CV Assignment 1

Vikranth Udandarao
2022570

Theory

1-a- $H(p,q) = E_p[-\log q(u)]$

$$H(y,q) = -\sum_{i=1}^{K} y_i \log q_i \quad \text{(one hot encoded vector)}$$

$$= -\log q_i$$

Label smoothing $= (y \rightarrow \tilde{y})$ smooth label

$$\tilde{y}_{i=c} = 1 - \varepsilon + \frac{\varepsilon}{K}, \quad \text{if } c \text{ is correct class } (i=c)$$

$$\tilde{y}_{i \neq c} = \frac{\varepsilon}{K}, \quad \text{if it is incorrect class } (i \neq c)$$

$$H(\tilde{y},q) = -\sum_{i=1}^{K} \tilde{y}_i \log q_i$$

$$= -\sum_{i \neq c} \frac{\varepsilon}{K} \log q_i + (1-\varepsilon+\frac{\varepsilon}{K}) \log q_c$$

$$= -\left(\sum_{i \neq c} \frac{\varepsilon}{K} \log q_i + \log q_c - \varepsilon \log q_c + \frac{\varepsilon}{K} \log q_c\right)$$

$$= -\left(\log q_c - \varepsilon \log q_c + \frac{\varepsilon}{K} \sum_{i=1}^{K} \log q_i\right)$$

$$= -\left((1-\varepsilon)\log q_c + \frac{\varepsilon}{K} \sum_{i=1}^{K} \log q_i\right)$$

$$= -(\log q_c)(1-\varepsilon) - \frac{\varepsilon}{K} \sum_{i=1}^{K} \log q_i$$

$$= -(1-\varepsilon)(-H(y,q)) - \frac{\varepsilon}{K}\sum_{i=1}^{K} \log q_i$$

y: one hot encoded

$$= (1-\varepsilon)\, H(y,q) - \frac{\varepsilon}{K}\sum_{i=1}^{K} \log q_i$$

b- Label smoothing modifies the loss by introducing a factor of $(1-\varepsilon)$ which balances the true labels and encourages conservative predictions.

~~This regularization technique~~

This regularization discourages overfitting in predictions by preventing the model from assuming probabilities too close to 0 or 1. It is reducing the overconfidence in predictions indirectly.

It helps in reducing overfitting, the loss becomes softer thus giving limits of gradient explosion / vanishing and also fixes ambiguity or noise in labels, promoting smoother decision boundaries.

2- $p(x) = N(\mu_p, \sigma_p^2)$, $q(x) = N(\mu_q, \sigma_q^2)$

a- We know $H(p,q) = -E_{p(x)}[\log q(x)]$ ①

from ① q

$$q(x) = \frac{1}{\sqrt{2\pi}\,\sigma_q} e^{\left(\frac{-1}{2}\left(\frac{x-\mu}{\sigma_q}\right)^2\right)}$$

log on both sides →

$$\log(q(x)) = \frac{-1}{2}\log 2\pi\sigma_q^2 - \frac{1}{2}\left(\frac{x-\mu}{\sigma_q}\right)^2$$

Substituting in ①,

$$H(p,q) = \frac{1}{2}\log(2\pi\sigma_q^2) + \frac{1}{2\sigma_q^2} E_{p(x)}[(x-\mu_q)^2]$$

b- from ② → $H(p,q) = \ldots\ldots - \frac{1}{2\sigma_q^2} E_{p(x)}[(x-\mu_q)^2]$

$$E_{p(x)}[(x-\mu_q)^2] = E_{p(x)}[(x-\mu_p+\mu_p-\mu_q)^2]$$

$$= E_{p(x)}[((x-\mu_p)+(\mu_p-\mu_q))^2]$$

$$= E_{p(x)}[(x-\mu_p)^2] + E_{p(x)}[(\mu_p-\mu_q)^2] + E_{p(x)}[(\mu_p-\mu_q)(x-\mu_p)]$$

$$= \sigma_p^2 + (\mu_p-\mu_q)^2 + 2(\mu_p-\mu_q) + 0 \quad (x=\mu_p)$$

$$= \sigma_p^2 + (\mu_p-\mu_q)^2$$

$$\therefore H(p,q) = \frac{1}{2}\log(2\pi\sigma_q^2) + \frac{1}{2\sigma_q^2}(\sigma_p^2 + (\mu_p-\mu_q)^2)$$

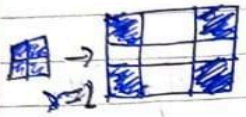
c- $\sigma_p = \sigma_q = \sigma$,

$$\rightarrow H(p,q) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{\sigma^2+(\mu_p-\mu_q)^2}{2\sigma^2}$$

$$= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2} + \frac{(\mu_p - \mu_q)}{2\sigma^2}$$

$$= \frac{1}{2}\left[(\log(2\pi\sigma^2) + 1) + \frac{(\mu_p - \mu_q)^2}{\sigma^2}\right] -$$

3 - a - In atrous convolution $r = 1 \rightarrow$ normal convolution, $r = 2 \rightarrow$ elements are spaced 1 pixel apart and so on -



, effective kernel size (receptive field)

$\rightarrow$ (kernel size -1) $r + 1$

$= (K-1)\, r + 1$ -

When using L such layers -

receptive field $\rightarrow$ $R_L = \sum_{i=1}^{L} (K-1) r_i + 1$ -

It's given that the dilation rate r increases exponentially which means $r_i = 2^{i-1}$. (geometric increase)

$\rightarrow$ $R_L = \sum_{i=1}^{L} (K-1)\, 2^{i-1} + 1$

$= 1 + (K-1)(2^L - 1)$ -

Since, it's a function of L of $\mathcal{O}(2^L)$, we can say that the receptive field grows exponentially -

b- $R = 1 + (K-1)(2^L - 1)$

$R_{area} = R_{height} \times R_{width}$

$= [1 + (K-1)(2^L - 1)]^2$ _)

(since $H = W$)

$K \times K \rightarrow$ square

now R_L changes as $f() \sim 4^L$, the area of receptive field grows quadratically exponentially.

c- for standard $K \times K$ convolution –

Let Input $= m \times n$ (ignoring channels for now)

No. of operations $= m \times n \times k^2$

for dilated $K \times K$ convolution –

Let Input $= m \times n$

No. of equations $= m \times n + k^2$ (we don't include holes)

→ We observe that dilation of the kernel in atrous convolution requires the same computational power as normal convolution but it covers a larger receptive field thus giving better global content for future interaction.

Coding

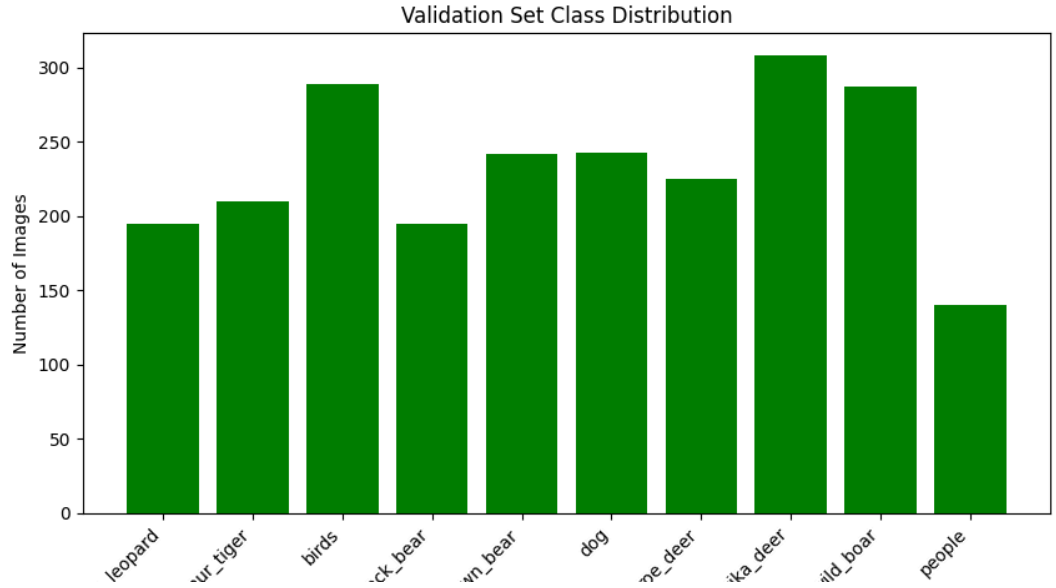
2. 1. a.

```python
# Custom Dataset Class

class WildlifeDataset(Dataset):
    def __init__(self, img_dir, class_mapping, transform=None, train=False):
        self.img_dir = img_dir
        self.transform = transform
        self.class_mapping = class_mapping
        self.train = train  # Flag to determine if this is training set
        self.img_labels = []

        # # Data augmentation for training
        # self.train_transform = transforms.Compose([
        #     transforms.ToPILImage(),
        #     transforms.RandomHorizontalFlip(p=0.5),  # Randomly flip image horizontally
        #     transforms.RandomRotation(degrees=15),    # Randomly rotate image up to 15 degrees
        #     transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),  # Adjust color properties
        #     transforms.Resize((224, 224)),
        #     transforms.ToTensor(),
        #     transforms.Normalize(mean=[0.485, 0.456, 0.406],
        #                          std=[0.229, 0.224, 0.225])
        # ])

        # Transform for validation (no augmentation)
        self.val_transform = transforms.Compose([
            transforms.ToPILImage(),
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
        ])

        # Collect image paths and labels
        for class_name in os.listdir(img_dir):
            class_path = os.path.join(img_dir, class_name)
            if os.path.isdir(class_path):
                for img_name in os.listdir(class_path):
                    img_path = os.path.join(class_path, img_name)
                    self.img_labels.append((img_path, class_mapping[class_name]))
```

b. None

```python
# Create dataloaders
batch_size = 64
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
```

c. None

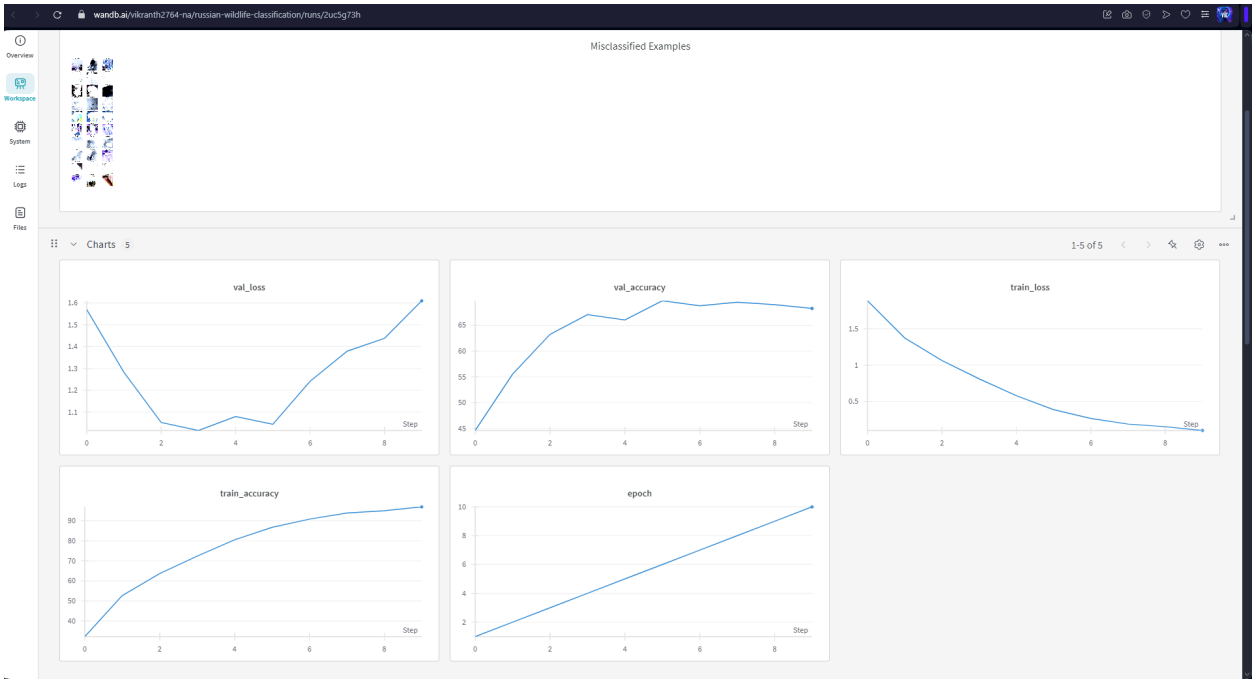

Validation Set Class Distribution
Number of Images
0
50
100
150
200
250
300
leopard
birds
dog
people

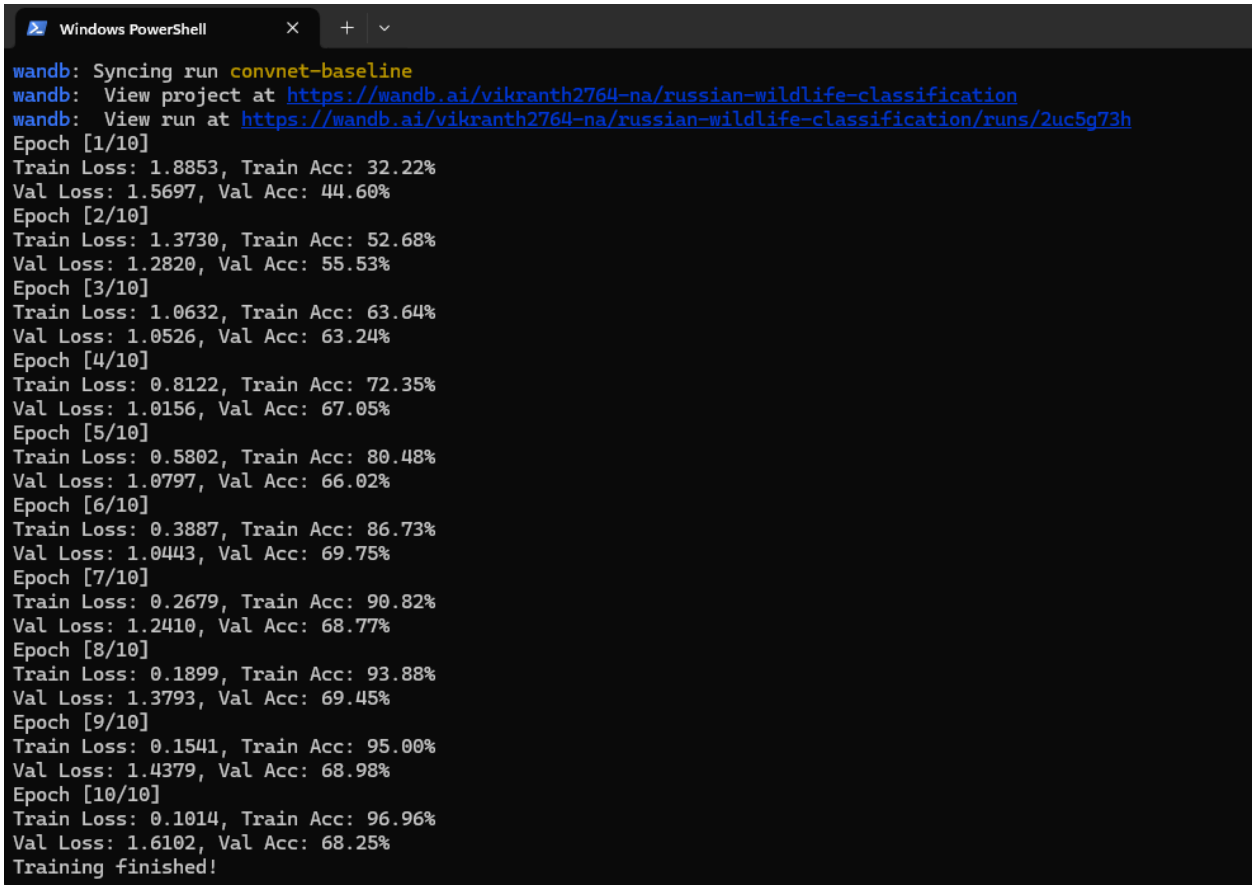

Training Set Class Distribution
Number of Images
0
200
400
600
800
1000
1200
leopard
birds
dog
people

2. a.

