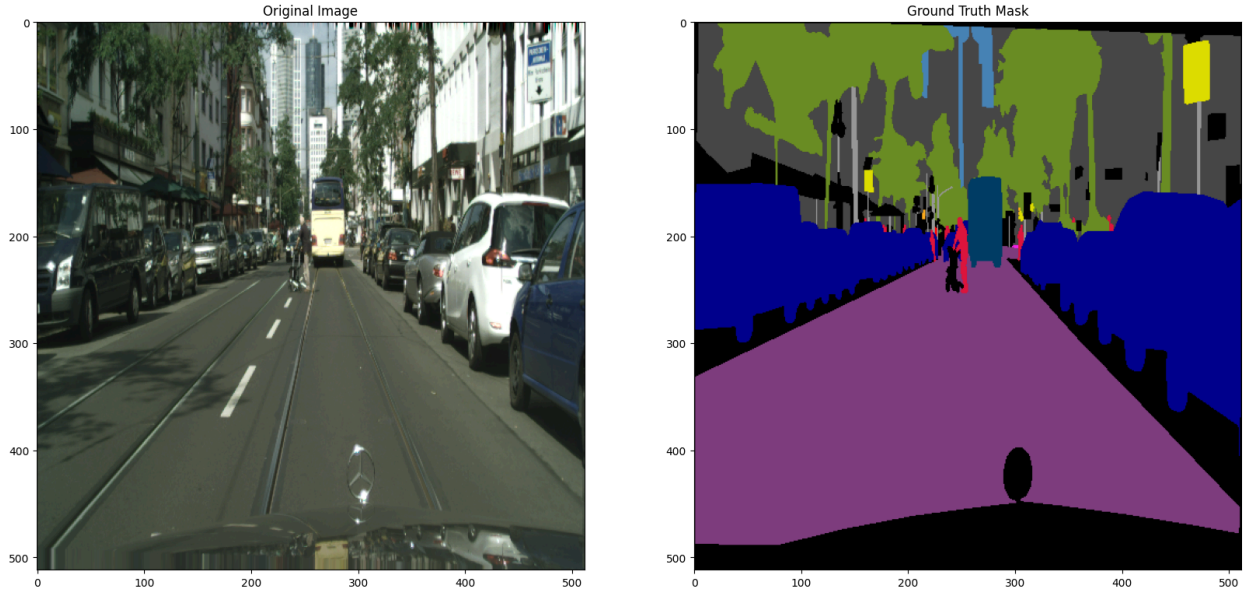
- Classes like "bus," "truck," "rider," and "person" have high accuracy, suggesting that the model performs well in correctly classifying these classes.

However, classes with fewer instances or smaller regions (e.g., "bicycle," "traffic sign") may have high accuracy but low IoU scores, indicating that the model struggles with precise segmentation for these classes.

