Vehicle Speed Prediction Using Optical Flow and Deep Learning

Introduction

In this project, I aimed to estimate vehicle speeds from video sequences captured in urban environments. This task has significant real-world applications in autonomous driving, traffic analysis, and road safety. I used a two-step approach:

1. **Optical Flow Estimation**: RAFT (Recurrent All-Pairs Field Transforms) was used to compute optical flow from consecutive video frames.
2. **Speed Prediction**: Convolutional Neural Networks (CNNs) were employed to predict vehicle speed based on the extracted optical flow.

Methodology

Step 1: Optical Flow Estimation Using RAFT

Why RAFT?

RAFT was chosen for its robustness, accuracy, and ability to handle large displacements and brightness changes. Trained on the KITTI dataset, RAFT is particularly suited for urban driving scenarios, making it an ideal choice for our task.

Implementation Details

- **Frame Preparation**: Consecutive frames were extracted from video sequences to serve as inputs for optical flow computation.
- **Model Setup**:
  - The RAFT model was loaded with pre-trained weights (`raft-things.pth`).
  - Frames were padded to ensure compatibility with RAFT's input requirements.
- **Flow Computation**: For each pair of consecutive frames, the model computed the optical flow, saved in HSV format for visualization.

Visualization

A sample optical flow visualization is shown below:

The optical flow images effectively capture pixel-wise motion, serving as an informative representation for downstream speed prediction.

Step 2: Speed Prediction Using CNNs

Data Preprocessing

- Optical flow images were resized to 224x224 and normalized using ImageNet statistics.
- Corresponding speed labels were loaded and paired with the preprocessed flow images.
- Data was saved in `.npy` format for efficient loading during model training.

Dataset

- **Training Dataset**: Preprocessed images and speeds from `train_flows` and `train_speeds.txt`.
- **Test Dataset**: Preprocessed images and speeds from `test_flows` and `test_speeds.txt`.

Model Architectures

1. **ResNet18**:
   - Pre-trained ResNet18 was modified by replacing the final fully connected layer with a single-output linear layer for speed regression.
   - Advantage: Strong feature extraction capabilities.
2. **EfficientNet**:
   - EfficientNet-B0 was adapted similarly for regression tasks.
   - Advantage: Compact design with efficient parameter usage.

Training Setup

- **Loss Function**: Mean Squared Error (MSE) to penalize deviations from true speed values.
- **Optimizer**: Adam optimizer with a learning rate of 0.001.
- **Training Duration**:
  - ResNet18: 12 epochs.
  - EfficientNet: 10 epochs.
- **Hardware**: Training and inference were conducted on a GPU to accelerate computation.

Results

Training Performance

ResNet18:

- **Training Loss**: Loss values decreased steadily, indicating effective learning.
  Example loss values:
  - Epoch 1: 4.19
  - Epoch 6: 2.74
  - Epoch 12: 0.46

EfficientNet:

- Training was slightly more unstable compared to ResNet18.
  Example loss values:
  - Epoch 1: 24.41
  - Epoch 6: 4.60
  - Epoch 10: 4.12

Evaluation on Test Data

The models were evaluated using mean test loss on unseen data.

- **ResNet18**: Achieved a mean test loss of **~0.69**, indicating accurate speed predictions.
- **EfficientNet**: Test loss was slightly higher at **~1.55**, showing relatively lower accuracy.

Discussion

Model Performance

- ResNet18 outperformed EfficientNet in both training and test evaluations. This is likely due to its mature feature extraction capabilities and architectural simplicity, which allowed faster convergence and better generalization.
- EfficientNet struggled with convergence, possibly due to its compact architecture being more sensitive to hyperparameter tuning for regression tasks.

Impact of Optical Flow

- The quality of optical flow significantly influenced the models' ability to predict speed. RAFT’s robust flow estimation ensured that the input to the CNNs captured motion patterns effectively, leading to successful predictions.

Challenges

- **Data Imbalance**: Variations in speed distribution across the dataset could have affected model performance.
- **Computation**: Optical flow estimation is computationally intensive, requiring substantial GPU resources.