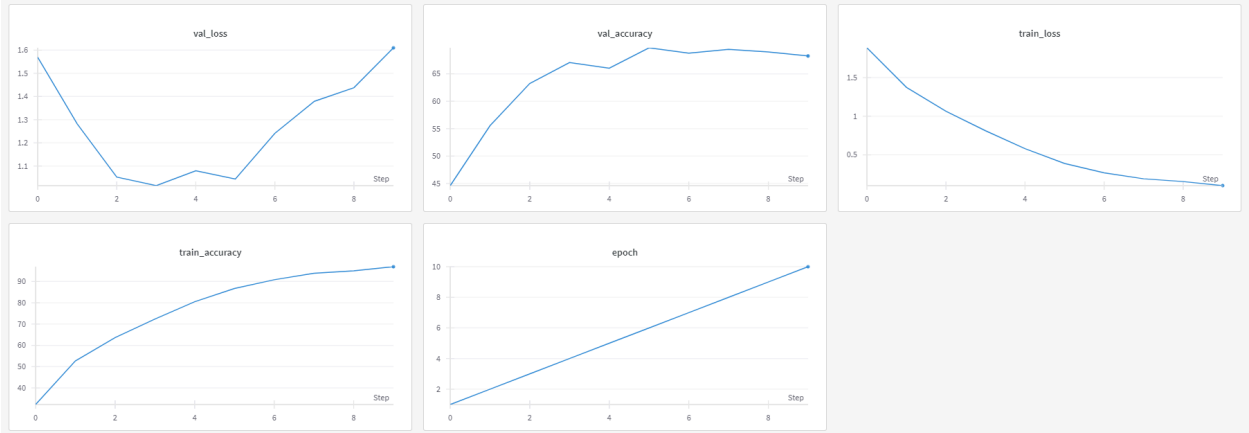
```
class ConvNet(nn.Module):
    def __init__(self, num_classes=10):
        super(ConvNet, self).__init__()

        # First Convolutional Block
        self.conv1 = nn.Conv2d(
            in_channels=3,          # RGB input
            out_channels=32,        # 32 feature maps
            kernel_size=3,          # 3x3 kernel
            stride=1,
            padding=1
        )
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=4, stride=4)

        # Second Convolutional Block
        self.conv2 = nn.Conv2d(
            in_channels=32,         # Input from previous layer
            out_channels=64,        # 64 feature maps
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Third Convolutional Block
        self.conv3 = nn.Conv2d(
            in_channels=64,         # Input from previous layer
            out_channels=128,       # 128 feature maps
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.relu3 = nn.ReLU()
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Classification head
        self.fc1 = nn.Linear(128 * 14 * 14, 256)  # Adjust size based on pooling
        self.relu4 = nn.ReLU()
        self.fc2 = nn.Linear(256, num_classes)  # Output layer
```

b.


wandb.ai/vikranth2764-na/russian-wildlife-classification/runs/2uc5g73h
Overview
Workspace
System
Logs
Files
Misclassified Examples
Charts 5
1-5 of 5
val_loss
val_accuracy
train_loss
train_accuracy
epoch
Step


Windows PowerShell
wandb: Syncing run convnet-baseline
wandb: View project at https://wandb.ai/vikranth2764-na/russian-wildlife-classification
wandb: View run at https://wandb.ai/vikranth2764-na/russian-wildlife-classification/runs/2uc5g73h
Epoch [1/10]
Train Loss: 1.8853, Train Acc: 32.22%
Val Loss: 1.5697, Val Acc: 44.60%
Epoch [2/10]
Train Loss: 1.3730, Train Acc: 52.68%
Val Loss: 1.2820, Val Acc: 55.53%
Epoch [3/10]
Train Loss: 1.0632, Train Acc: 63.64%
Val Loss: 1.0526, Val Acc: 63.24%
Epoch [4/10]
Train Loss: 0.8122, Train Acc: 72.35%
Val Loss: 1.0156, Val Acc: 67.05%
Epoch [5/10]
Train Loss: 0.5802, Train Acc: 80.48%
Val Loss: 1.0797, Val Acc: 66.02%
Epoch [6/10]
Train Loss: 0.3887, Train Acc: 86.73%
Val Loss: 1.0443, Val Acc: 69.75%
Epoch [7/10]
Train Loss: 0.2679, Train Acc: 90.82%
Val Loss: 1.2410, Val Acc: 68.77%
Epoch [8/10]
Train Loss: 0.1899, Train Acc: 93.88%
Val Loss: 1.3793, Val Acc: 69.45%
Epoch [9/10]
Train Loss: 0.1541, Train Acc: 95.00%
Val Loss: 1.4379, Val Acc: 68.98%
Epoch [10/10]
Train Loss: 0.1014, Train Acc: 96.96%
Val Loss: 1.6102, Val Acc: 68.25%
Training finished!

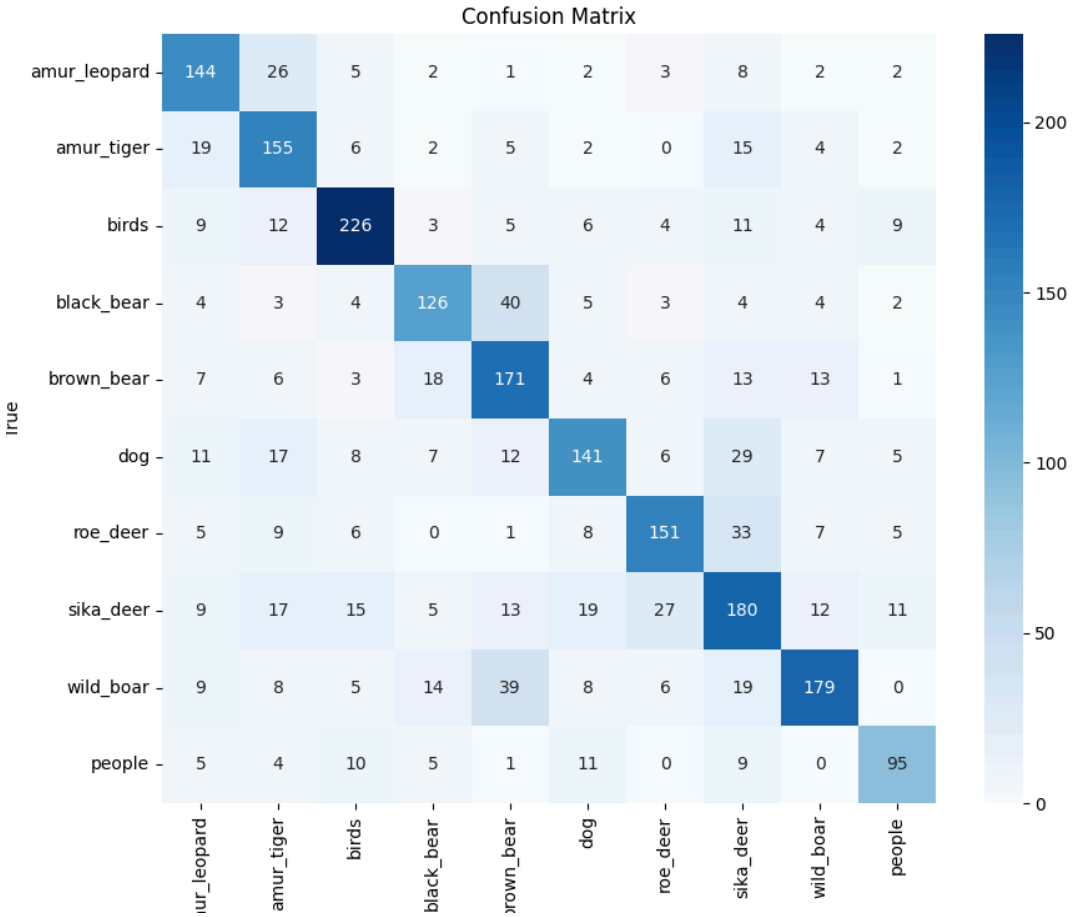
c.



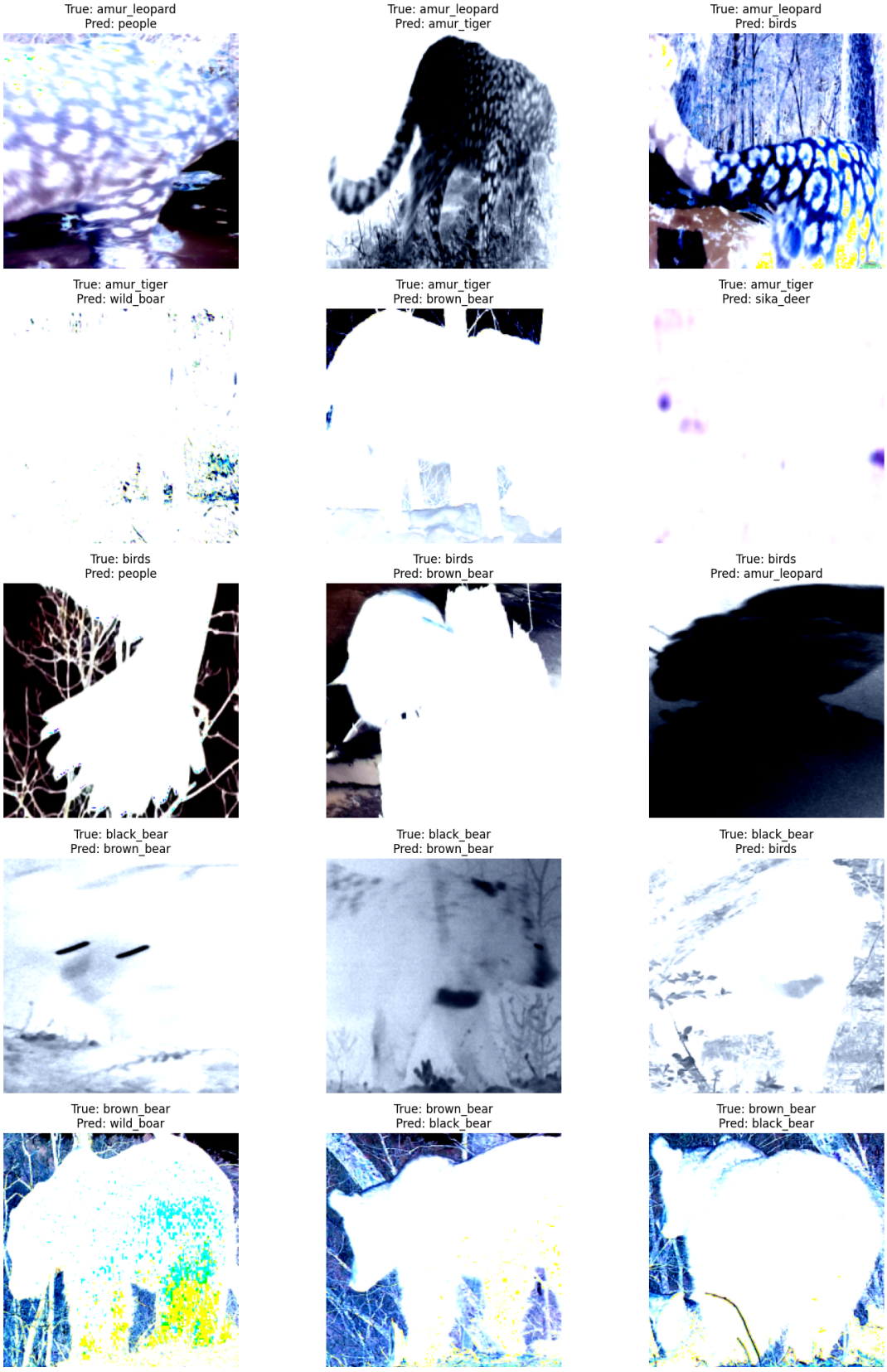
Yes model is overfitting as Train Accuracy is 96.96% whereas Validation Accuracy is only 68.25%.

d.

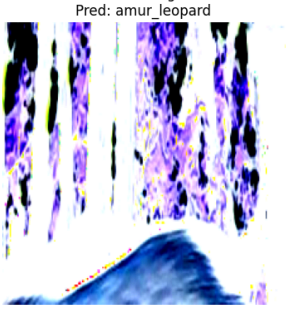
```
Validation Metrics:
Accuracy: 0.6718
F1-Score: 0.6724
```



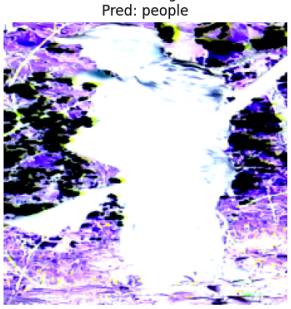
e.


