كليـــــــــة العلوم السملالية - مراكش

FACULTÉ DES SCIENCES SEMLALIA - MARRAKECH

Predicting Driver Lane Change Maneuvers Using Federated Learning

Done by:

Maghrane Wail
El arachi Mounia
Ait Bigane Youssef
Eddarbali Niama

Supervised by:

Mr. Abdellatif Moussaid

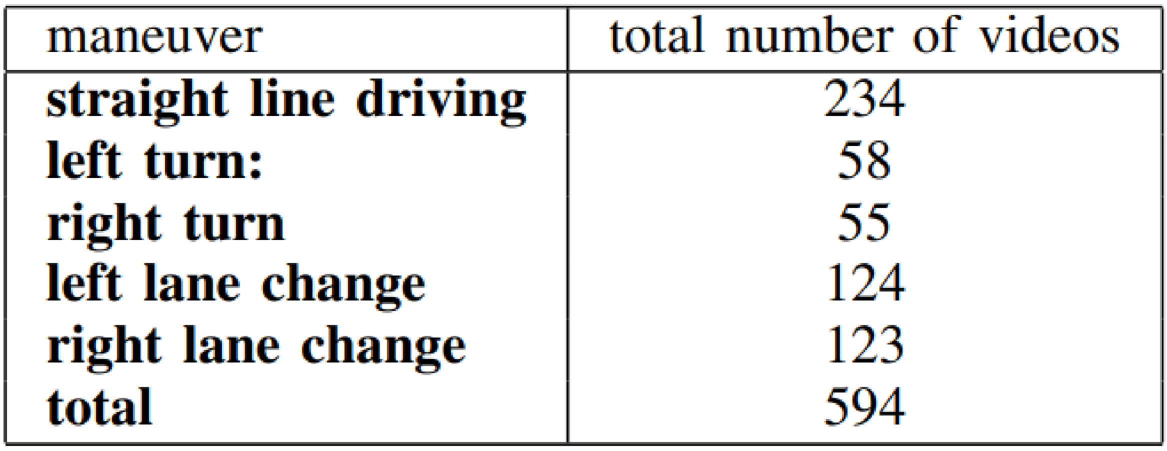
Abstract:
This report examines the efficacy of federated learning (FL) and Long Short-Term Memory (LSTM) models in the context of video classification. Initially, attempts to implement a TensorFlow Federated environment were unsuccessful, leading to a manual approach using LSTM models across three clients.

Data Understanding:

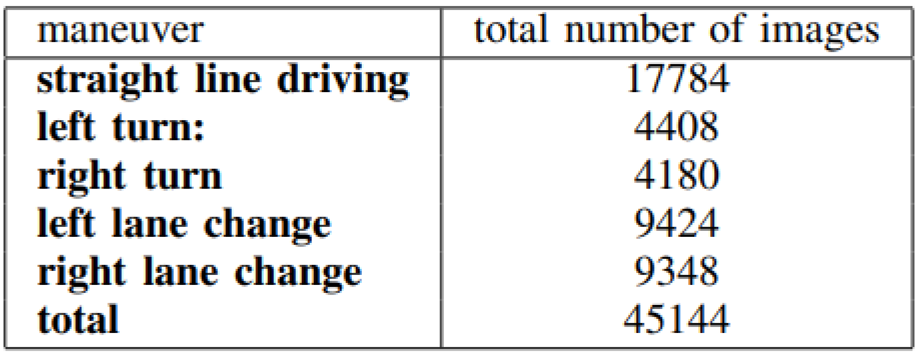
A. Dataset Description

Source: Brain4Cars team, featuring data from 10 different drivers in various situations and roads.

Classes: 5 classes, totaling 594 maneuvers (left lane change, right lane change, left turn, right turn, straight-line drive), each with 5-second videos.



