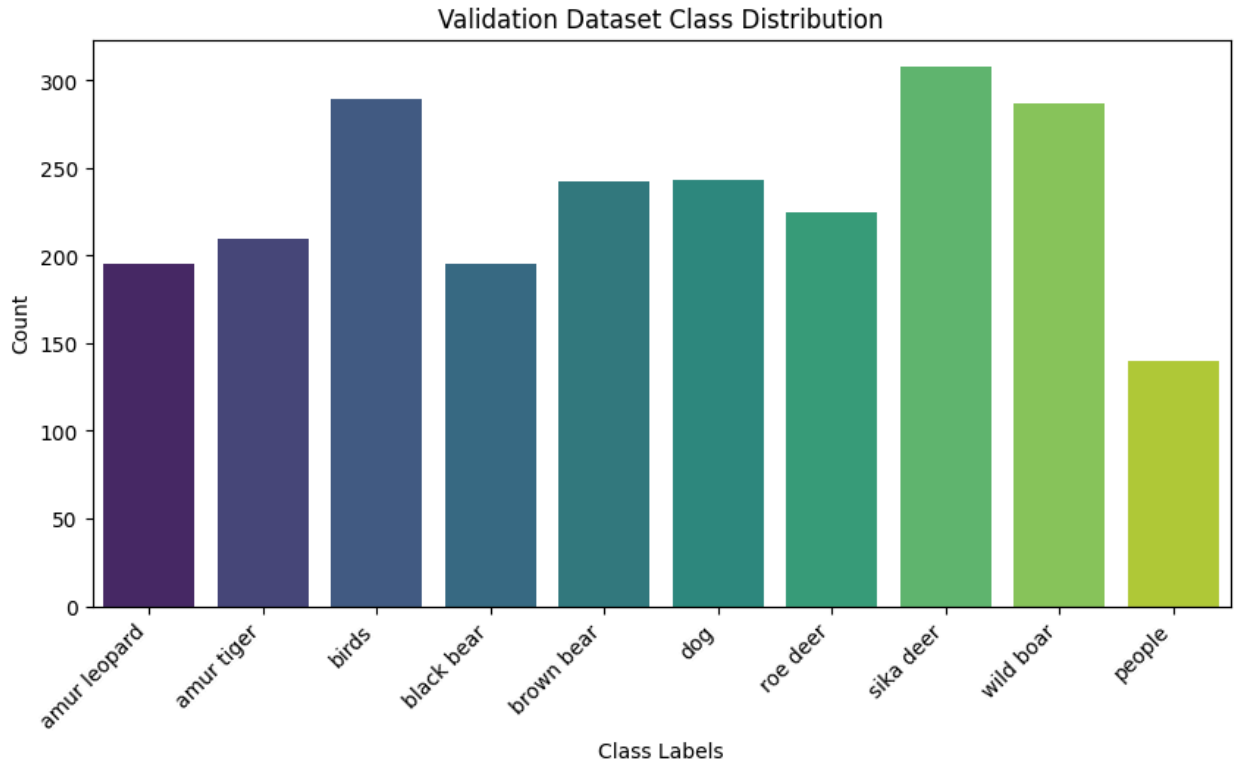
CV A1 Early Bird 1

Q2) Image Classification

1.a. Custom Dataset Class

```python
class RussianWildlifeDataset(Dataset):
    def __init__(self, image_paths, labels, transforms=None, augment_transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transforms = transforms
        self.augment_transform = augment_transform

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, index):
        image_path = self.image_paths[index]
        image = Image.open(image_path).convert('RGB')
        label = self.labels[index]
        if self.transforms:
            image = self.transforms(image)

        if self.augment_transform:
            image = self.augment_transform(image)

        return image, label
```

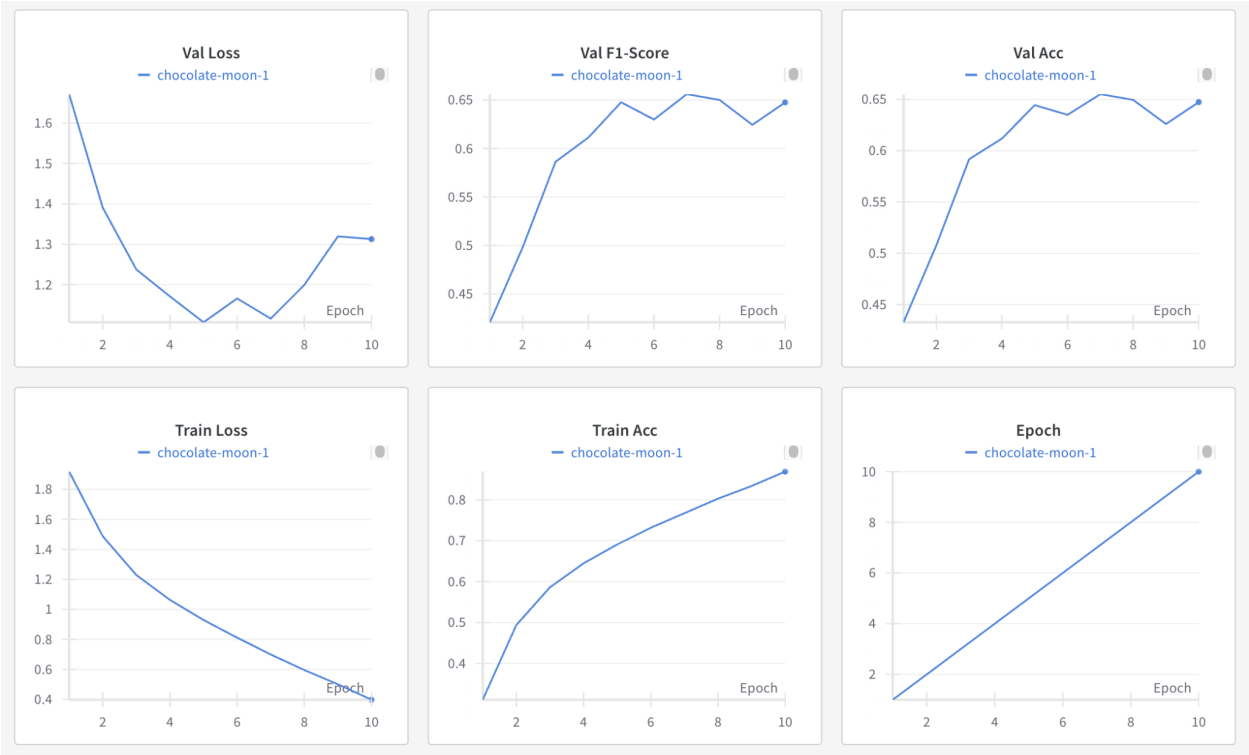
1.c. Data Visualization



2.a. CNN Model Class:

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        # First Conv Layer:
        # 32 filters, 3x3 kernel, stride=1, padding=1
        # Max Pool (4x4, stride=4)
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.pool1 = nn.MaxPool2d(kernel_size=4, stride=4)

        # Second Conv Layer:
        # 64 filters, 3x3 kernel, stride=1, padding=1
        # Max Pool (2x2, stride=2)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Third Conv Layer:
        # 128 filters, 3x3 kernel, stride=1, padding=1
        # Max Pool (2x2, stride=2)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1)
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.fc = nn.Linear(128 * 14 * 14, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.relu(self.conv1(x))
        out = self.pool1(out)

        out = self.relu(self.conv2(out))
        out = self.pool2(out)

        out = self.relu(self.conv3(out))
        out = self.pool3(out)

        out = torch.flatten(out, start_dim=1)
        out = self.fc(out)
        return out
```

2.b. Convnet plots



	amur leopard	amur tiger	birds	black bear	brown bear	dog	roe deer	sika deer	wild boar	people
1	145	16	8	2	4	5	2	10	3	0
2	21	131	13	2	6	6	12	12	0	7
3	8	1	210	6	7	15	1	19	6	16
4	1	0	5	122	35	3	1	8	18	2
5	3	6	3	28	160	8	6	12	12	4
6	5	7	10	11	8	134	13	38	10	7
7	7	8	1	2	4	16	136	38	11	2
8	5	6	22	5	9	23	41	165	17	15
9	2	5	3	11	14	11	8	15	214	4
10	6	4	11	0	6	5	0	12	2	94

Yes data is overfitting as the validation loss is increasing as the train loss is decreasing. The misclassified images are very similar to the other class images, and there were also instances of image containing dogs classified as “people”. Data Augmentation would be a good method to approach the problem.