True: amur_leopard
Pred: people
True: amur_leopard
Pred: amur_tiger
True: amur_leopard
Pred: birds
True: amur_tiger
Pred: wild_boar
True: amur_tiger
Pred: brown_bear
True: amur_tiger
Pred: sika_deer
True: birds
Pred: people
True: birds
Pred: brown_bear
True: birds
Pred: amur_leopard
True: black_bear
Pred: brown_bear
True: black_bear
Pred: brown_bear
True: black_bear
Pred: birds
True: brown_bear
Pred: wild_boar
True: brown_bear
Pred: black_bear
True: brown_bear
Pred: black_bear

True: dog
Pred: amur_leopard



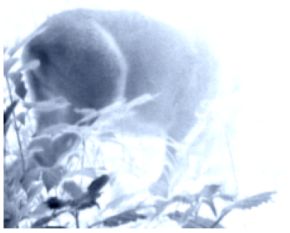
True: dog
Pred: people



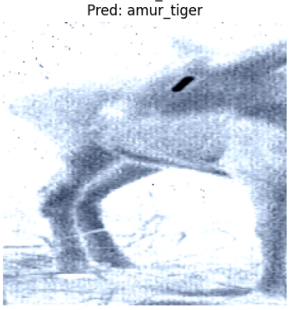
True: dog
Pred: roe_deer

True: roe_deer
Pred: sika_deer

True: roe_deer
Pred: sika_deer



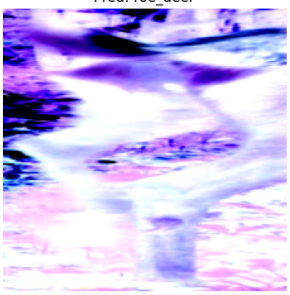
True: roe_deer
Pred: amur_tiger



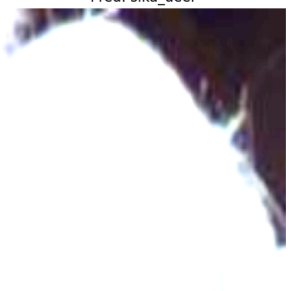
True: sika_deer
Pred: dog

True: sika_deer
Pred: amur_tiger

True: sika_deer
Pred: roe_deer

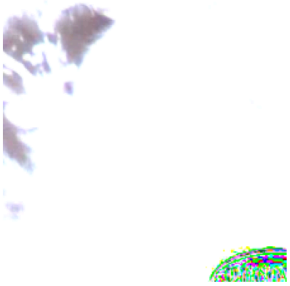


True: wild_boar
Pred: sika_deer

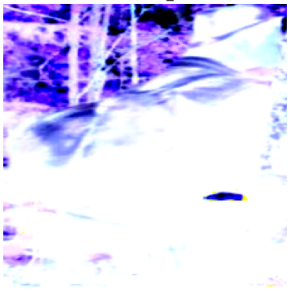


True: wild_boar
Pred: birds

True: wild_boar
Pred: sika_deer



True: people
Pred: brown_bear



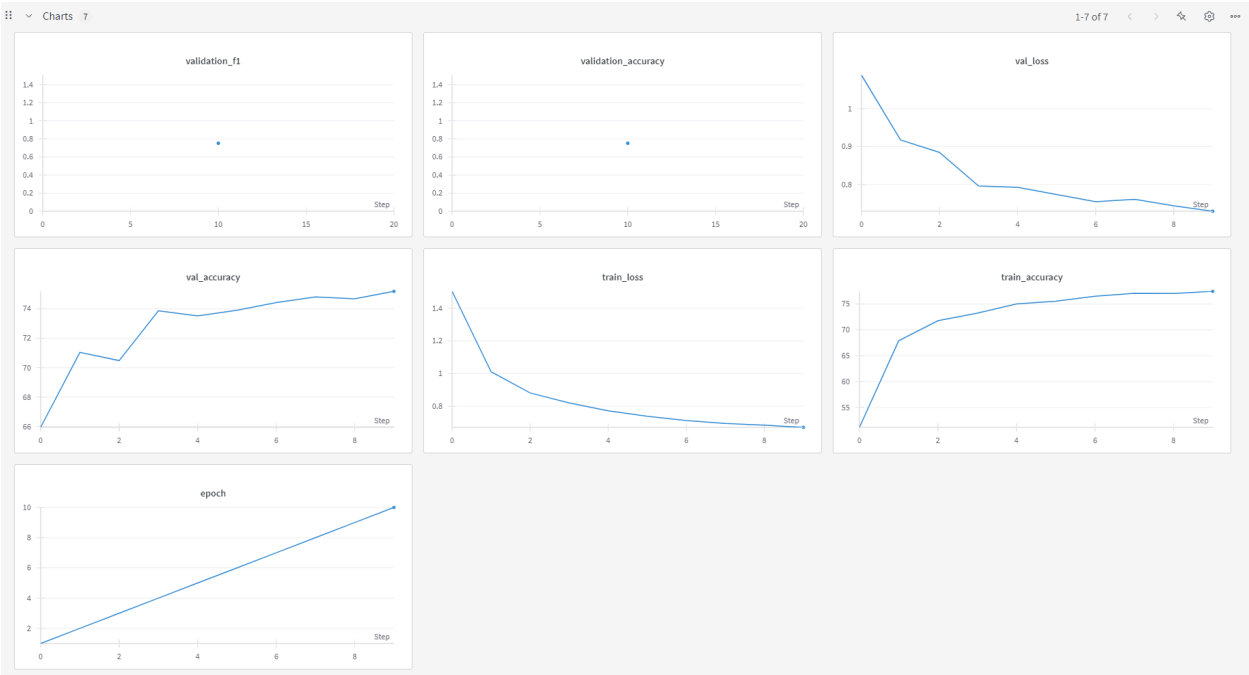
True: people
Pred: sika_deer

True: people
Pred: amur_leopard

Visual Similarity: Many misclassified images belong to visually similar categories (e.g., "amur_leopard" vs. "amur_tiger" and "black_bear" vs. "brown_bear").
Partial Occlusion: In certain cases, only a part of the object is visible, leading the model to misinterpret it.

Data Augmentation: We can try to Introduce transformations such as contrast adjustments, brightness corrections, and different lighting conditions.
Feature Engineering: Extract more discriminative features like texture patterns specific to each class

3. a.



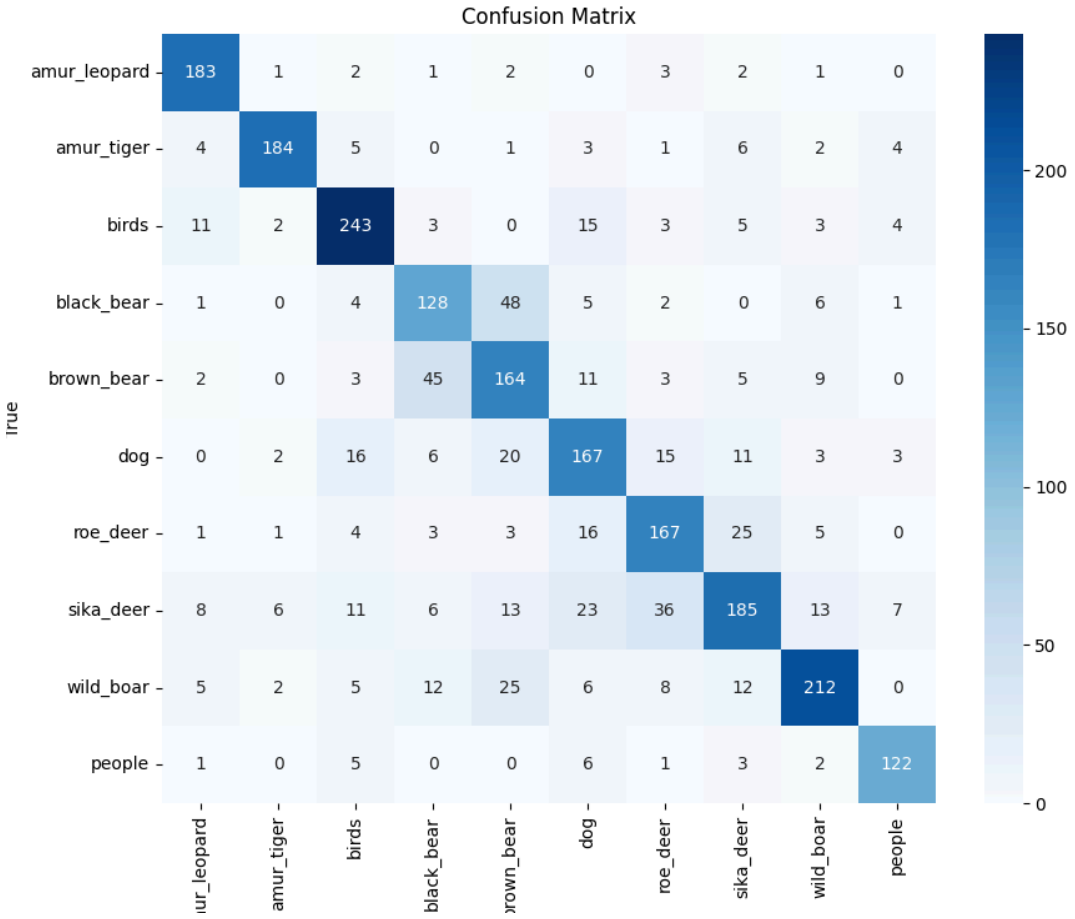
b.

```
Epoch [1/10]
Train Loss: 1.5037, Train Acc: 51.12%
Val Loss: 1.0882, Val Acc: 65.94%
Epoch [2/10]
Train Loss: 1.0110, Train Acc: 67.88%
Val Loss: 0.9175, Val Acc: 71.04%
Epoch [3/10]
Train Loss: 0.8819, Train Acc: 71.77%
Val Loss: 0.8846, Val Acc: 70.48%
Epoch [4/10]
Train Loss: 0.8208, Train Acc: 73.22%
Val Loss: 0.7960, Val Acc: 73.86%
Epoch [5/10]
Train Loss: 0.7730, Train Acc: 75.02%
Val Loss: 0.7925, Val Acc: 73.52%
Epoch [6/10]
Train Loss: 0.7394, Train Acc: 75.54%
Val Loss: 0.7737, Val Acc: 73.91%
Epoch [7/10]
Train Loss: 0.7131, Train Acc: 76.49%
Val Loss: 0.7549, Val Acc: 74.42%
Epoch [8/10]
Train Loss: 0.6949, Train Acc: 77.07%
Val Loss: 0.7608, Val Acc: 74.81%
Epoch [9/10]
Train Loss: 0.6851, Train Acc: 77.03%
Val Loss: 0.7439, Val Acc: 74.68%
Epoch [10/10]
Train Loss: 0.6713, Train Acc: 77.46%
Val Loss: 0.7295, Val Acc: 75.19%
```

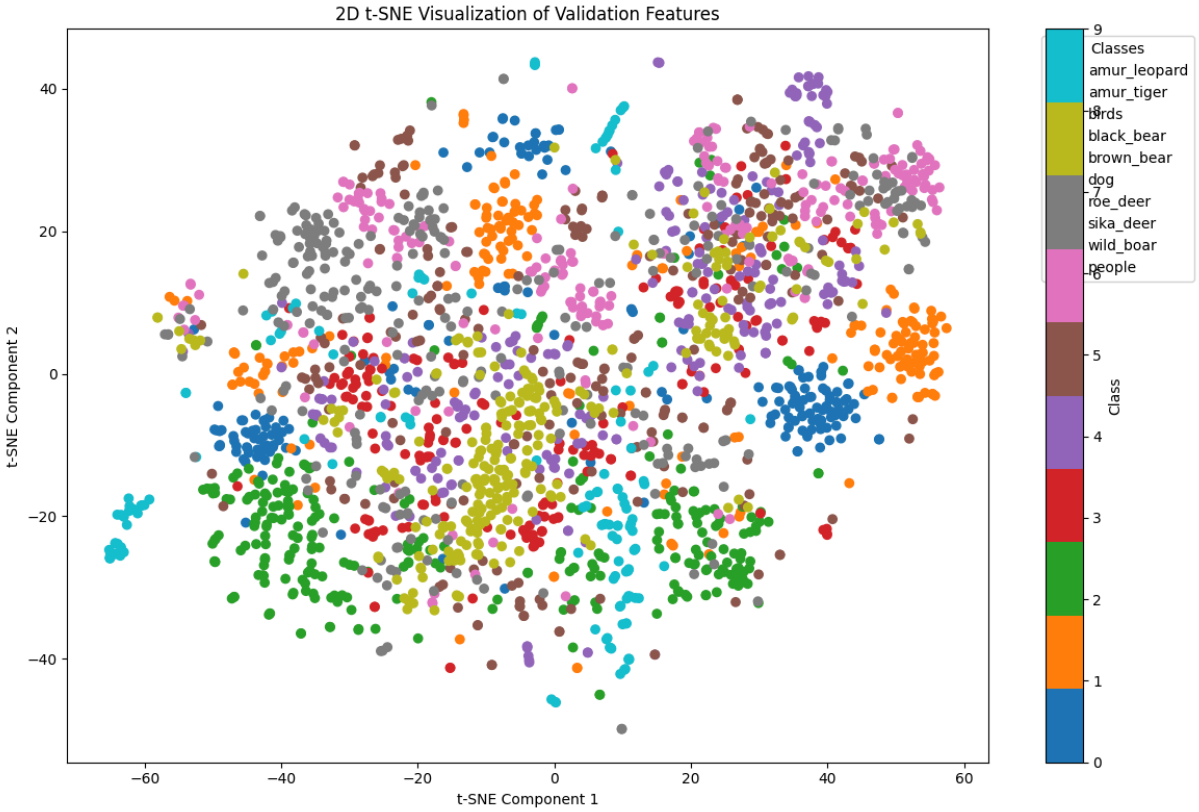
No, the model is not overfitting as training ad validation accuracy is almost similar with not much difference between them and training accuracy is not very very higher compared to validation accuracy.

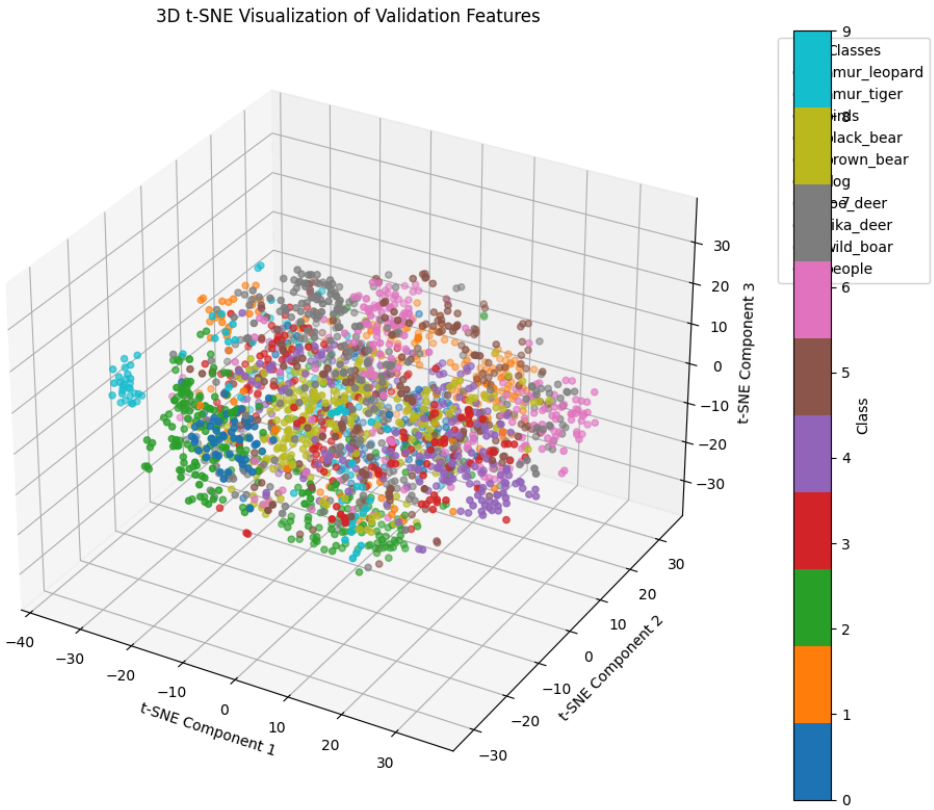
C.

```
Validation Metrics:
Accuracy: 0.7519
F1-Score: 0.7520
Training finished!
```



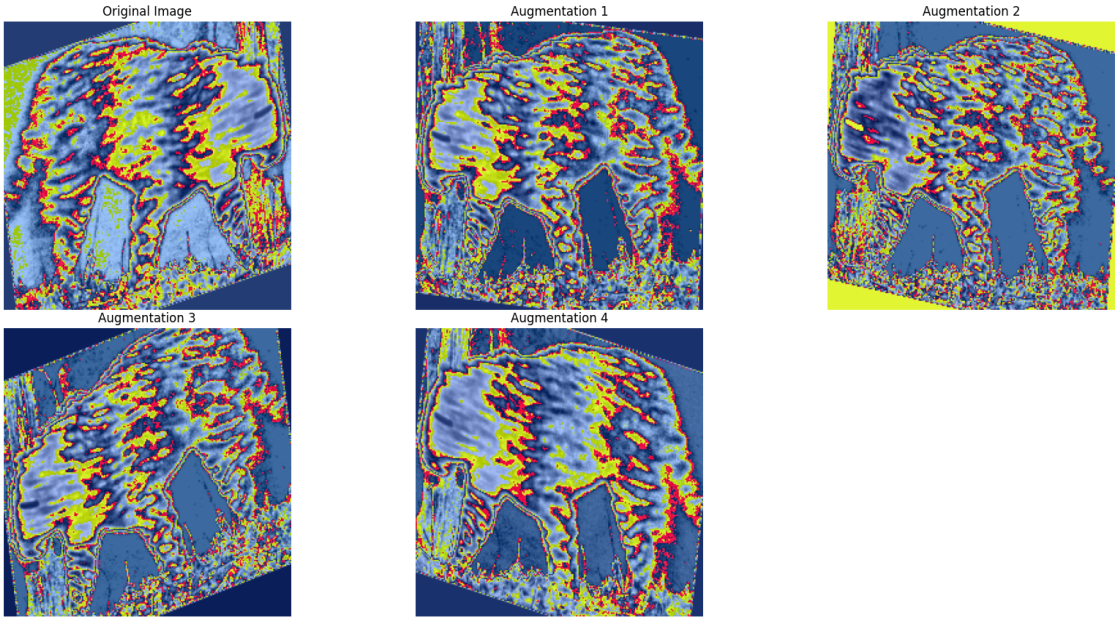
d.