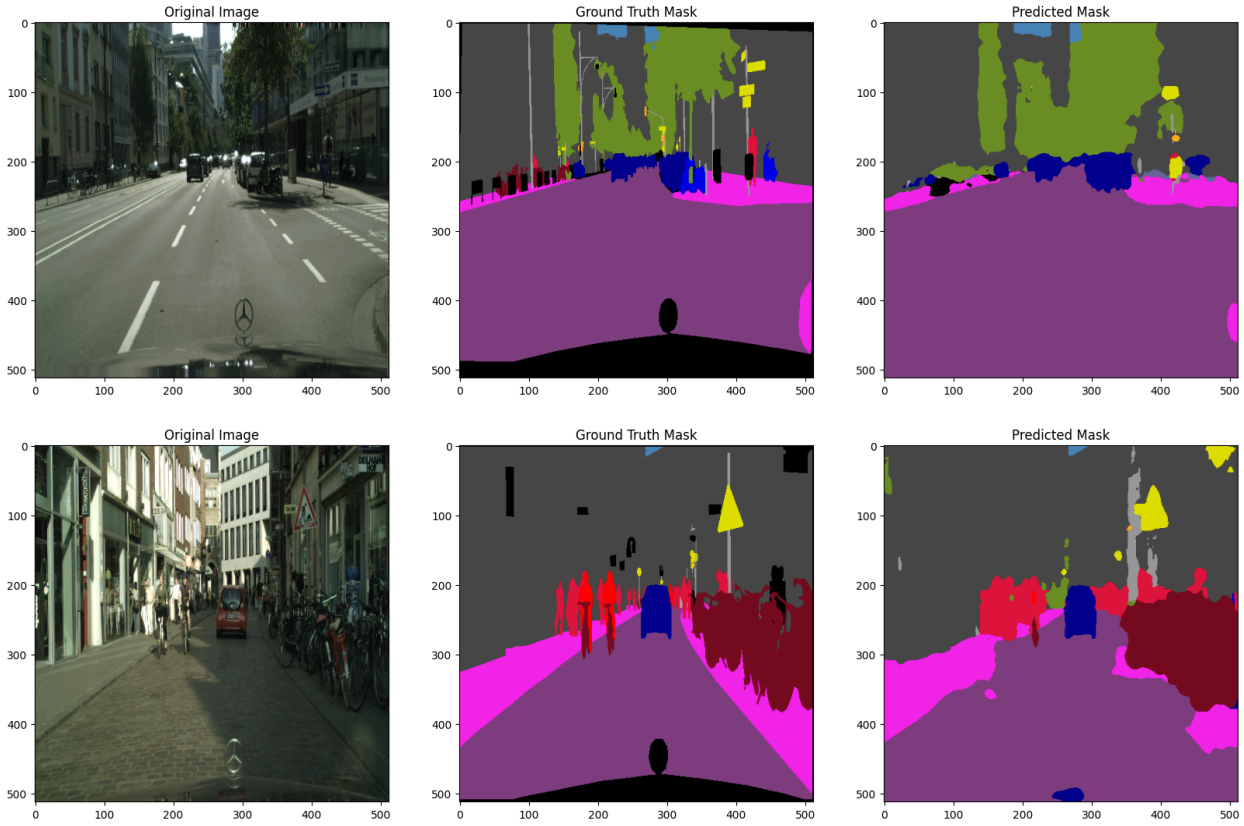
3.4 **(BONUS)**

3.4.(a) Inference is performed on the validation set of the Cityscapes dataset, where the images and masks are resized to a size of 512 x 512, followed by bringing the masks to the required masks dimension, and the unlabeled class assigned as “9”.

Data visualization of the cityscapes dataset is as follows-



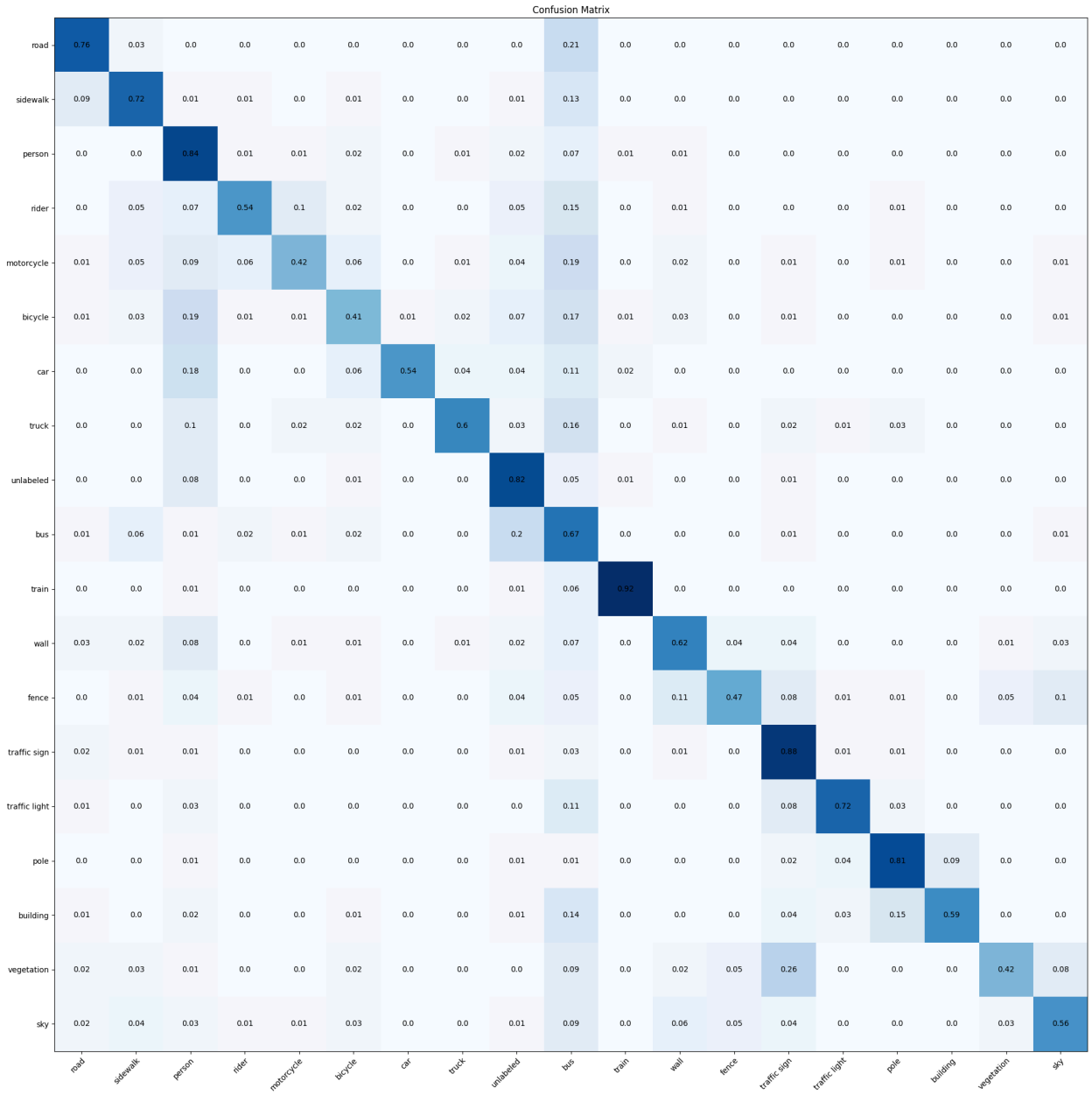
Qualitative analysis of few predictions by the DeepLabv3+ model are given as follows-