2.d. On Validation Dataset:
Accuracy: 0.64739
F1-score: 0.64754

2.e. Misclassified Images

Misclassified Images for Class: black bear

Pred: wild boar

Pred: brown bear

Pred: sika deer

Misclassified Images for Class: brown bear

Pred: wild boar

Pred: people

Pred: dog

Misclassified Images for Class: dog

Pred: brown bear

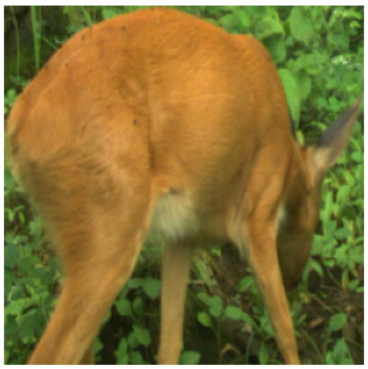
Pred: birds

Pred: wild boar

Misclassified Images for Class: roe deer

Pred: dog

Pred: wild boar



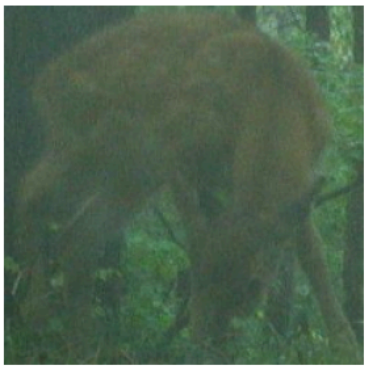
Pred: amur leopard