2D t-SNE Visualization of Validation Features
t-SNE Component 1
t-SNE Component 2
Class
Classes
amur_leopard
amur_tiger
birds
black_bear
brown_bear
dog
roe_deer
sika_deer
wild_boar
people

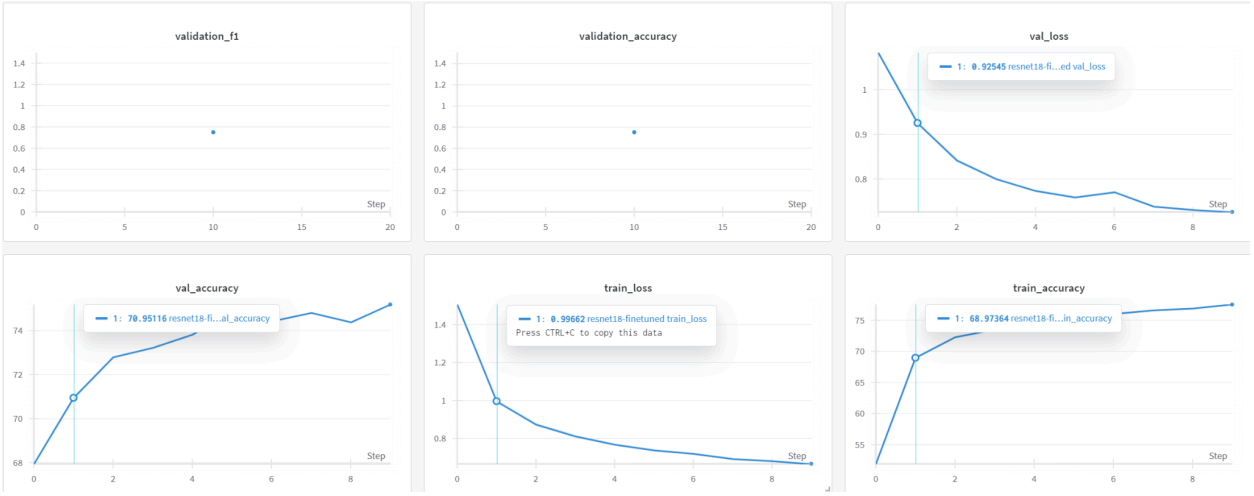

3D t-SNE Visualization of Validation Features
t-SNE Component 1
t-SNE Component 2
t-SNE Component 3
Class
Classes
mur_leopard
mur_tiger
ds
lack_bear
rown_bear
og
7_deer
ika_deer
ild_boar
eople

4. a.

```python
# Use this for data augmentation for resnet augmentation
transform = transforms.Compose([
transforms.ToPILImage(),
transforms.RandomHorizontalFlip(p=0.5),  # Flip images horizontally with 50% probability
transforms.RandomRotation(degrees=15),   # Rotate images randomly within ±15 degrees
transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),  # can adjust brightness, contrast, saturation but as of now I putt 0.2 for all
transforms.Resize((224, 224)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406],
                     std=[0.229, 0.224, 0.225])  # ImageNet normalization
])
```



b.



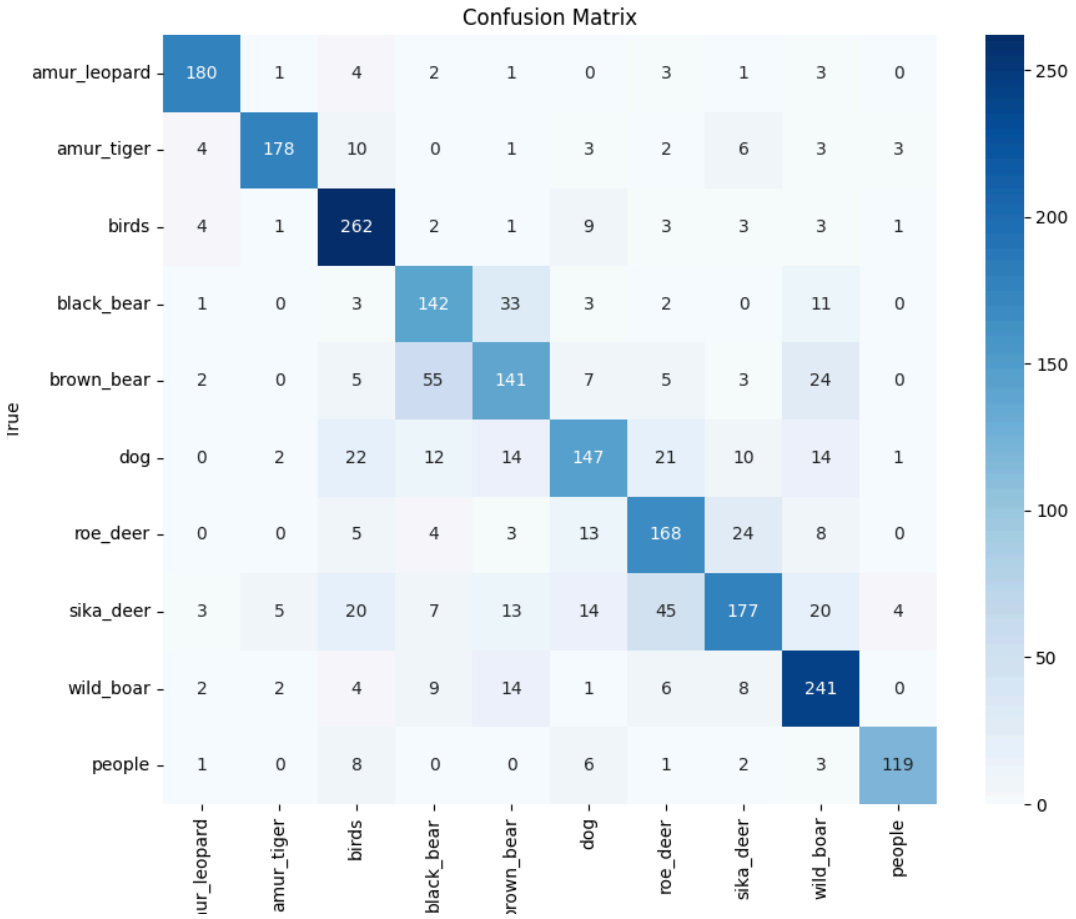
c.

```
Epoch [1/10]
Train Loss: 1.5065, Train Acc: 51.92%
Val Loss: 1.0830, Val Acc: 67.95%
Epoch [2/10]
Train Loss: 0.9966, Train Acc: 68.97%
Val Loss: 0.9254, Val Acc: 70.95%
Epoch [3/10]
Train Loss: 0.8748, Train Acc: 72.28%
Val Loss: 0.8420, Val Acc: 72.79%
Epoch [4/10]
Train Loss: 0.8127, Train Acc: 73.52%
Val Loss: 0.8003, Val Acc: 73.22%
Epoch [5/10]
Train Loss: 0.7690, Train Acc: 74.56%
Val Loss: 0.7740, Val Acc: 73.82%
Epoch [6/10]
Train Loss: 0.7391, Train Acc: 75.48%
Val Loss: 0.7596, Val Acc: 74.85%
Epoch [7/10]
Train Loss: 0.7209, Train Acc: 76.02%
Val Loss: 0.7711, Val Acc: 74.42%
Epoch [8/10]
Train Loss: 0.6935, Train Acc: 76.59%
Val Loss: 0.7389, Val Acc: 74.81%
Epoch [9/10]
Train Loss: 0.6825, Train Acc: 76.89%
Val Loss: 0.7317, Val Acc: 74.38%
Epoch [10/10]
Train Loss: 0.6663, Train Acc: 77.56%
Val Loss: 0.7262, Val Acc: 75.19%
```

Yes, now the model is not overfitting, there is not high variance like how Convnet had.

d.

```
Validation Metrics:
Accuracy: 0.7519
F1-Score: 0.7497
Training finished!
```

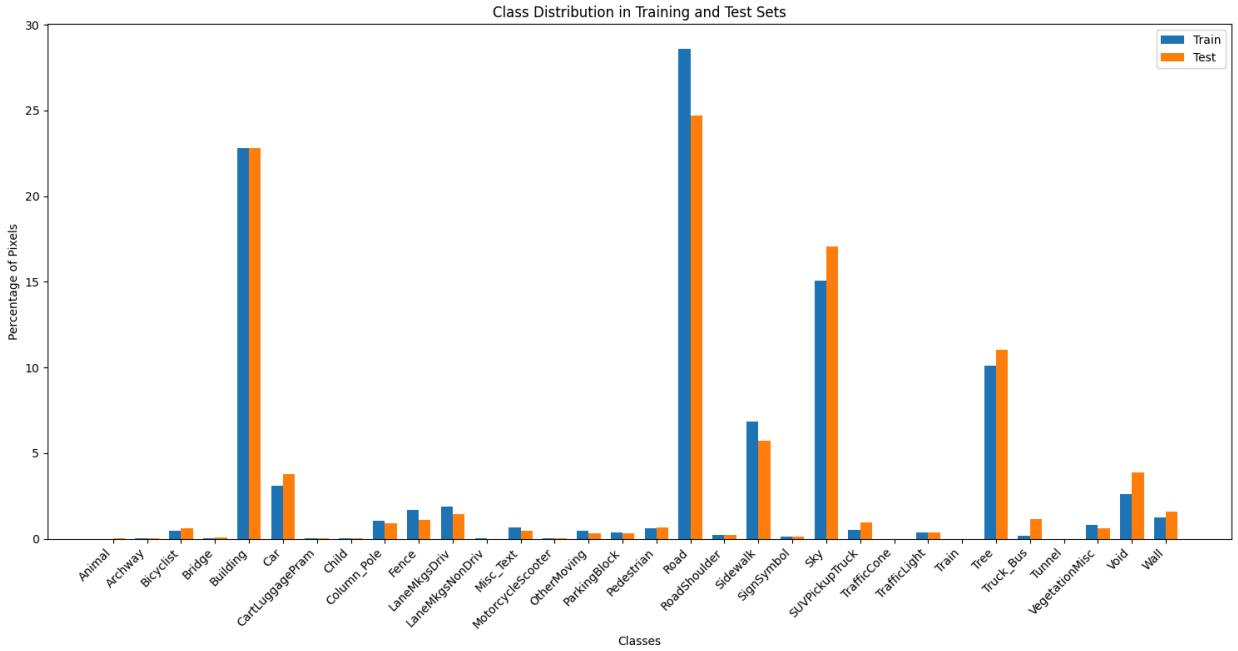


5. Convnet model is highly overfitting and has high variance and low bias and only has 68% validation accuracy.
Resnet model does not overfit and has low variance and both training and validation accuracy is around 75% accuracy.
Similarly for resnet model with data augmentation techniques, I did not get much different results from resnet.
Maybe this is because of the data augmentation techniques I am doing as mentioned in 4.a.
I am using 0.2 for brightness, contrast, saturation and only rotating the images by 15°.

3. 1. a.

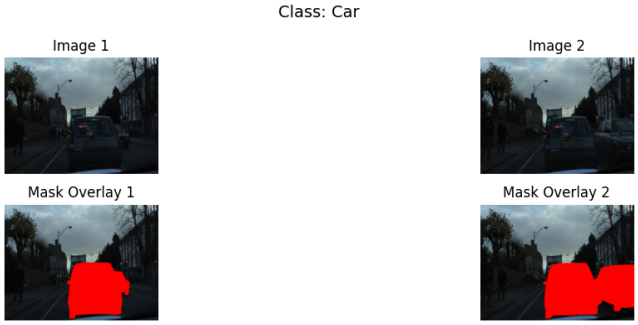
```python
class SegmentationDataset(Dataset):
    def __init__(self, data_dir, split='train', transform=None):
        self.data_dir = data_dir
        self.split = split
        self.transform = transform
        self.class_mapping = CLASS_MAPPING

        # Get all image and mask files based on split
        if split == 'train':
            self.images_dir = os.path.join(data_dir, 'train')
            self.masks_dir = os.path.join(data_dir, 'train_labels')
        else:  # test
            self.images_dir = os.path.join(data_dir, 'test_images')
            self.masks_dir = os.path.join(data_dir, 'test_labels')

        self.image_files = sorted([f for f in os.listdir(self.images_dir)])
        self.mask_files = sorted([f for f in os.listdir(self.masks_dir)])

        # Transformation according to what is said in 1.a. for both image and mask
        self.resize = transforms.Resize((360, 480), interpolation=transforms.InterpolationMode.BILINEAR)
        self.resize_mask = transforms.Resize((360, 480), interpolation=transforms.InterpolationMode.NEAREST)
        self.normalize = transforms.Normalize(
            mean=[0.485, 0.456, 0.406],
            std=[0.229, 0.224, 0.225]
        )
```

b.

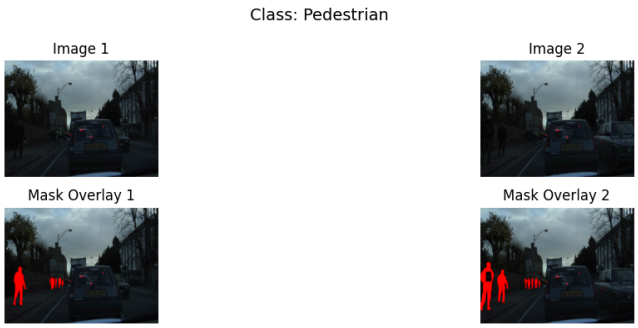


c.