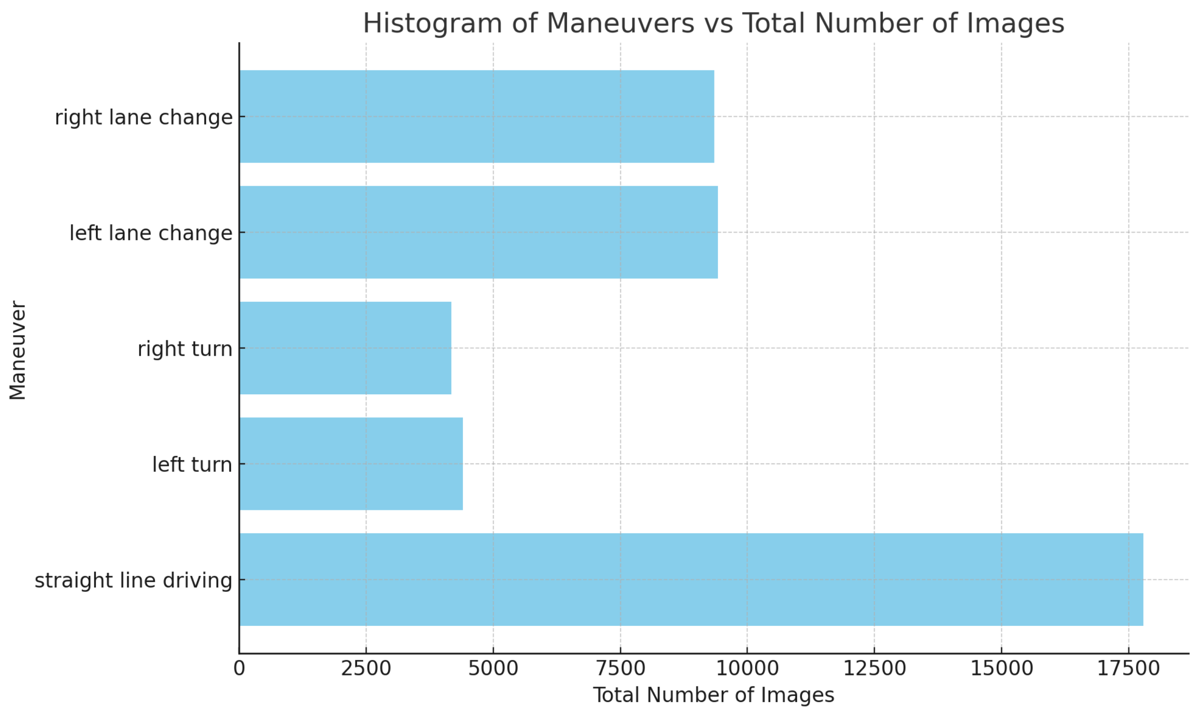
maneuver	total number of videos
straight line driving	234
left turn:	58
right turn	55
left lane change	124
right lane change	123
total	594



maneuver	total number of images
straight line driving	17784
left turn:	4408
right turn	4180
left lane change	9424
right lane change	9348
total	45144

Outside Features:

ID Video: Identifier for each video.
Lane Left: Number of empty lanes on the left side.
Lane Right: Number of empty lanes on the right side.
Road Artifact: Binary indicator of road artifacts like intersections.
Speed: Seven speed segments (V1 to V7) for each 5-second video.
Manoeuvre:



Data Preparation:

Frame Extraction: Each video is split into 76 frames, representing 5 seconds at 15 frames per second.
Feature Generation:
Lane Information: For each frame, the number of empty lanes on both sides (left and right) is recorded.
Road Artifact: A binary feature indicating the presence of road artifacts like intersections.
Speed Sequence: The 5-second video is divided into 7 speed segments (V1 to V7), each represented across 11 frames, totaling 76 frames.
Final CSV Structure: Each video ID is repeated 76 times to correspond with the number of frames, along with the features obtained from processing the frames through

DenseNet121(1023 feature)and others like road artifact, lane right, lane left, speed features, and manoeuvre label.

LSTM model

The model architecture employs a Long Short-Term Memory (LSTM) neural network with 64 units, followed by a softmax activation layer for multi-class classification. The input shape is specified as (1, train_features.shape[1]), indicating a sequence of features for each sample. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss, optimising for accuracy during training. To select the best-performing model, the training process utilises the ModelCheckpoint callback, which saves the model with the highest validation accuracy. During training, this callback continually monitors the validation accuracy and saves the model's weights to a file named 'best_model.h5' whenever an improvement is observed. This technique ensures the retention of the most accurate model based on the validation dataset throughout the training epochs.