Misclassified Images for Class: sika deer

Pred: birds

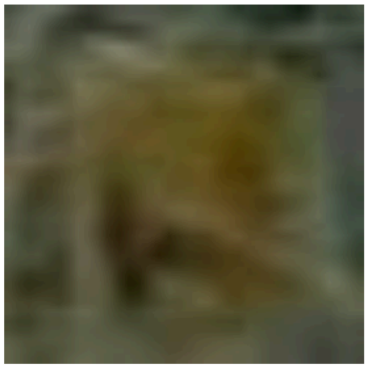
Pred: roe deer

Pred: wild boar



Misclassified Images for Class: wild boar

Pred: dog



Pred: brown bear

Pred: roe deer

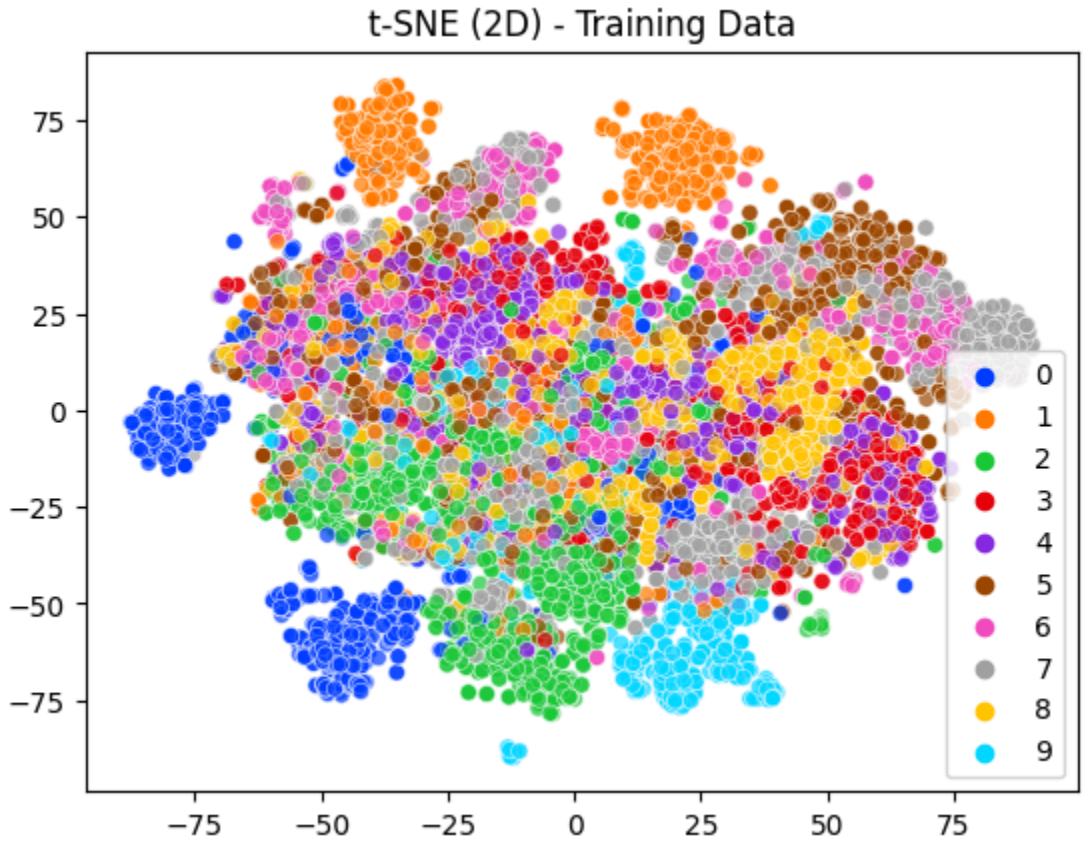
3.a. Resnet fine-tuned model plots

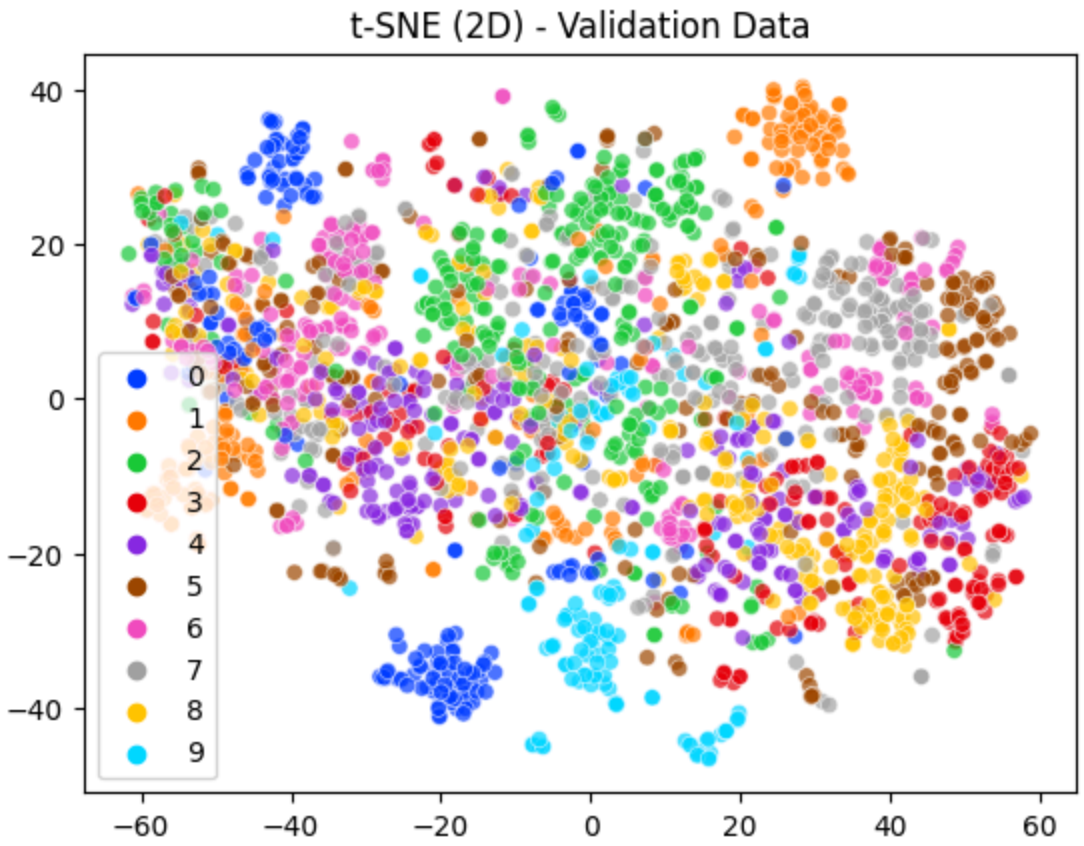
	amur leopard	amur tiger	birds	black bear	brown bear	dog	roe deer	sika deer	wild boar	people
1	179	3	2	0	0	2	6	3	0	0
2	0	194	1	1	1	4	2	2	3	2
3	0	1	259	2	2	12	2	1	6	4
4	0	0	3	144	37	0	2	2	7	0
5	1	0	2	51	158	5	3	5	16	1