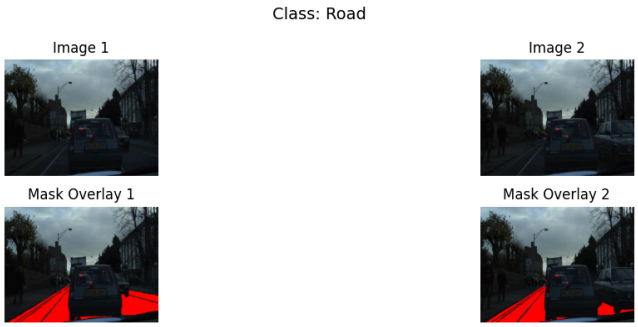
Car:



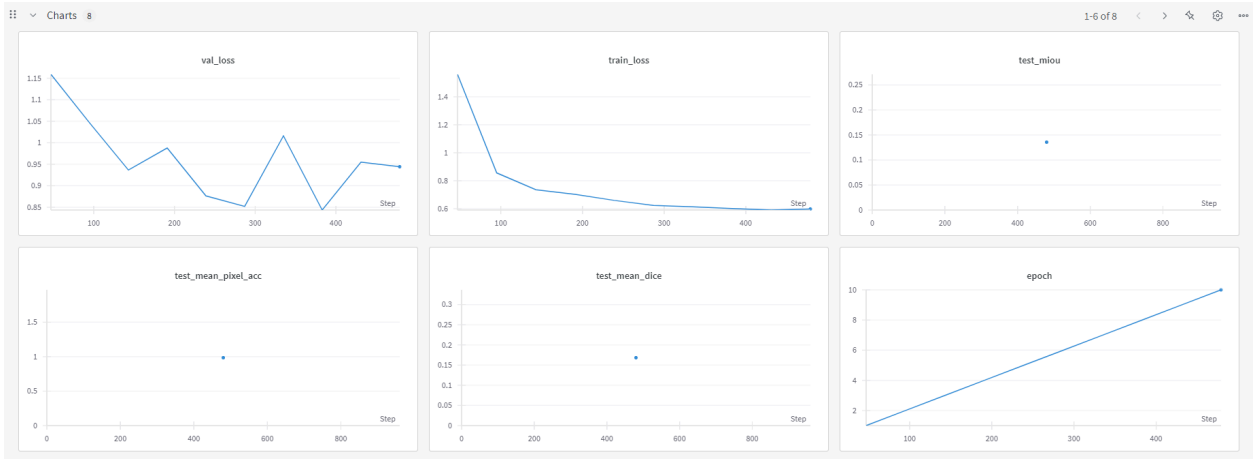
Pedestrian:



Road:



2. a.




```
Epoch: 1/10, Batch: 0/47, Loss: 3.6498
Epoch: 1/10, Batch: 10/47, Loss: 1.9176
Epoch: 1/10, Batch: 20/47, Loss: 1.3355
Epoch: 1/10, Batch: 30/47, Loss: 1.0550
Epoch: 1/10, Batch: 40/47, Loss: 1.0565
Epoch 1 - Train Loss: 1.5608, Val Loss: 1.1592
Epoch: 2/10, Batch: 0/47, Loss: 1.0502
Epoch: 2/10, Batch: 10/47, Loss: 0.8119
Epoch: 2/10, Batch: 20/47, Loss: 0.7049
Epoch: 2/10, Batch: 30/47, Loss: 0.7077
Epoch: 2/10, Batch: 40/47, Loss: 0.9901
Epoch 2 - Train Loss: 0.8560, Val Loss: 1.0460
Epoch: 3/10, Batch: 0/47, Loss: 0.7251
Epoch: 3/10, Batch: 10/47, Loss: 0.7439
Epoch: 3/10, Batch: 20/47, Loss: 0.7165
Epoch: 3/10, Batch: 30/47, Loss: 0.7515
Epoch: 3/10, Batch: 40/47, Loss: 0.7608
Epoch 3 - Train Loss: 0.7360, Val Loss: 0.9366
Epoch: 4/10, Batch: 0/47, Loss: 0.8133
Epoch: 4/10, Batch: 10/47, Loss: 0.6993
Epoch: 4/10, Batch: 20/47, Loss: 0.7403
Epoch: 4/10, Batch: 30/47, Loss: 0.6390
Epoch: 4/10, Batch: 40/47, Loss: 0.6421
Epoch 4 - Train Loss: 0.7027, Val Loss: 0.9879
Epoch: 5/10, Batch: 0/47, Loss: 0.9759
Epoch: 5/10, Batch: 10/47, Loss: 0.6857
Epoch: 5/10, Batch: 20/47, Loss: 0.5453
Epoch: 5/10, Batch: 30/47, Loss: 0.5877
Epoch: 5/10, Batch: 40/47, Loss: 0.5404
Epoch 5 - Train Loss: 0.6594, Val Loss: 0.8764
Epoch: 6/10, Batch: 0/47, Loss: 0.7045
Epoch: 6/10, Batch: 10/47, Loss: 0.6294
Epoch: 6/10, Batch: 20/47, Loss: 0.6523
Epoch: 6/10, Batch: 30/47, Loss: 0.5113
Epoch: 6/10, Batch: 40/47, Loss: 0.6570
Epoch 6 - Train Loss: 0.6234, Val Loss: 0.8519
Epoch: 7/10, Batch: 0/47, Loss: 0.7070
Epoch: 7/10, Batch: 10/47, Loss: 0.5301
Epoch: 7/10, Batch: 20/47, Loss: 0.4554
Epoch: 7/10, Batch: 30/47, Loss: 0.5686
Epoch: 7/10, Batch: 40/47, Loss: 0.5387
Epoch 7 - Train Loss: 0.6136, Val Loss: 1.0161
Epoch: 8/10, Batch: 0/47, Loss: 0.5417
Epoch: 8/10, Batch: 10/47, Loss: 0.5306
Epoch: 8/10, Batch: 20/47, Loss: 0.4750
Epoch: 8/10, Batch: 30/47, Loss: 0.5025
Epoch: 8/10, Batch: 40/47, Loss: 0.4854
Epoch 8 - Train Loss: 0.6010, Val Loss: 0.8435
Epoch: 9/10, Batch: 0/47, Loss: 0.5341
Epoch: 9/10, Batch: 10/47, Loss: 0.7498
Epoch: 9/10, Batch: 20/47, Loss: 0.5770
Epoch: 9/10, Batch: 30/47, Loss: 0.3847
Epoch: 9/10, Batch: 40/47, Loss: 0.4590
Epoch 9 - Train Loss: 0.5905, Val Loss: 0.9551
Epoch: 10/10, Batch: 0/47, Loss: 0.5567
Epoch: 10/10, Batch: 10/47, Loss: 0.4210
Epoch: 10/10, Batch: 20/47, Loss: 0.6478
Epoch: 10/10, Batch: 30/47, Loss: 0.5122
Epoch: 10/10, Batch: 40/47, Loss: 0.5144
Epoch 10 - Train Loss: 0.5981, Val Loss: 0.9442
```

b.

```
Test Set Evaluation Results:
================================================
Mean IoU (mIoU): 0.1356

Class-wise Results:
------------------------------------------------
Class                Pixel Acc       Dice        IoU
------------------------------------------------
Animal                  0.9999     0.0000     0.0000
Archway                 0.9998     0.0000     0.0000
Bicyclist               0.9934     0.0854     0.0479
Bridge                  0.9994     0.0000     0.0000
Building                0.8693     0.7306     0.6057
Car                     0.9789     0.6772     0.5418
CartLuggagePram         0.9997     0.0000     0.0000
Child                   0.9997     0.0000     0.0000
Column_Pole             0.9899     0.0606     0.0321
Fence                   0.9816     0.1496     0.0959
LaneMkgsDriv            0.9868     0.1705     0.0999
LaneMkgsNonDriv         1.0000     0.0000     0.0000
Misc_Text               0.9953     0.0000     0.0000
MotorcycleScooter       0.9998     0.0000     0.0000
OtherMoving             0.9970     0.0070     0.0036
ParkingBlock            0.9965     0.0000     0.0000
Pedestrian              0.9925     0.0342     0.0176
Road                    0.9580     0.9081     0.8358
RoadShoulder            0.9979     0.0000     0.0000
Sidewalk                0.9684     0.6980     0.5642
SignSymbol              0.9990     0.0000     0.0000
Sky                     0.9532     0.8451     0.7524
SUVPickupTruck          0.9904     0.0000     0.0000
TrafficCone             1.0000     0.0000     0.0000
TrafficLight            0.9963     0.0000     0.0000
Train                   1.0000     0.0000     0.0000
Tree                    0.9516     0.6265     0.5007
Truck_Bus               0.9885     0.0000     0.0000
Tunnel                  1.0000     0.0000     0.0000
VegetationMisc          0.9941     0.0017     0.0008
Void                    0.9525     0.1184     0.0634
Wall                    0.9854     0.2719     0.1777
```

Precision and Recall at different IoU thresholds:

Class	Threshold	Precision	Recall
Animal	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Archway	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Bicyclist	0.0	0.2387	0.0597
	0.1	0.1419	0.0377
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Bridge	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Building	0.0	0.6196	0.9521
	0.1	0.6196	0.9521
	0.2	0.6085	0.8919
	0.3	0.5996	0.8626
	0.4	0.5757	0.7984
	0.5	0.5456	0.7325
	0.6	0.4503	0.5699
	0.7	0.3599	0.4383
	0.8	0.1774	0.2029
	0.9	0.0318	0.0342
	1.0	0.0000	0.0000

Car	0.0	0.6526	0.7394
	0.1	0.6526	0.7394
	0.2	0.6469	0.7248
	0.3	0.6263	0.6799
	0.4	0.5999	0.6230
	0.5	0.5373	0.5509
	0.6	0.3643	0.3788
	0.7	0.1733	0.1839
	0.8	0.0319	0.0319
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
CartLuggagePram	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Child	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Column_Pole	0.0	0.1869	0.0379
	0.1	0.0145	0.0049
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Fence	0.0	0.2040	0.1412
	0.1	0.1674	0.1222
	0.2	0.1033	0.1066
	0.3	0.0612	0.0652
	0.4	0.0612	0.0652
	0.5	0.0216	0.0249
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000

Fence	0.0	0.2040	0.1412
	0.1	0.1674	0.1222
	0.2	0.1033	0.1066
	0.3	0.0612	0.0652
	0.4	0.0612	0.0652
	0.5	0.0216	0.0249
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
LaneMkgsDriv	0.0	0.6487	0.1031
	0.1	0.3927	0.0844
	0.2	0.1227	0.0392
	0.3	0.0283	0.0112
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
LaneMkgsNonDriv	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Misc_Text	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
MotorcycleScooter	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
OtherMoving	0.0	0.1164	0.0037
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

OtherMoving	0.0	0.1164	0.0037
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
ParkingBlock	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Pedestrian	0.0	0.1527	0.0214
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Road	0.0	0.8687	0.9531
	0.1	0.8687	0.9531
	0.2	0.8687	0.9531
	0.3	0.8687	0.9531
	0.4	0.8687	0.9531
	0.5	0.8687	0.9531
	0.6	0.8480	0.9239
	0.7	0.8214	0.8941
	0.8	0.6522	0.7028
	0.9	0.2592	0.2692
	1.0	0.0000	0.0000
RoadShoulder	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Sidewalk	0.0	0.6204	0.8399
	0.1	0.6204	0.8399
	0.2	0.6138	0.8254
	0.3	0.6059	0.8011
	0.4	0.5350	0.6730
	0.5	0.4105	0.4763
	0.6	0.3879	0.4464
	0.7	0.2653	0.2876
	0.8	0.0923	0.1000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

Sidewalk	0.0	0.6204	0.8399
	0.1	0.6204	0.8399
	0.2	0.6138	0.8254
	0.3	0.6059	0.8011
	0.4	0.5350	0.6730
	0.5	0.4105	0.4763
	0.6	0.3879	0.4464
	0.7	0.2653	0.2876
	0.8	0.0923	0.1000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
SignSymbol	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Sky	0.0	0.9295	0.8035
	0.1	0.9295	0.8035
	0.2	0.9295	0.8035
	0.3	0.8971	0.7955
	0.4	0.8971	0.7955
	0.5	0.8635	0.7811
	0.6	0.7320	0.7024
	0.7	0.6047	0.6111
	0.8	0.5125	0.5235
	0.9	0.1333	0.1325
	1.0	0.0000	0.0000
SUVPickupTruck	0.0	0.0345	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
TrafficCone	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
TrafficLight	0.0	0.0690	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

	1.0	0.0000	0.0000
TrafficLight	0.0	0.0690	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Train	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Tree	0.0	0.7688	0.5576
	0.1	0.7512	0.5568
	0.2	0.6993	0.5369
	0.3	0.6530	0.5169
	0.4	0.6067	0.4845
	0.5	0.4805	0.4181
	0.6	0.3436	0.3031
	0.7	0.2238	0.2031
	0.8	0.1007	0.0910
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Truck_Bus	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Tunnel	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

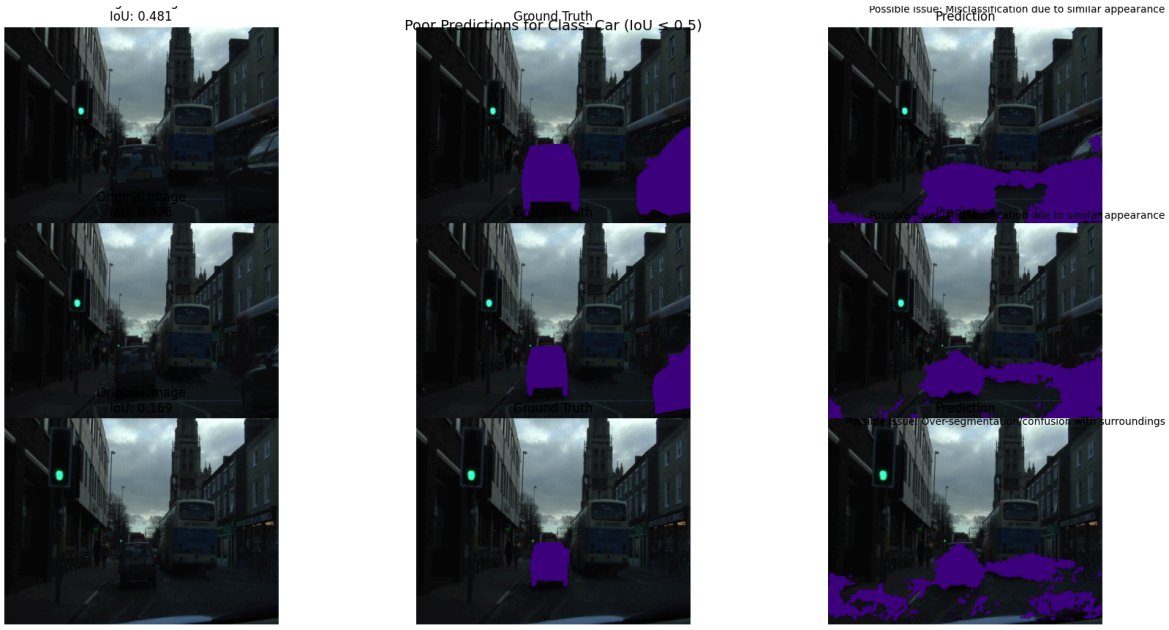
VegetationMisc	0.0	0.0953	0.0008
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Void	0.0	0.1984	0.0954
	0.1	0.0206	0.0105
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Wall	0.0	0.4121	0.2211
	0.1	0.3784	0.2140
	0.2	0.2938	0.1802
	0.3	0.1262	0.0918
	0.4	0.0808	0.0609
	0.5	0.0530	0.0442
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

wandb:

c.

