Project 2: Car Image Classification

Abhay Chaudhary

Objective

The goal of this project is to train a classification model that accurately predicts the **angle** of the car in a given image. Using the **VGG16** architecture, we perform feature extraction and classification for car angle prediction.

Dataset

The **Car Angle Classification Dataset** from Kaggle is used in this project. The dataset contains car images from multiple angles, categorized into 8 classes representing different viewing directions.

Steps Followed

1. Data Analysis & Preprocessing

- **Initial Analysis**:
  - The dataset consists of car images labeled with 8 different angles.
  - Image classes: South, North-West, North, North-East, East, South-East, South-West, West.
- **Preprocessing**:
  - **Image Resizing**: All images resized to 224×224 to match the input size expected by **VGG16**.
  - **Rescaling**: Images are normalized by scaling pixel values between 0 and 1 (`rescale=1./255`).
  - **Augmentation**: Applied augmentation such as rotation and zooming to improve model robustness.

The code snippet for preprocessing is as follows:

```
datagen = ImageDataGenerator(validation_split=0.2,
                             rescale=1./255,
                             rotation_range=10,
                             zoom_range=0.2)
```

2. Model Training

Model Architecture:

- We use a **pre-trained VGG16** model as the feature extractor.
- The VGG16 model's convolutional layers are frozen, while the top layers are retrained for classification.
- Added dense layers with dropout for the final prediction of 8 car angles.

The code snippet for the model architecture is as follows:

```
model_vgg = Sequential()
model_vgg.add(vgg_model)
model_vgg.add(Flatten())
model_vgg.add(Dense(256, activation='relu'))
model_vgg.add(Dropout(0.5))
model_vgg.add(Dense(8, activation='softmax'))
```

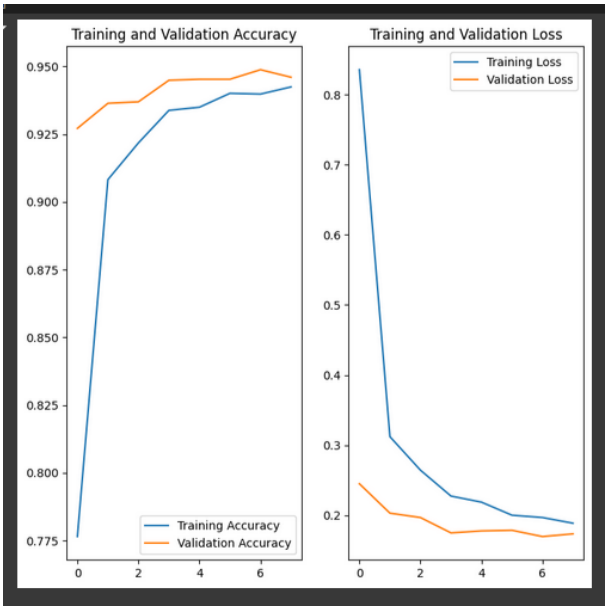

Figure 1: Training and Validation Accuracy and Loss

Training:

- **Optimizer**: Adam
- **Loss function**: Sparse Categorical Crossentropy
- **Evaluation metrics**: Accuracy
- The model is trained over **8 epochs** with an 80-20 training-validation split.

The code snippet for training is:

```
history_vgg = model_vgg.fit(train_generator,
                            validation_data=test_generator,
                            epochs=8)
```

Training Results:

- Accuracy: 95%
- Validation Accuracy: 93%

Below is the plot for Training and Validation Accuracy and Loss:

3. Model Evaluation

The model was evaluated using the validation set. Performance metrics include **accuracy** and **loss**, visualized in the plots.
The code snippet for evaluation is:

```
scores = model_vgg.evaluate(test_generator)
print(f"Validation Loss: {scores[0]}, Validation Accuracy: {scores[1]}")
```

4. API Development

- Developed a **Flask** API to deploy the model. The API accepts an image, predicts the car's angle, and returns the predicted class, confidence score, and the car's direction.
- **Endpoint**:
  - `/predict`: Accepts an image (JPEG/PNG) via POST request, predicts the car's angle, and returns a JSON response with:
    - `predicted_class`: The class representing the car angle.

    * `confidence_score`: The model's confidence in the prediction.
    * `direction`: The corresponding angle direction (e.g., North, South, East, etc.).

The code snippet for the Flask API is as follows:

```
@app.route('/predict', methods=['POST'])
def predict():
    file = request.files['file']
    file.save(file_path)
    predicted_class, confidence_score, direction = predict_car_angle(file_path)
    os.remove(file_path)
    return jsonify({
        "predicted_class": int(predicted_class),
        "confidence_score": float(confidence_score),
        "direction": direction
    })
```

Running the API:

- Install necessary packages:

```
pip install Flask tensorflow numpy
```

- Run the Flask app:

```
python app.py
```

- API will be live at `http://127.0.0.1:5000/predict`.

Conclusion

In this project, a VGG16-based model was trained to predict the car's viewing angle with 95% accuracy. The model was integrated into a Flask API, allowing users to predict the car angle by uploading an image.

Challenges Faced

- Balancing the dataset during training.
- Fine-tuning the VGG16 model without overfitting.

Future Work

- Improve accuracy by experimenting with other architectures (e.g., **ResNet**).
- Enhance the API with features like batch predictions and confidence intervals.