6	0	2	10	6	9	165	13	16	18	4
7	0	1	7	2	2	9	173	27	3	1
8	2	2	14	2	6	6	29	220	25	2
9	0	0	10	5	10	8	5	4	244	1
10	0	1	1	0	2	3	2	4	1	126

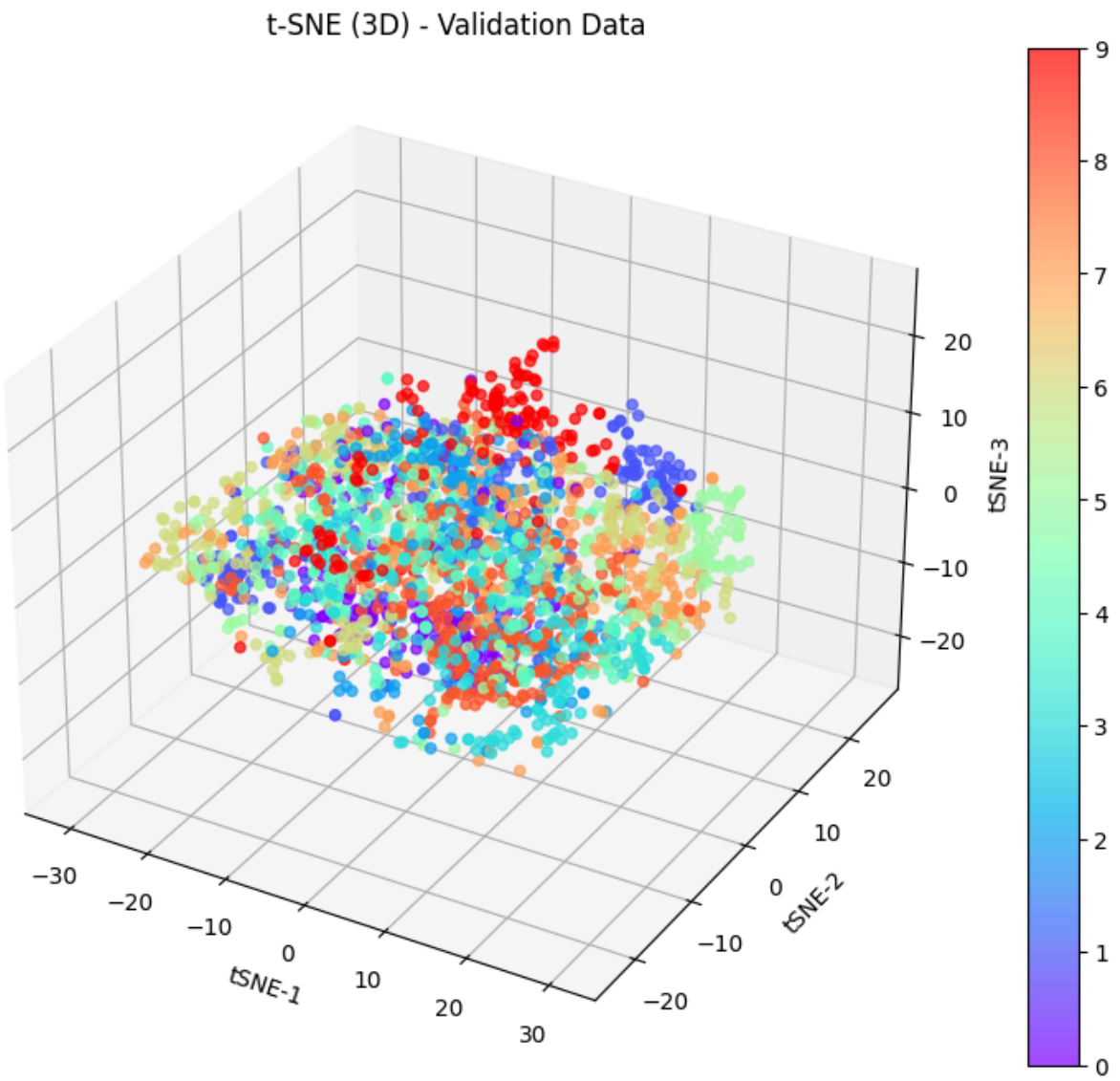
No the model is not overfitting as the train and validation losses are very close to each other.

3.c. On Validation Dataset:
Accuracy: 0.79777
F1-score: 0.79737

3.d. t-SNE plots

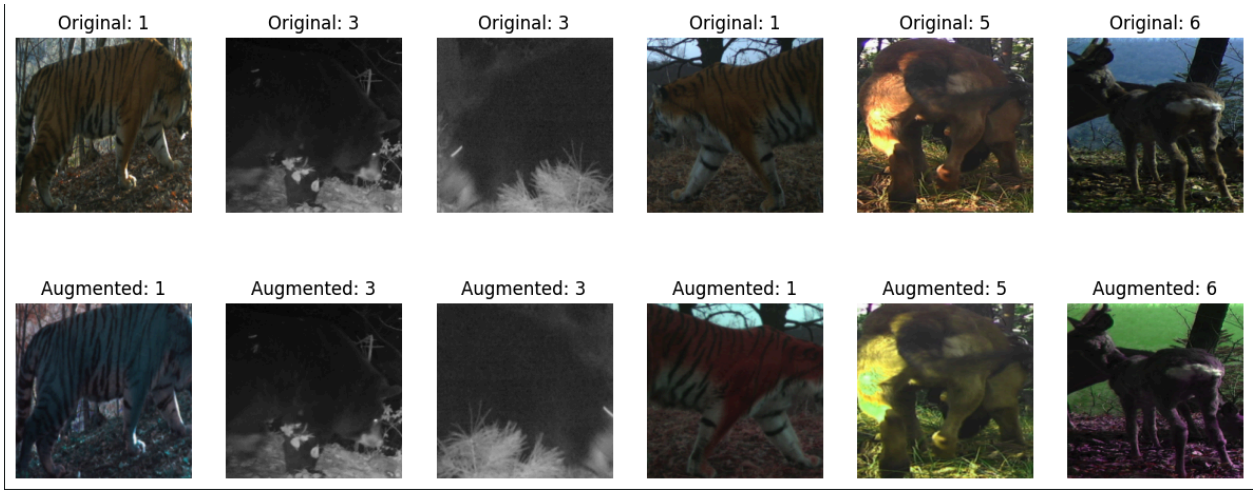



t-SNE (2D) - Validation Data
40
20
0
−20
−40
−60
−40
−20
0
20
40
60
0
1
2
3
4
5
6
7
8
9

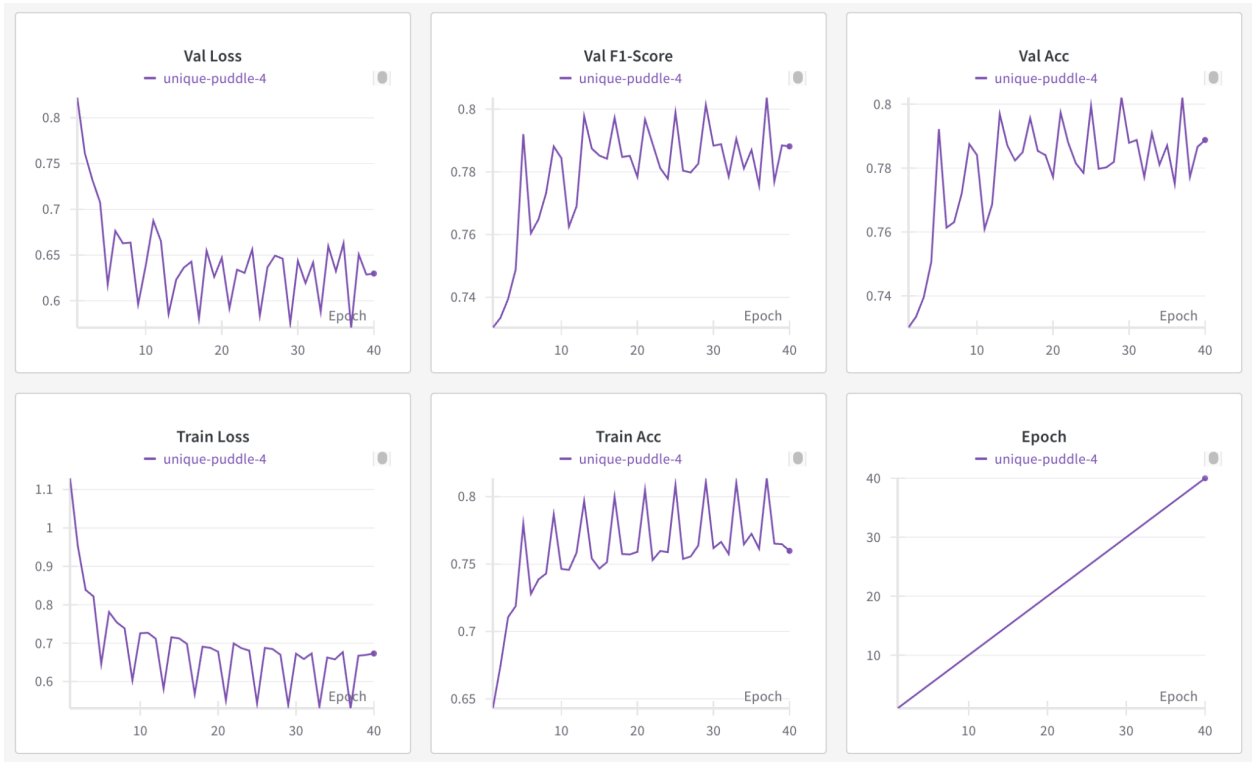


4.a. Augmentation

```python
