TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Vehicle Motion Prediction in Autonomous Vehicles

Seminar Electromobility SS 2022

Institute of Electromobility
University of Kaiserslautern

Nilay Gawade	421670
Kaiwalya Patkar	422429
Sarvesh Telang	422444

Agenda

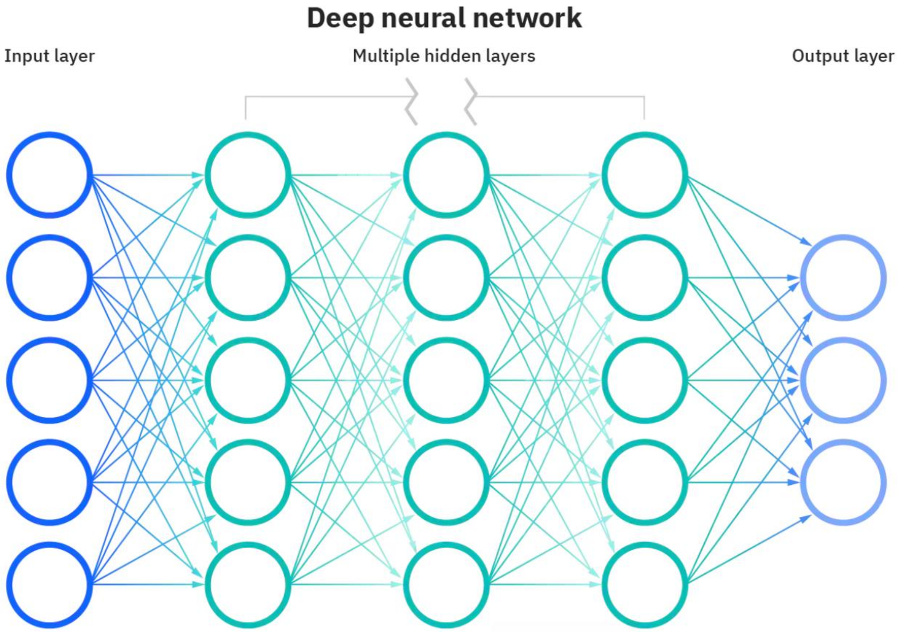
- **Selection of Motion prediction model**
  - Why Neural Network?
  - FCN Keras

- **Implementation**
  - Changed Hyperparameters
  - Activation functions
  - Random Search- Hyperparameter Tuner

- **Model Results**

Selection of Motion prediction model

Why Neural Network Model?



Source: [2]

- Computing system with interconnected nodes (neurons in Human brain)
- Algorithms can recognize hidden patterns and correlations in raw data
- Learning through several iterations and improvement
- High accuracy
- Accuracy = Number of correct predictions

Selection of Motion prediction model

- **Prediction techniques based on Neural Networks**
  - RNN (Recurrent Neural Networks)
    - LSTM Network (Long Short-Term Memory)
    - GRUs (Gated Recurrent Unit)
  - CNN (Convolutional Neural Networks)
    - FCN (Fully Convolutional Network
- **FCN Keras**

  Possibility of accepting input and generating output of image-like data

  Maintains spatial relationship of input data

  Contains 1x1 convolutions that perform the task of dense layers

  Possible to feed variable input

Implementation

Selection of training data

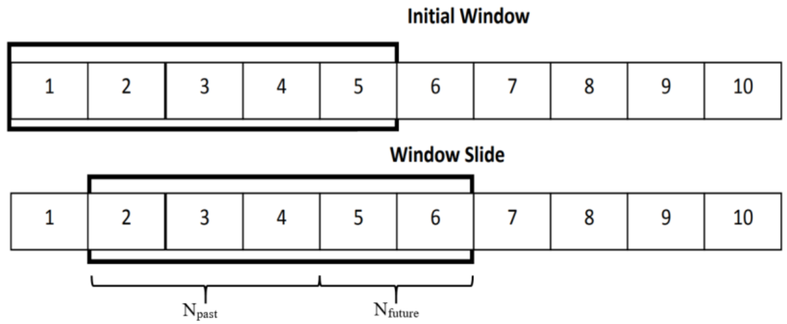
- inD dataset (Intersection Drone Dataset)
- Train Data: Recording 18 and 24 from Location set 2

Data Processing and Normalization

- Noisy, unnecessary, and large unstructured data
- Downsampling to shorten the sample time and save RAM usage.
- Min- max Scaling