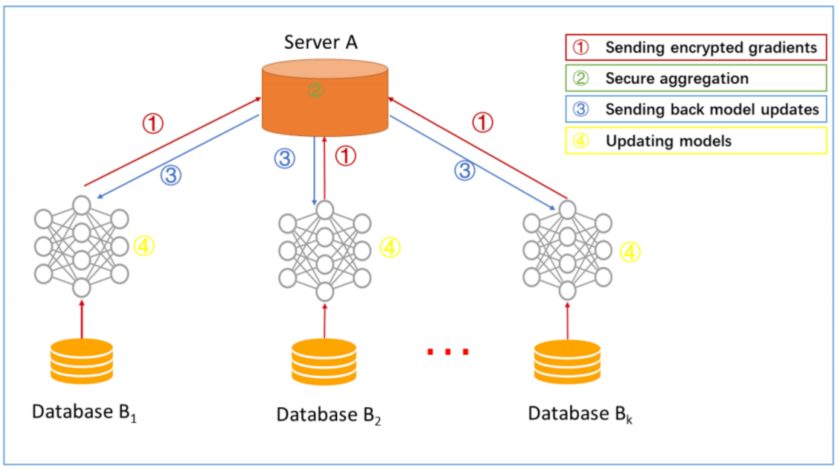
Federated Learning

Using federated learning for predicting where a driver will turn based on pictures of their gestures is a great idea! Here's a simplified step-by-step explanation of how you can approach this task:

Data Collection: First, gather your dataset of pictures showing drivers' gestures when turning corners. Make sure you have a good variety of images to represent different situations and gestures.

Data Labeling: Label each image to indicate the corresponding turning behavior, such as "left turn," "right turn," or "no turn."

Federated Learning Setup:



Divide your dataset into smaller subsets or "partitions" to represent different drivers or groups of drivers.
Each partition remains on the device or computer where it's collected, and no raw data is shared centrally.

Model Training:

Deploy a machine learning model (like a deep neural network) on each driver's device.
Train the models on the local dataset (pictures and labels) without sending the raw data to a central server. Each driver's model learns from their own pictures.
Model Update Sharing:

Periodically, the models share updates, like improvements or changes they've learned from their local data.
These updates are shared with a central server or between the devices using privacy-preserving techniques like federated learning.
Central Aggregation:

The central server aggregates all the updates received from different drivers' models without seeing the raw data.
This aggregated update is used to improve the global model's accuracy.

Model Evaluation:

Periodically, assess the global model's performance on a separate evaluation dataset.
Adjust the federated learning process to further train the models for better accuracy.
Prediction:

Once your global model is trained and has learned from various drivers' behaviors, you can use it to predict a new driver's behavior based on their gestures in real-time.

Federated learning allows you to train a model that can predict turning behaviors without compromising the privacy of the drivers' data. It keeps the data decentralized, so individual drivers' information stays on their devices, making it a privacy-friendly and effective way to develop a predictive model for your specific task.

Instructions :

- **Challenges with TensorFlow Federated:** Encountered difficulties while trying to implement the TensorFlow Federated environment for our project.

- **Adaptation to Manual, Client-Based Approach:** Due to these challenges, we shifted to a manual, client-based approach for video classification.

- **Consistent LSTM Architecture Across Clients and Dataset:** Implemented the same LSTM architecture for each client, mirroring that used for the entire dataset, ensuring consistency and comparability in model structures.

- **Data Preparation and Division:** The dataset was split into three parts, with each part allocated to a different client for training an individual LSTM model, facilitating diverse learning experiences.

- **Checkpoint Implementation in Training:** Incorporated checkpoints in the training process of each LSTM model to capture the most effective model state during training.

- **Modification in LSTM Input Due to Data Shape Constraints:** We did not set the LSTM input to 77, despite the intent to predict manoeuvres for the 77 frames of the videos. This was due to the shape of the data, which could not be evenly distributed by 77 when splitting into training and validation sets for the checkpoint callback.

- **Feature Encoding and Selection Focus:** Employed LabelEncoder for manoeuvre labels, concentrating on the numerical features within the dataset for model training.

- **Uniformity in LSTM Model Design:** Each model, distributed across clients, followed the same LSTM architecture, maintaining a standardised approach in the learning process.

- **Global Model Through Weight Averaging:** Post training, the weights of the three models were averaged to construct a global model, integrating insights from each individual client model.

- **Comprehensive Testing and Evaluation:** The global model was tested using videos of a driver the none of the models has ever seen (from test folder), achieving a Test Accuracy of 34.76%, a critical phase for assessing its efficacy.

- **Comparative Analysis with a Standalone LSTM Model:** A standalone LSTM model, sharing the same architecture, underwent similar testing, demonstrating a superior Test Accuracy of 45.18%.

Conclusion:

The findings indicated that the standalone LSTM model, even with the modified input settings, was more effective than the client-based federated learning approach.