aug_transform = transforms.Compose([
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomResizedCrop(size=224, scale=(0.8, 1.0)),
    transforms.ColorJitter(brightness=0, contrast=0, saturation=0, hue=0.5)
])
```



4.b. Data augmented Resnet fine-tuned model plots



	amur leopard	amur tiger	birds	black bear	brown bear	dog	roe deer	sika deer	wild boar	people
1	178	2	1	0	0	4	10	0	0	0
2	2	189	1	1	0	9	2	0	4	2
3	1	0	256	0	5	17	4	1	3	2
4	0	0	4	131	32	2	1	4	21	0
5	1	0	2	50	151	6	5	4	23	0
6	0	4	13	3	13	152	31	13	13	1
7	0	0	4	1	2	11	198	5	4	0
8	0	2	13	3	5	14	82	164	23	2
9	0	1	10	5	7	9	11	3	240	1
10	0	0	5	0	1	8	3	2	3	118

The problem of overfitting has been removed. The training and validation losses are very similar.

4.d. On Validation Dataset:
Accuracy: 0.78877
F1-score: 0.7881

5. The CNN model is poor, validation accuracy of around 65% and training accuracy of around 87%

The original ResNet18 fine-tuned model is doing well, val accuracy of around 80%, training accuracy of around 82%

The final data augmented ResNet18 fine-tuned model: val accuracy of around 80%, training accuracy of around 82%.