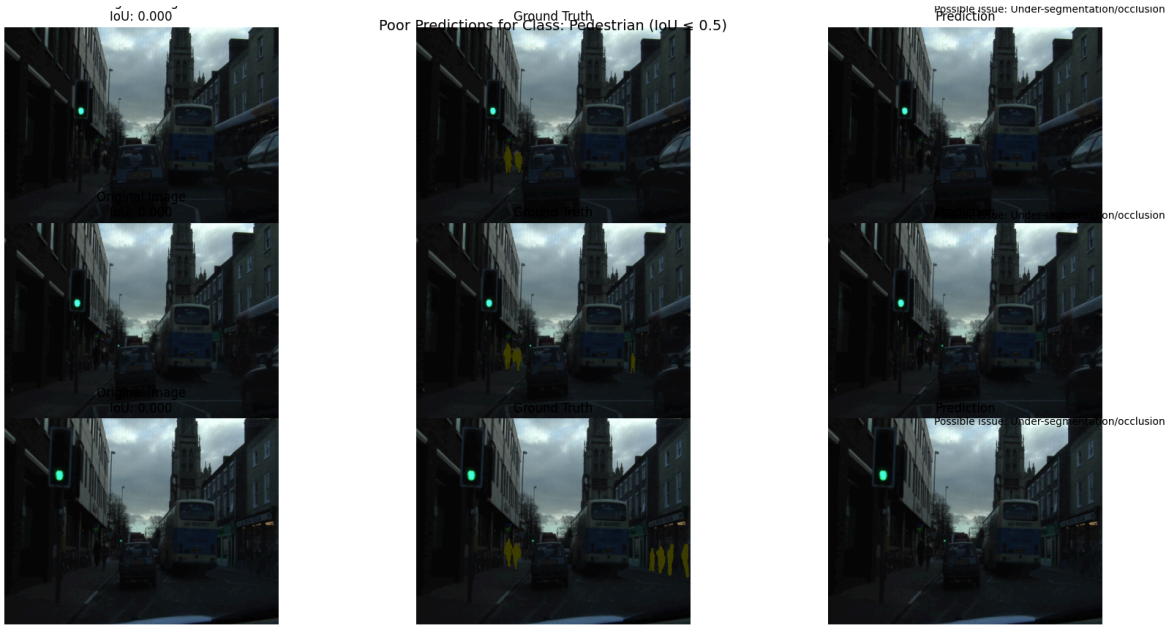
Car:

Possible issue: Misclassification due to similar appearance



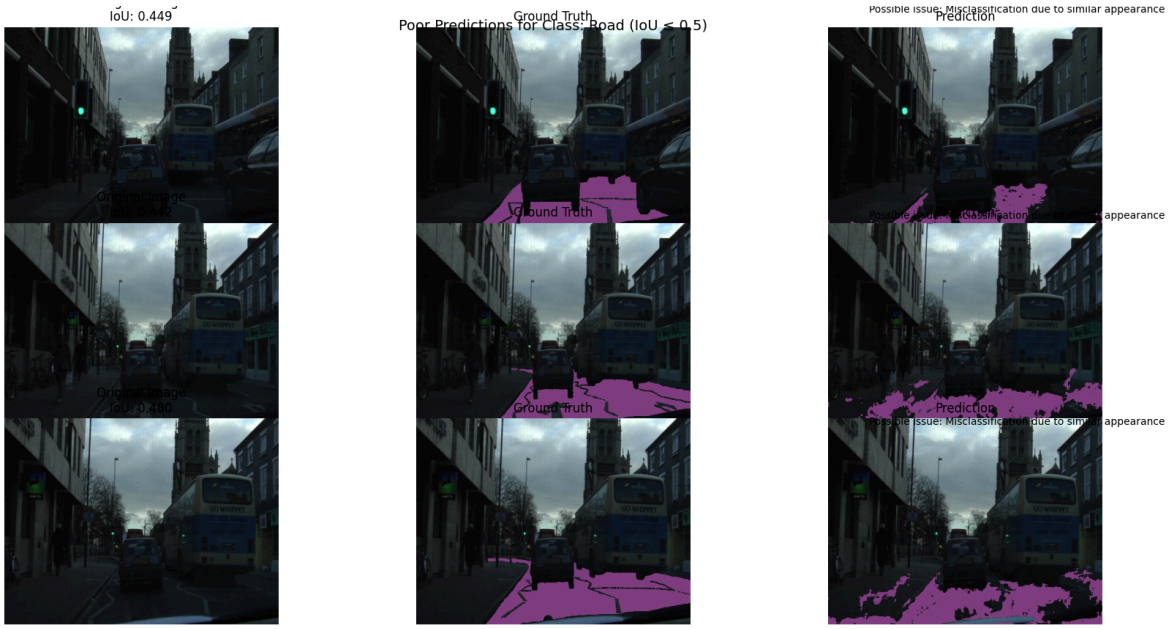
Pedestrian:

Possible issue: Under-segmentation

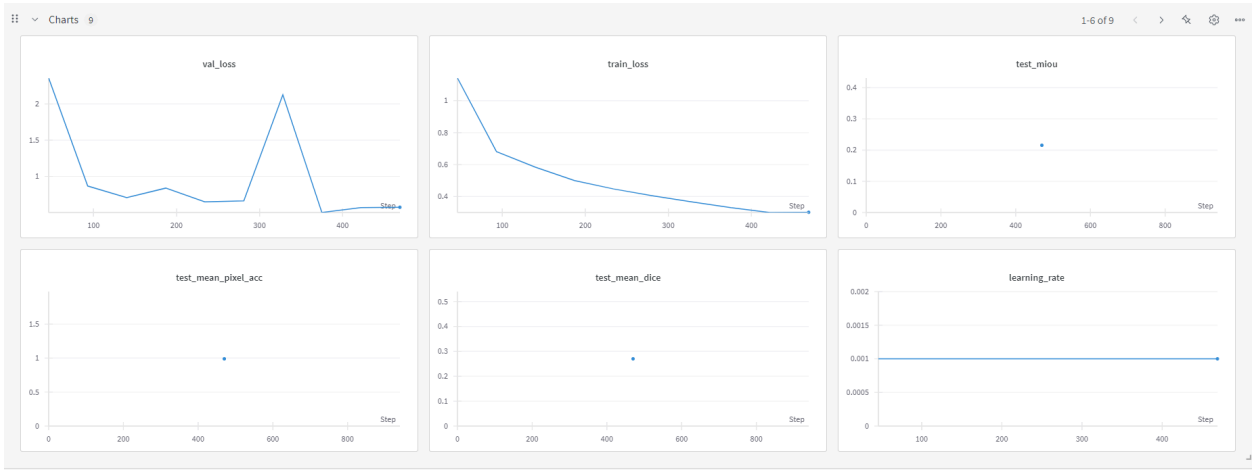


Road:

Possible issue: Misclassification due to similar appearance



3. a.




```
Epoch 1/10: 100%| 46/46 [10:46<00:00, 14.06s/it, loss=0.7357]
Epoch 1 - Train Loss: 1.1420, Val Loss: 2.3591
Epoch 2/10: 100%| 46/46 [11:13<00:00, 14.64s/it, loss=0.6097]
Epoch 2 - Train Loss: 0.6813, Val Loss: 0.8663
Epoch 3/10: 100%| 46/46 [11:06<00:00, 14.49s/it, loss=0.5411]
Epoch 3 - Train Loss: 0.5835, Val Loss: 0.7067
Epoch 4/10: 100%| 46/46 [11:09<00:00, 14.55s/it, loss=0.4711]
Epoch 4 - Train Loss: 0.5001, Val Loss: 0.8375
Epoch 5/10: 100%| 46/46 [11:19<00:00, 14.77s/it, loss=0.3616]
Epoch 5 - Train Loss: 0.4476, Val Loss: 0.6476
Epoch 6/10: 100%| 46/46 [11:16<00:00, 14.70s/it, loss=0.4172]
Epoch 6 - Train Loss: 0.4051, Val Loss: 0.6605
Epoch 7/10: 100%| 46/46 [11:23<00:00, 14.86s/it, loss=0.3260]
Epoch 7 - Train Loss: 0.3668, Val Loss: 2.1280
Epoch 8/10: 100%| 46/46 [11:12<00:00, 14.62s/it, loss=0.2989]
Epoch 8 - Train Loss: 0.3306, Val Loss: 0.5002
Epoch 9/10: 100%| 46/46 [11:08<00:00, 14.54s/it, loss=0.2611]
Epoch 9 - Train Loss: 0.3000, Val Loss: 0.5671
Epoch 10/10: 100%| 46/46 [11:08<00:00, 14.53s/it, loss=0.3151]
Epoch 10 - Train Loss: 0.3011, Val Loss: 0.5731
```

b.

```
Test Set Evaluation Results:
================================================
Mean IoU (mIoU): 0.2155

Class-wise Results:
------------------------------------------------
Class                 Pixel Acc       Dice        IoU
------------------------------------------------
Animal                   0.9999     0.0000     0.0000
Archway                  0.9998     0.0000     0.0000
Bicyclist                0.9940     0.2946     0.2090
Bridge                   0.9994     0.0000     0.0000
Building                 0.9444     0.8254     0.7210
Car                      0.9780     0.7048     0.5789
CartLuggagePram          0.9997     0.0000     0.0000
Child                    0.9997     0.0000     0.0000
Column_Pole              0.9892     0.2480     0.1477
Fence                    0.9898     0.2166     0.1490
LaneMkgsDriv             0.9881     0.5131     0.3559
LaneMkgsNonDriv          1.0000     0.0000     0.0000
Misc_Text                0.9957     0.2612     0.1675
MotorcycleScooter        0.9998     0.0000     0.0000
OtherMoving              0.9968     0.1783     0.1221
ParkingBlock             0.9948     0.1229     0.0814
Pedestrian               0.9899     0.3495     0.2252
Road                     0.9681     0.9270     0.8667
RoadShoulder             0.9959     0.0803     0.0663
Sidewalk                 0.9814     0.7692     0.6465
SignSymbol               0.9991     0.0399     0.0267
Sky                      0.9837     0.9513     0.9081
SUVPickupTruck           0.9781     0.0784     0.0463
TrafficCone              1.0000     0.0000     0.0000
TrafficLight             0.9975     0.3325     0.2485
Train                    1.0000     0.0000     0.0000
Tree                     0.9532     0.7334     0.6137
Truck_Bus                0.9891     0.0803     0.0530
Tunnel                   1.0000     0.0000     0.0000
VegetationMisc           0.9876     0.1762     0.1208
Void                     0.9735     0.3993     0.2860
Wall                     0.9883     0.3550     0.2554
```

Precision and Recall at different IoU thresholds:

Class	Threshold	Precision	Recall
Animal	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Archway	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Bicyclist	0.0	0.3422	0.3600
	0.1	0.3043	0.3280
	0.2	0.2748	0.3227
	0.3	0.2010	0.2529
	0.4	0.1124	0.1501
	0.5	0.0926	0.1278
	0.6	0.0491	0.0682
	0.7	0.0269	0.0340
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Bridge	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

Fence	0.0	0.3493	0.1666
	0.1	0.3116	0.1595
	0.2	0.2730	0.1462
	0.3	0.1742	0.1102
	0.4	0.1742	0.1102
	0.5	0.0292	0.0264
	0.6	0.0292	0.0264
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
LaneMkgsDriv	0.0	0.5513	0.5019
	0.1	0.5513	0.5019
	0.2	0.4879	0.4716
	0.3	0.3870	0.3982
	0.4	0.2648	0.2689
	0.5	0.0930	0.1002
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
LaneMkgsNonDriv	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Misc_Text	0.0	0.5488	0.1909
	0.1	0.4197	0.1743
	0.2	0.3077	0.1342
	0.3	0.1605	0.0760
	0.4	0.0965	0.0509
	0.5	0.0305	0.0194
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
MotorcycleScooter	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

OtherMoving	0.0	0.2020	0.1825
	0.1	0.1790	0.1636
	0.2	0.1400	0.1321
	0.3	0.1079	0.1100
	0.4	0.0725	0.0799
	0.5	0.0725	0.0799
	0.6	0.0272	0.0288
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
ParkingBlock	0.0	0.1499	0.1790
	0.1	0.0989	0.1707
	0.2	0.0866	0.1124
	0.3	0.0777	0.0856
	0.4	0.0371	0.0528
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Pedestrian	0.0	0.3474	0.4297
	0.1	0.3095	0.4053
	0.2	0.1952	0.2914
	0.3	0.0950	0.1326
	0.4	0.0629	0.0873
	0.5	0.0452	0.0610
	0.6	0.0259	0.0295
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Road	0.0	0.9242	0.9308
	0.1	0.9242	0.9308
	0.2	0.9242	0.9308
	0.3	0.9242	0.9308
	0.4	0.9242	0.9308
	0.5	0.9242	0.9308
	0.6	0.9242	0.9308
	0.7	0.9004	0.9017
	0.8	0.8394	0.8441
	0.9	0.3948	0.3978
	1.0	0.0000	0.0000
RoadShoulder	0.0	0.0742	0.0972
	0.1	0.0691	0.0969
	0.2	0.0691	0.0969
	0.3	0.0691	0.0969
	0.4	0.0691	0.0969
	0.5	0.0514	0.0643
	0.6	0.0514	0.0643
	0.7	0.0293	0.0328
	0.8	0.0293	0.0328
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

	1.0	0.0000	0.0000
Sidewalk	0.0	0.8338	0.7264
	0.1	0.8338	0.7264
	0.2	0.8338	0.7264
	0.3	0.7981	0.7047
	0.4	0.7804	0.6894
	0.5	0.6929	0.6346
	0.6	0.6050	0.5708
	0.7	0.3189	0.3061
	0.8	0.1942	0.1896
	0.9	0.0663	0.0652
	1.0	0.0000	0.0000
SignSymbol	0.0	0.1222	0.0294
	0.1	0.0547	0.0277
	0.2	0.0547	0.0277
	0.3	0.0547	0.0277
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Sky	0.0	0.9632	0.9404
	0.1	0.9632	0.9404
	0.2	0.9632	0.9404
	0.3	0.9632	0.9404
	0.4	0.9632	0.9404
	0.5	0.9632	0.9404
	0.6	0.9632	0.9404
	0.7	0.9632	0.9404
	0.8	0.9632	0.9404
	0.9	0.6340	0.6314
	1.0	0.0000	0.0000
SUVPickupTruck	0.0	0.0766	0.1017
	0.1	0.0559	0.0794
	0.2	0.0411	0.0434
	0.3	0.0198	0.0159
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
TrafficCone	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