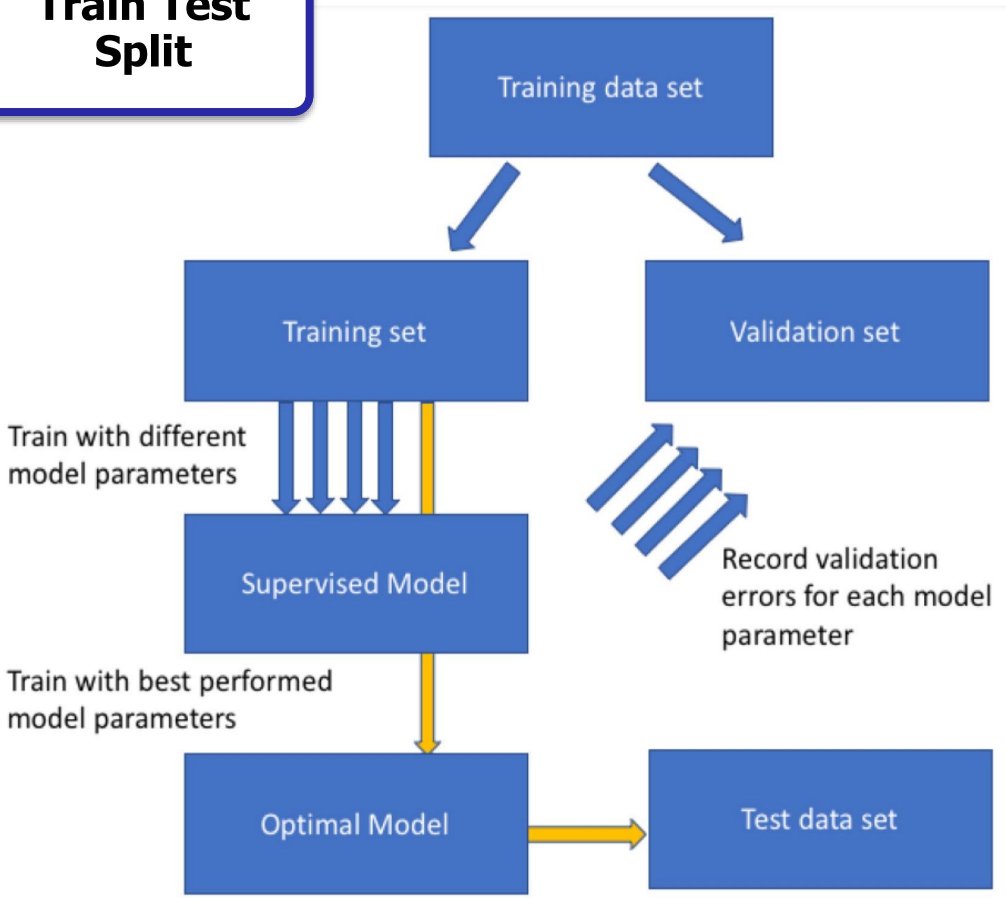
Data Preparation

- Sliding window technique



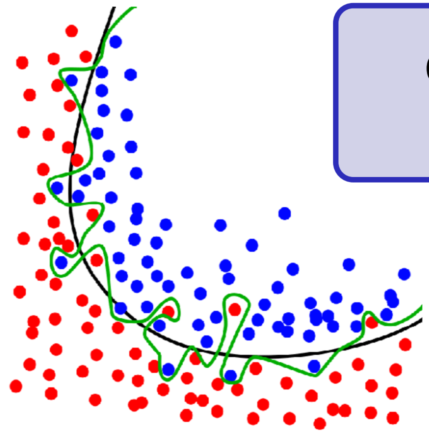
Train test split

Train Test Split

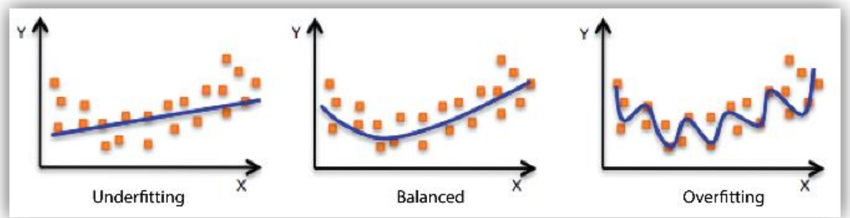


Source: [6]