The metrics computed over the validation dataset (class-wise) are as obtained-

Class Name	IoU	Precision	Recall	Dice Coefficient	Accuracy	F1 Score
road	0.7422	0.7543	0.9455	0.8363	0.8906	0.8363
vegetation	0.7032	0.7368	0.9099	0.8002	0.9603	0.8001
building	0.6893	0.777	0.8238	0.7918	0.9477	0.7918
car	0.6686	0.7414	0.7972	0.754	0.9885	0.754
sky	0.5951	0.7254	0.6496	0.6647	0.9938	0.6647
sidewalk	0.4885	0.6305	0.6254	0.6013	0.973	0.6013
traffic sign	0.2695	0.4639	0.3567	0.3656	0.9952	0.3656
person	0.2072	0.3118	0.3088	0.2845	0.9917	0.2845
pole	0.2014	0.3735	0.3187	0.3113	0.9848	0.3113
bicycle	0.1619	0.2491	0.2624	0.2256	0.9946	0.2256
traffic light	0.0732	0.1763	0.098	0.1049	0.9984	0.1049
fence	0.0669	0.1249	0.0977	0.0925	0.9917	0.0925
rider	0.0632	0.1147	0.0922	0.0914	0.998	0.0914
wall	0.0558	0.1355	0.068	0.0776	0.9939	0.0776
bus	0.0351	0.0639	0.0374	0.0432	0.9979	0.0432
truck	0.0196	0.0471	0.0212	0.025	0.9981	0.025
motorcycle	0.0089	0.0277	0.012	0.0137	0.9993	0.0137
train	0.0065	0.0206	0.0072	0.0094	0.9991	0.0094

The confusion matrix is plotted as-



Through the confusion matrix, we can observe that the model classifies classes more accurately in the validation set of the cityscapes dataset compared to the IDD dataset, with most of the classes being classified with reasonable accuracy. Road, person, train, traffic sign, pole are classified well, whereas vegetation, bicycle and motorcycle are classified poorly by the model

3.4.(b)

Observing the metrics obtained for both the Indian Driving Dataset and the Cityscapes Dataset, several differences can be noted:

- The IoU scores for various classes differ between the two datasets. For instance, while the road class has an IoU of 0.72 in the Indian Driving Dataset, it is slightly higher at 0.7422 in the Cityscapes Dataset. Similar variations can be observed across other classes. On an average, it can be noted that the IoU scores of the classes are higher on the cityscapes dataset.
- Precision and recall values also exhibit differences between the two datasets. The vegetation class in the Indian Driving Dataset has a precision of 0.67 and a recall of 0.85, whereas in the Cityscapes Dataset, the precision is slightly higher at 0.7368, and the recall is also higher at 0.9099.
- The "building" class in the Indian Driving Dataset has a Dice coefficient of 0.33, whereas it is higher at 0.7918 in the Cityscapes Dataset.
- The accuracy values for both datasets vary, reflecting the overall performance of the models on the respective datasets. However, it is to be noted that accuracy is not a very relevant metric for this problem since most of the pixels have a False label, which evaluates to a true negative, and that boosts the accuracy value across classes, even though the other metrics might be lower.
- The confusion matrix obtained from the cityscapes dataset has significantly better results, with much higher correct classifications compared to the IDD dataset.

3.4.(c)

Based on the obtained metrics, the worst-performing class is train with an IoU of 0.0065 and the best-performing class is road with an IoU of 0.7422.

Train Class: (Worst class)

The "train" class has the lowest IoU, precision, recall, and other metrics among all classes. While train has a very high accuracy, that is primarily due to the true negative cases boosting the accuracy value, with all other metrics being significantly poor.

If we observe the true positive, false positive, and false negative counts for the "train" class, we find that the model struggles to correctly identify instances of trains in the images.

Possible reasons for poor performance could include:

- Limited diversity and quantity of train instances in the dataset, leading to inadequate model learning.
- Similar visual characteristics between trains and other objects in the scene, causing misclassification.
- Challenges in detecting and segmenting trains due to occlusions, variations in lighting conditions, or complex backgrounds.

Road Class: (Best Class)

The "road" class has the highest IoU, precision, recall, and other metrics among all classes.
Reasons for the great performance could include:

- Abundance of road instances in the dataset, providing ample training data for the model to learn accurate segmentation. Compared to classes like train, road would usually be occupying roughly 30-40% of any image's pixels, resulting in its better performance.
- Clear visual characteristics of roads, making them distinguishable from other classes with minimal confusion.