Methodology and Implementation Details

1. Task 1: Single-View Rider Intention Prediction

1.1 Methodology Overview

For Task 1, the objective is to predict rider maneuvers from single-view video features using a ResNet50-based feature extractor. The approach involves building a model to classify these maneuvers based on features extracted from frontal-view videos.

1.2 Data Preparation

1. **Dataset**: The dataset consists of frontal-view videos, each providing features extracted using ResNet50. The features are stored in .npy files.
2. **Dataset Class**:
    - A custom NpyDataset class is implemented to handle the loading and preprocessing of these features. It pads or truncates the feature sequences to a fixed length (300 frames).
    - The dataset is split into training, validation, and testing sets. Empty or improperly formatted files are filtered out.
3. **DataLoader**:
    - A DataLoader is configured to batch and shuffle the data during training. The collate_fn function is used to handle varying sequence lengths by padding them to the maximum length in a batch.

1.3 Model Architecture

1. **Model Definition**:
    - A simple neural network (SimpleNN) is used, consisting of two fully connected layers:
        - An initial layer with 2048 input features and 512 output features.
        - A final layer with 6 output features corresponding to the 6 maneuver classes.
    - The model processes the input sequences by averaging over the sequence dimension and applying ReLU activation before producing class scores.
2. **Training**:
    - The model is trained using the Adam optimizer and cross-entropy loss function.
    - The training process includes saving the best-performing model based on validation F1 score.
3. **Evaluation**:
    - After training, the best model is used to generate predictions on the test set.
    - Results are saved in a CSV file with one-hot encoded maneuver predictions.

2. Task 2: Multi-View Rider Intention Prediction

2.1 Methodology Overview

For Task 2, the goal is to improve maneuver prediction accuracy by leveraging multi-view video features (frontal, left side-mirror, right side-mirror). This approach involves aggregating features from multiple views to enhance the prediction.

2.2 Data Preparation

1. **Dataset**: Multi-view features are extracted from videos and stored in .npy files, organized by view type.
2. **Dataset Class**:
   - The MultiViewDataset class is designed to handle multi-view data, loading features from each view and padding or truncating them to a fixed sequence length (300 frames).
   - It supports both training/validation and testing modes.
3. **DataLoader**:
   - A DataLoader is created with a custom collate_fn to handle multi-view features, padding them as needed and collating them into batches.

2.3 Model Architecture

1. **Model Definition**:
   - The MultiViewModel is a neural network with a similar architecture to the single-view model, but designed to aggregate features from multiple views.
   - The input is taken from the frontal view, which is used to make the final prediction. Additional views are loaded but not directly used in the current architecture.
2. **Training**:
   - The model is trained with the Adam optimizer and cross-entropy loss.
   - The training loop includes saving the best model based on validation F1 score.
3. **Evaluation**:
   - The best model is used to generate predictions on the test set.
   - Results are saved in a CSV file with one-hot encoded maneuver predictions.

Conclusion

Both Task 1 and Task 2 leverage neural network models to predict rider maneuvers from video features. Task 1 uses a simple model with single-view features, while Task 2 incorporates multi-view data to potentially enhance prediction accuracy. The models are trained and evaluated using F1 score as a primary metric, with results saved for further analysis.