TrafficLight	0.0	0.3638	0.3349
	0.1	0.3638	0.3349
	0.2	0.3568	0.3245
	0.3	0.2819	0.2942
	0.4	0.2430	0.2612
	0.5	0.1705	0.1957
	0.6	0.1059	0.1117
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Train	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Tree	0.0	0.6530	0.8640
	0.1	0.6530	0.8640
	0.2	0.6530	0.8640
	0.3	0.6454	0.8328
	0.4	0.5896	0.7352
	0.5	0.5744	0.7067
	0.6	0.4924	0.5817
	0.7	0.3702	0.4251
	0.8	0.0931	0.1016
	0.9	0.0315	0.0339
	1.0	0.0000	0.0000
Truck_Bus	0.0	0.1897	0.0604
	0.1	0.1161	0.0465
	0.2	0.0851	0.0420
	0.3	0.0657	0.0329
	0.4	0.0319	0.0209
	0.5	0.0319	0.0209
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Tunnel	0.0	0.0000	0.0000
	0.1	0.0000	0.0000
	0.2	0.0000	0.0000
	0.3	0.0000	0.0000
	0.4	0.0000	0.0000
	0.5	0.0000	0.0000
	0.6	0.0000	0.0000
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

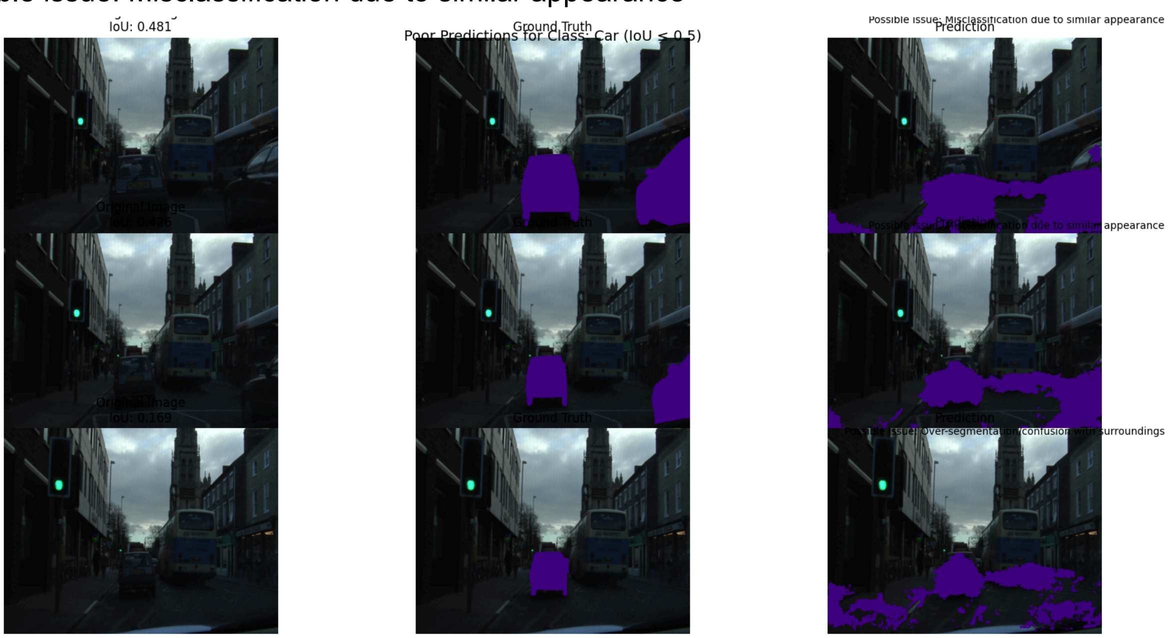
VegetationMisc	0.0	0.1349	0.3528
	0.1	0.1261	0.2668
	0.2	0.1110	0.1856
	0.3	0.0760	0.1248
	0.4	0.0643	0.0914
	0.5	0.0643	0.0914
	0.6	0.0255	0.0297
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Void	0.0	0.6358	0.3205
	0.1	0.4904	0.2976
	0.2	0.4286	0.2755
	0.3	0.4048	0.2648
	0.4	0.2909	0.2108
	0.5	0.2067	0.1571
	0.6	0.0615	0.0517
	0.7	0.0324	0.0266
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000
Wall	0.0	0.5533	0.3151
	0.1	0.4445	0.2774
	0.2	0.3455	0.2455
	0.3	0.3056	0.2275
	0.4	0.2012	0.1722
	0.5	0.2012	0.1722
	0.6	0.1478	0.1266
	0.7	0.0000	0.0000
	0.8	0.0000	0.0000
	0.9	0.0000	0.0000
	1.0	0.0000	0.0000

wandb:

```
wandb:
wandb:
wandb: Run history:
wandb:                batch
wandb:           batch_loss
wandb:                epoch
wandb:        learning_rate
wandb:       test_mean_dice
wandb: test_mean_pixel_acc
wandb:            test_miou
wandb:           train_loss
wandb:             val_loss
wandb:
wandb: Run summary:
wandb:                batch 459
wandb:           batch_loss 0.31513
wandb:                epoch 10
wandb:        learning_rate 0.001
wandb:       test_mean_dice 0.26991
wandb: test_mean_pixel_acc 0.9892
wandb:            test_miou 0.21549
wandb:           train_loss 0.30113
wandb:             val_loss 0.57311
wandb:
wandb:  View run trim-cherry-3 at: https://wandb.ai/vikranth2764-na/segmentation-deeplabv3/runs/e10v66ic
wandb:  View project at: https://wandb.ai/vikranth2764-na/segmentation-deeplabv3
wandb: Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
wandb: Find logs at: .\wandb\run-20250223_172020-e10v66ic\logs
```

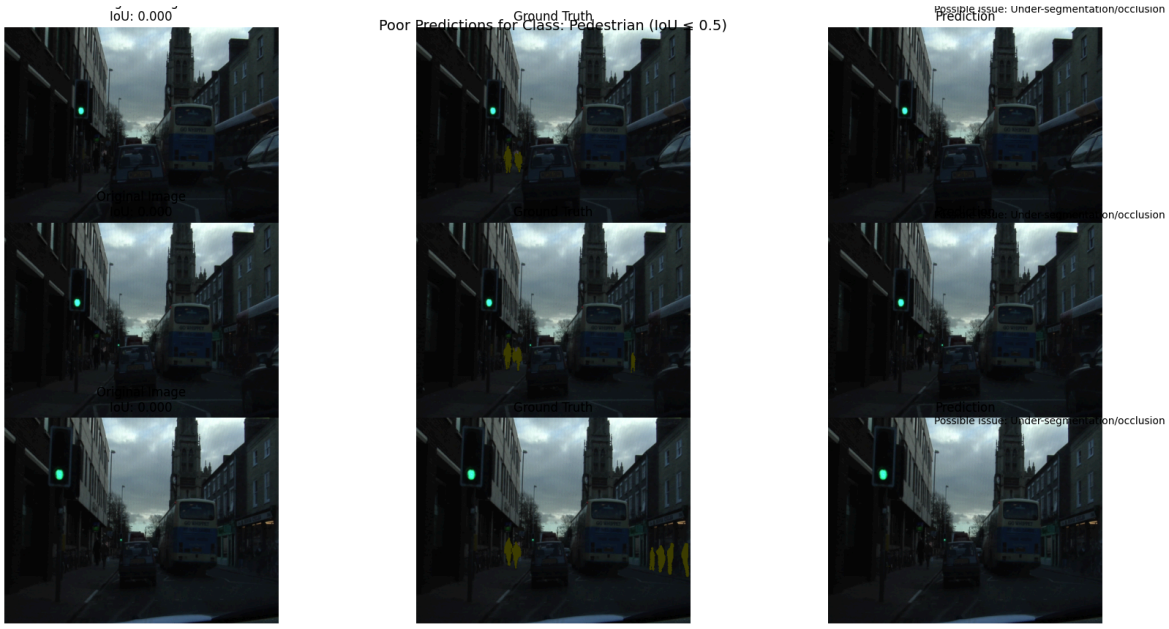
c. Car:

Possible issue: Misclassification due to similar appearance



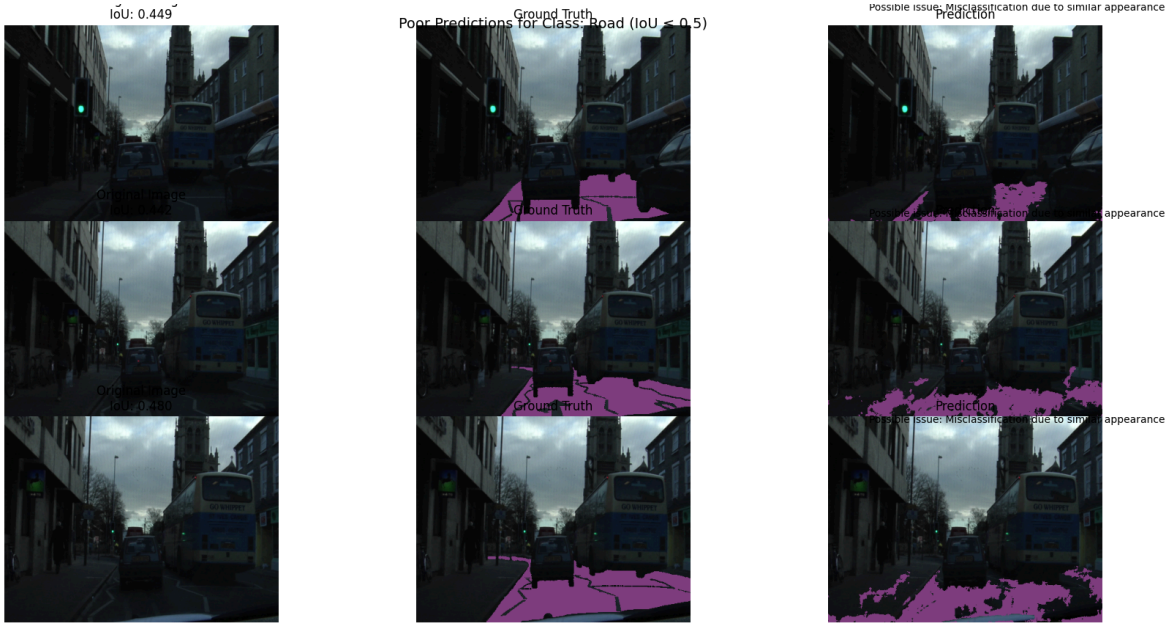
Pedestrian:

Possible issue: Under-segmentation



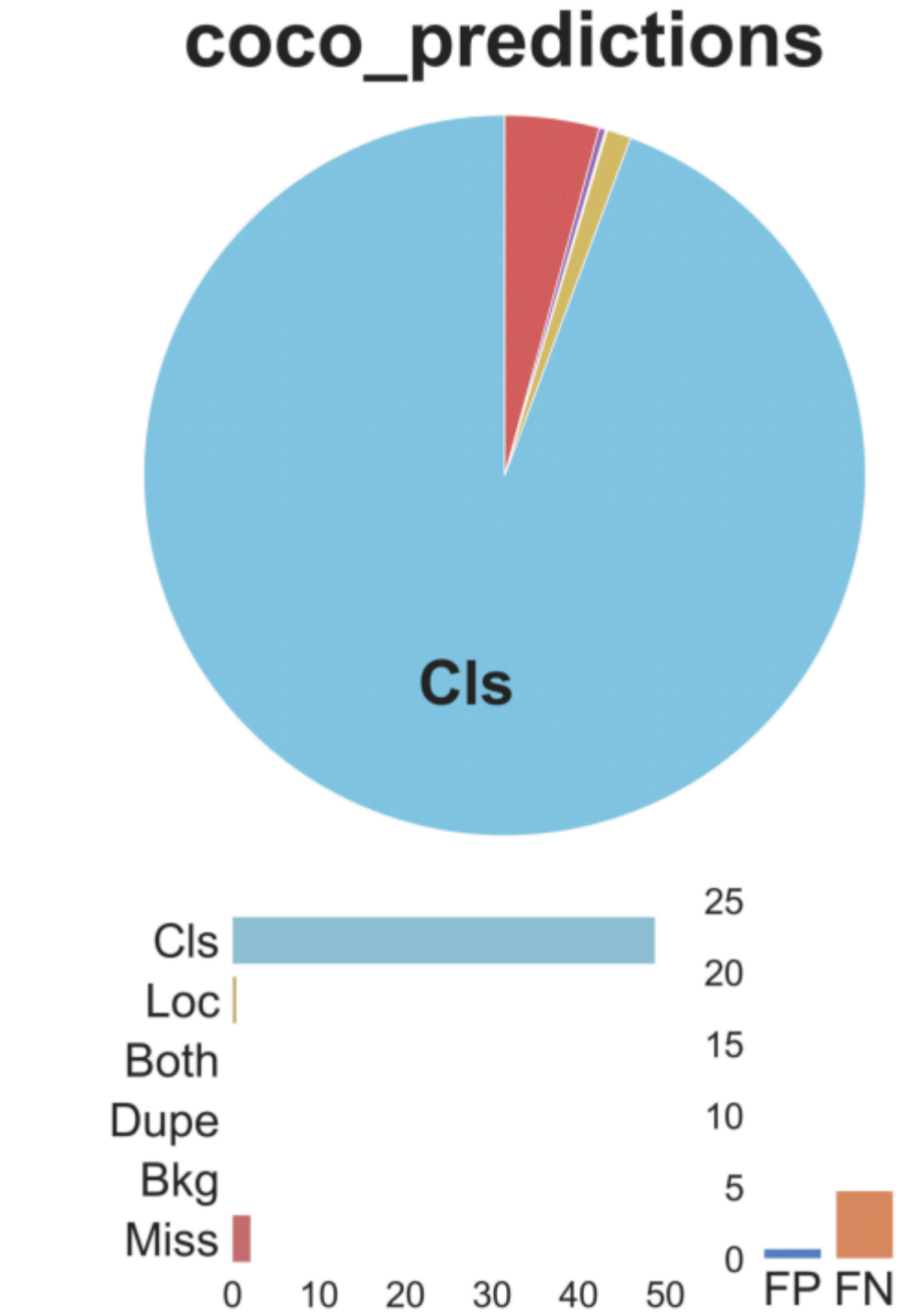
Road:

Possible issue: Misclassification due to similar appearance



4. 1. a. No deliverables

```
# Download the Dataset by using Ultralytics API

from ultralytics.data.utils import download
download(url='https://github.com/ultralytics/yolov5/releases/download/v1.0/coco2017val.zip')

with zipfile.ZipFile('coco2017val.zip', 'r') as zip_ref:
    zip_ref.extractall()
```


coco_predictions
Cls
Cls
Loc
Both
Dupe
Bkg
Miss
0
10
20
30
40
50
25
20
15
10
5
0
FP
FN

b. c.

```
-- coco_predictions --

bbox AP @ 50: 8.61

                      Main Errors
============================================================
  Type      Cls      Loc     Both     Dupe      Bkg     Miss
------------------------------------------------------------
   dAP    48.84     0.56     0.03     0.02     0.15     2.19
============================================================

       Special Error
============================
  Type   FalsePos   FalseNeg
----------------------------
   dAP       0.73       4.78
============================
```

mAP @ 50: 8.61%

- This is the **Mean Average Precision (mAP)** at an **IoU threshold of 0.5**, which measures how well the model's predictions align with the ground truth.

