Assignment 3: Dog vs Cat vs Bird Classifier

Deep Learning CS669
Dr. Tahir Syed
Hiba Azeem (29404) & **Fizza Fatima (29406)**

December 31, 2024

1 Introduction

In this project, the task is to classify images of dogs, cats, and birds using deep learning techniques. The model was built using PyTorch and trained on a custom dataset of labeled images. Various optimization techniques were employed, and both a baseline CNN model and pre-trained models (ResNet and VGGNet) were tested.

2 Dataset Preprocessing and Exploration

The dataset was preprocessed using the following transformations to standardize the image size and normalize pixel values:

```
transform = transforms.Compose([
    transforms.Resize((32, 32)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])
```

For the dataset loading, two custom Dataset classes were created: one for training data and one for testing data:

```
class BirdDogCatDataset(Dataset):
    def __init__(self, dataframe, data_dir, transform=None):
        self.dataframe = dataframe
        self.data_dir = data_dir
        self.transform = transform
        self.classes = {label: idx for idx, label in
        enumerate(sorted(dataframe['label'].unique()))}

    def __len__(self):
        return len(self.dataframe)
```

```
    def __getitem__(self, idx):
        img_name = self.dataframe.iloc[idx, 0]
        label_str = self.dataframe.iloc[idx, 1]
        label = self.classes.get(label_str, -1)
        if label == -1:
            raise ValueError(f"Label {label_str} not found in the classes mapping.")

        img_path = os.path.join(self.data_dir, img_name)
        image = Image.open(img_path).convert("RGB")

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

        return image, label

class TestDataset(Dataset):
    def __init__(self, data_dir, transform=None):
        self.data_dir = data_dir
        self.image_files = os.listdir(data_dir)
        self.transform = transform

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

    def __getitem__(self, idx):
        image_name = self.image_files[idx]
        image_path = os.path.join(self.data_dir, image_name)
        image = Image.open(image_path).convert("RGB")

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

        return image, image_name
```

3 Baseline Model Development

A Convolutional Neural Network (CNN) was developed from scratch using PyTorch. The architecture consists of two convolutional layers followed by fully connected layers to classify the images into three classes: Dog, Cat, and Bird.

```
class ConvolutionalNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 3, 1)
        self.conv2 = nn.Conv2d(6, 16, 3, 1)
```

```
        self.Fc1 = nn.Linear(16 * 6 * 6, 120)
        self.Fc2 = nn.Linear(120, 84)
        self.Fc3 = nn.Linear(84, 3)

    def forward(self, X):
        X = F.relu(self.conv1(X))
        X = F.max_pool2d(X, 2, 2)

        X = F.relu(self.conv2(X))
        X = F.max_pool2d(X, 2, 2)

        X = X.view(-1, 16 * 6 * 6)

        X = F.relu(self.Fc1(X))
        X = F.relu(self.Fc2(X))
        X = self.Fc3(X)

        return F.log_softmax(X, dim=1)
```

Training Configuration:

- Optimizer: Adam
- Learning Rate: 0.001
- Loss Function: CrossEntropyLoss
- Epochs: 15

4 Optimization Techniques

Optimization techniques such as using different learning rates, weight decay, and momentum values were experimented with to enhance the model's performance. The results from the experiments are summarized in Table 1.

Model	**Optimizer**	**Scheduler**	**Learning Rate**	**Weight Decay**	**Momentum**
Baseline Model	Adam	None	0.001	None	0.00
ResNet	Adam	None	0.001	1e-4	0.00
VGGNet	Adam	None	0.001	1e-4	0.00
ResNet	Adam	StepLR	0.001	1e-4	0.00
ResNet	SGD	StepLR	0.01	5e-4	0.90
ResNet	SGD	StepLR	0.005	1e-3	0.95
ResNet	SGD	ReduceLROnPlateau	0.005	1e-3	0.95
ResNet	SGD	ReduceLROnPlateau	0.005	1e-3	0.95

Table 1: Optimization Techniques for Different Models

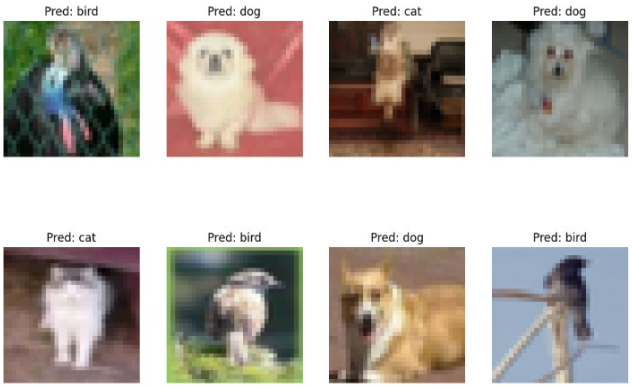

Figure 1: Model Architecture for Dog, Cat, and Bird Classification

5 Transfer Learning

In this section, transfer learning was applied using pre-trained models such as ResNet and VGGNet. These models were fine-tuned with our dataset to improve classification accuracy. Transfer learning helps leverage pre-trained weights learned from large datasets like ImageNet, improving model performance.

6 Optimization of Transfer Learning

The transfer learning models were further optimized using different learning rates, weight decay, and momentum values. The best configurations were chosen based on validation accuracy and loss.

7 Results, Metrics, and Insights

After training the models, the following metrics were analyzed:

- **Training Loss** indicates how well the model is fitting the training data.
- **Training Accuracy** reflects the percentage of correct predictions on the training set.
- **Validation Loss** helps in identifying overfitting, where the model performs well on training data but poorly on unseen data.
- **Validation Accuracy** is the key metric for evaluating the model's generalization performance.

8 Results

After training the models, the following performance metrics were analyzed. The results are summarized in Table 2.

Model	Number of Epochs	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
Baseline Model	15	0.2549	90.26%	1.3370	60.08%
ResNet	15	0.0800	96.99%	0.7792	78.25%
VGGNet	15	0.3135	87.53%	0.7034	76.17%
ResNet	15	0.3929	82.12%	0.5618	79.58%
ResNet	15	0.3593	81.24%	0.5292	80.17%
ResNet	15	0.3350	82.04%	0.5466	79.04%
ResNet	30	0.3269	82.68%	0.5568	80.08%
ResNet	25	0.3157	83.51%	0.5592	79.96%

Table 2: Model Training and Validation Results

9 Conclusion

Based on the results, the **ResNet model(SGD, StepLR, LR0.001, Weight Decay: 1e-4)** with the best configuration achieved the highest validation accuracy of **80.17%**. It outperformed both the baseline CNN and VGGNet models, demonstrating the efficacy of pre-trained models in this classification task.