TIDE Statistics:

Cls (Classification Error): 48.84%

- The **dominant error type**.
- Model is **misclassifying objects** (wrong labels).

Loc (Localization Error): 0.56%

- Very low, meaning **bounding boxes are mostly accurate**.

Both (Cls + Loc Errors Combined): 0.03%

- Model rarely makes both classification and localization errors together.

Dupe (Duplicate Detections): 0.02%

- Model almost never predicts the **same object multiple times**.

Bkg (Background Error): 0.15%

- Minimal **false positives from the background**.

Miss (Missed Detections): 2.19%

- The model misses a small percentage of actual objects.

alse Positives (FP): 0.73%

- Incorrect detections where nothing should be detected.

False Negatives (FN): 4.78%

- Ground truth objects were not detected.

d. ECE: 0.2308927360370347.

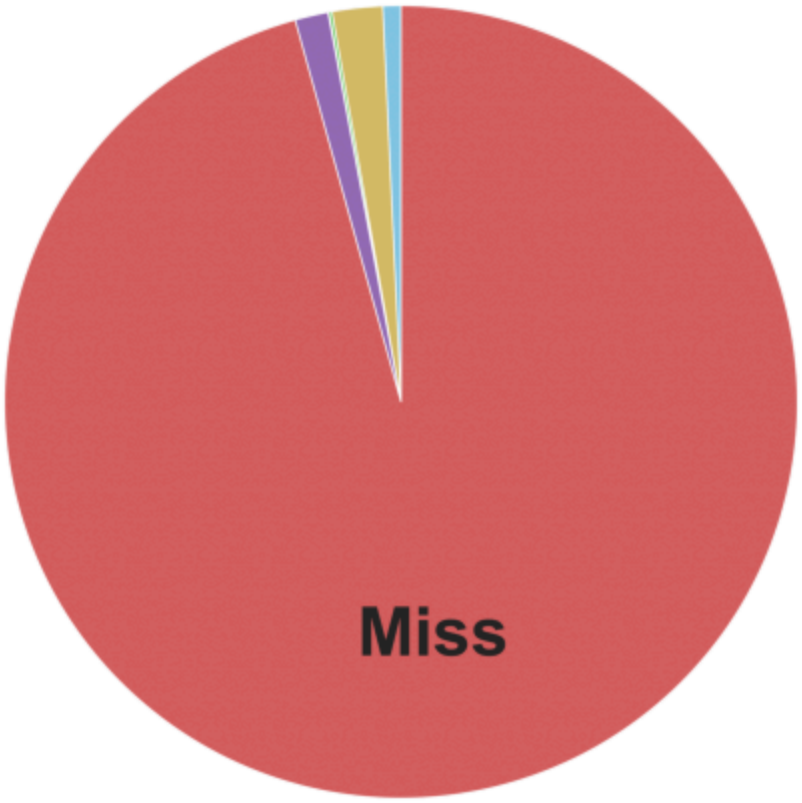
This indicates significant **overconfidence**, meaning the model assigns high confidence to incorrect predictions. This aligns with the **48.84% classification error** from TIDE, showing that misclassification is the biggest issue. To improve, apply **temperature scaling** or **lower the confidence threshold** to better align confidence scores with accuracy.

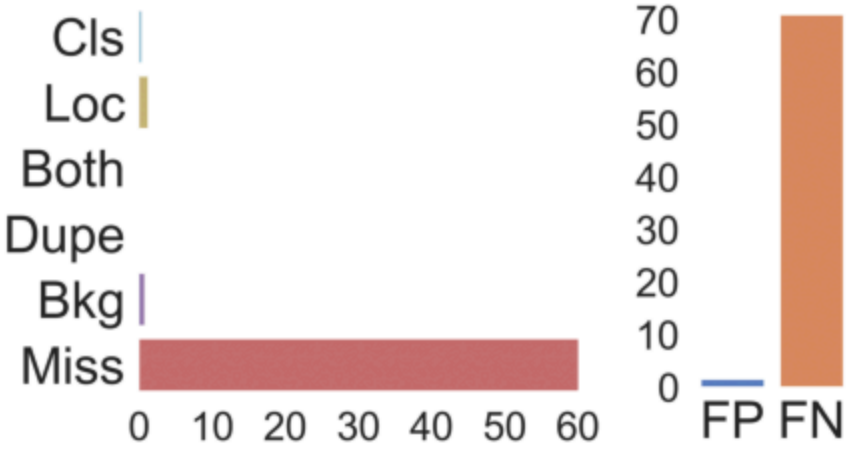
```
loading annotations into memory...
Done (t=0.12s)
creating index...
index created!

Analyzing small objects...
Loading and preparing results...
DONE (t=0.04s)
creating index...
index created!
Processing images: 100%|██████████| 5000/5000 [00:00<00:00, 42112.75it/s]
ECE for small objects: 0.12887502587339783

Analyzing medium objects...
Loading and preparing results...
DONE (t=0.14s)
creating index...
index created!
Processing images: 100%|██████████| 5000/5000 [00:00<00:00, 48291.88it/s]
ECE for medium objects: 0.15358833936401287

Analyzing large objects...
Loading and preparing results...
DONE (t=0.05s)
creating index...
index created!
Processing images: 100%|██████████| 5000/5000 [00:00<00:00, 26996.56it/s]
ECE for large objects: 0.16464595028723078
```

e.

```
Analyzing small objects...

TIDE Statistics for small objects:
-- coco_predictions_small --

bbox AP @ 50: 9.38

                        Main Errors
============================================================
  Type      Cls      Loc     Both     Dupe      Bkg     Miss
------------------------------------------------------------
   dAP     0.46     1.29     0.08     0.02     0.85    60.12
============================================================

        Special Error
============================
  Type   FalsePos   FalseNeg
----------------------------
   dAP       1.54      70.94
============================
```

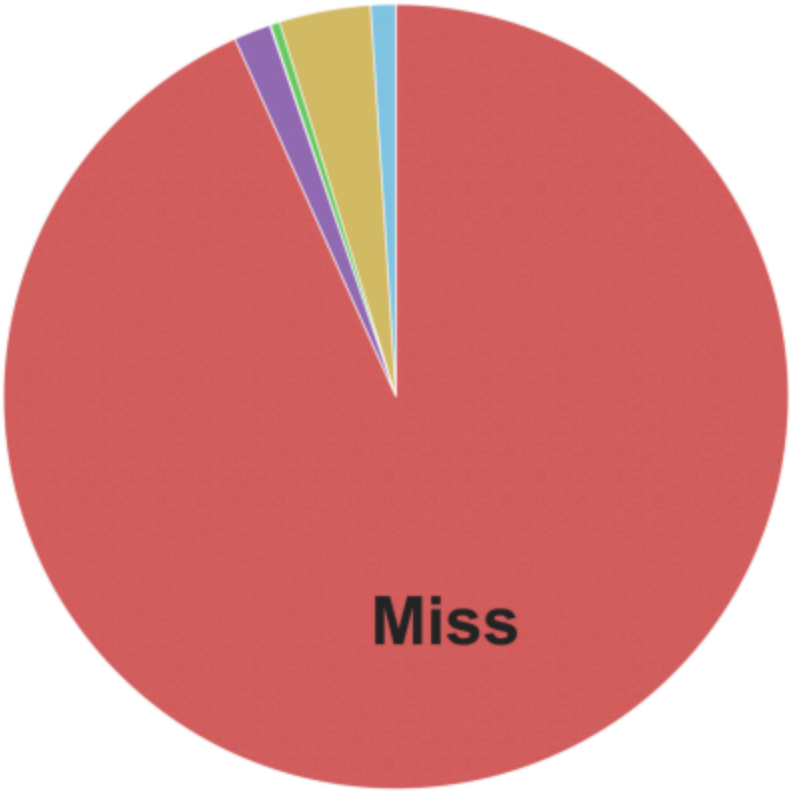
coco_predictions_small

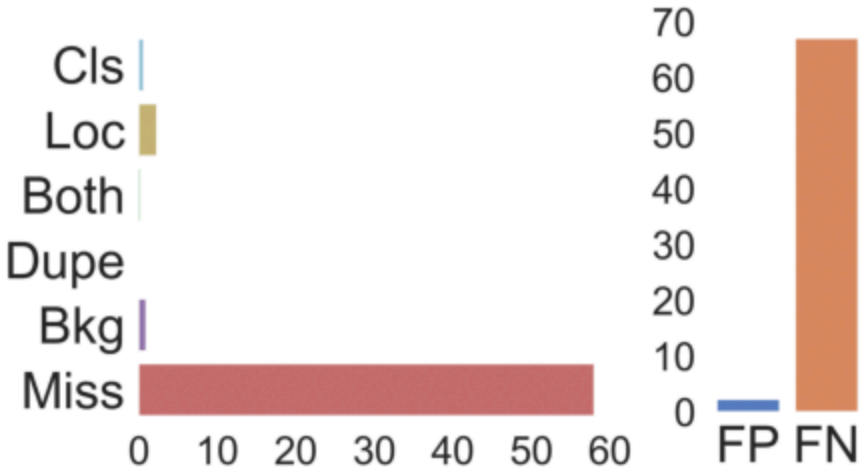




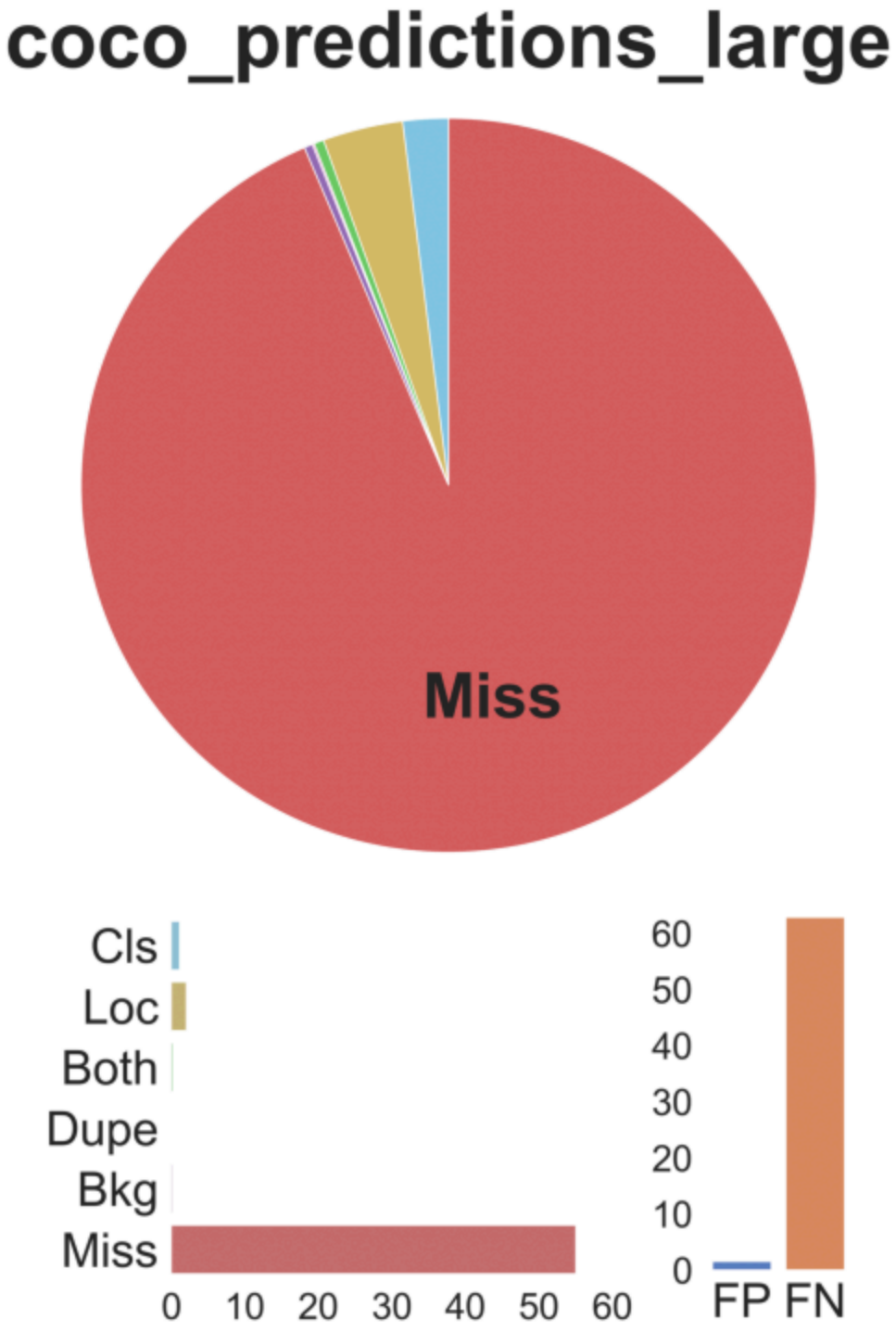
```
Analyzing medium objects...

TIDE Statistics for medium objects:
-- coco_predictions_medium --

bbox AP @ 50: 22.19

                        Main Errors
============================================================
  Type      Cls      Loc     Both     Dupe      Bkg     Miss
------------------------------------------------------------
   dAP     0.64     2.34     0.24     0.02     0.94    58.00
============================================================

        Special Error
============================
  Type   FalsePos   FalseNeg
----------------------------
   dAP       2.27      67.04
============================
```

coco_predictions_medium





```
Analyzing large objects...

TIDE Statistics for large objects:
-- coco_predictions_large --

bbox AP @ 50: 31.29

                        Main Errors
============================================================
  Type      Cls      Loc     Both     Dupe      Bkg     Miss
------------------------------------------------------------
   dAP     1.18     2.07     0.26     0.05     0.21    55.07
============================================================

        Special Error
============================
  Type   FalsePos   FalseNeg
----------------------------
   dAP       1.56      62.91
============================
```



ECE (Calibration Error)

- **Small objects: 0.1289**, **Medium: 0.1536**, **Large: 0.1646 → ECE increases with object size**.

- **Lower ECE for small objects** suggests the model is better calibrated for them.
- **Higher ECE for large objects** indicates overconfidence in incorrect predictions.

AP @ 50 for Small: 9.38, Medium: 22.19, Large: 31.29 → Performance improves with size.

- **Missed Detections are dominant (Small: 60.12%, Medium: 58.00%, Large: 55.07%)** → The model struggles to detect objects, especially small ones.
- **Localization error is highest for medium & large objects (2.34%, 2.07%)**, meaning **bounding box accuracy decreases as object size increases**.
- **False Negatives are very high across all sizes (Small: 70.94%, Medium: 67.04%, Large: 62.91%)**, indicating **many objects remain undetected**.

The model performs best on large objects but still struggles with missed detections.

False Negatives are the biggest issue across all sizes, suggesting objects are not being detected at all.

Calibration is better for small objects but worsens for medium and large ones.