Overfitting and underfitting



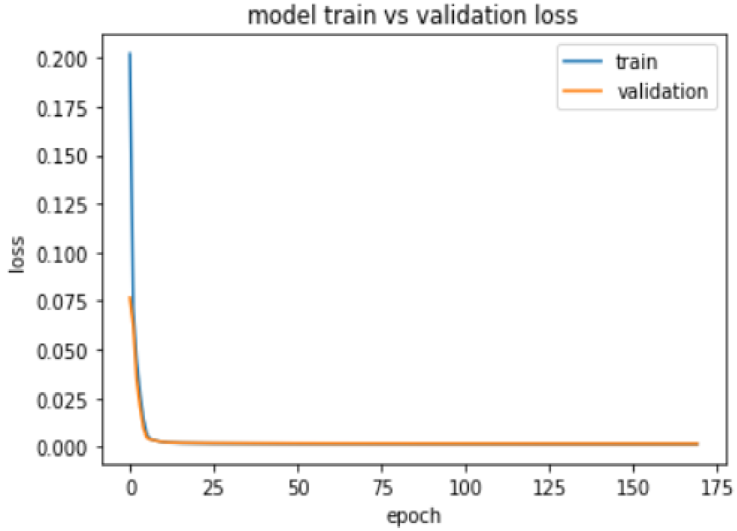
Source: [7]



– **Overfitting:** Good performance on the training data, poor generalization to other data.
– **Underfitting:** Poor performance on the training data and poor generalization to other data

Trained model result

```
1856/1856 [==============================] - 6s 3ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 168/170
1856/1856 [==============================] - 6s 3ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 169/170
1856/1856 [==============================] - 6s 3ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 170/170
1856/1856 [==============================] - 6s 3ms/step - loss: 0.0014 - val_loss: 0.0015
```



- **The validation loss: How well the model fits new data**
- **Training loss: How well it matches training data.**
- **Overfitting : Validation loss > training loss**
- **Underfitting: Validation loss < the training loss.**

Implemented Model and Results

FCN Keras Model

```
# define multi-layer perceptron model
model = Sequential()

#Input layer
model.add(Dense(279, activation='sigmoid', input_shape=(n_input,)))

#Hidden layers
model.add(Dense(162, activation='sigmoid'))
model.add(Dense(45, activation='sigmoid'))

#Output layer
model.add(Dense(n_output, activation='linear'))

#Optimizer and learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5), loss='mse')

# fit the keras model on the dataset
history =model.fit(xTrain, yTrain, epochs=170 ,batch_size=72,
                   verbose=1, validation_data=(xTest, yTest))
```

ADE = 1.004 m

FDE = 0.888 m

AHE = 5.5 degrees

Changed Hyperparameters

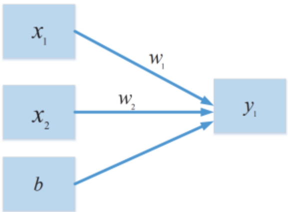
FCN Keras Model

```
# define multi-layer perceptron model
model = Sequential()

#Input layer
model.add(Dense(279, activation='sigmoid', input_shape=(n_input,)))

#Hidden layers
model.add(Dense(162, activation='sigmoid'))
model.add(Dense(45, activation='sigmoid'))

#Output layer
model.add(Dense(n_output, activation='linear'))

#Optimizer and learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5), loss='mse')

# fit the keras model on the dataset
history =model.fit(xTrain, yTrain, epochs=170 ,batch_size=72,
                   verbose=1, validation_data=(xTest, yTest))
```

- ***No. of Neurons***
- ***No. of Hidden layers***
- ***No. of epochs***
- ***Activation functions of Input, Output and Hidden layers***
- ***Learning rate***

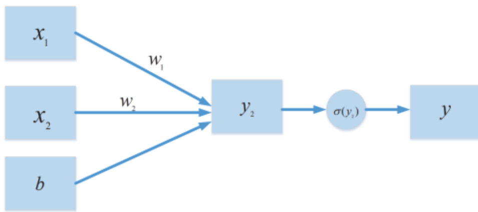
Activation Functions

- *To add non-linearity to the neural network*



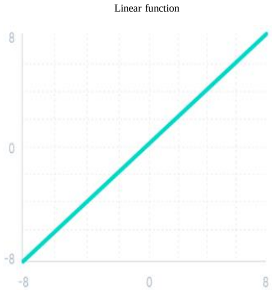
Without activation function

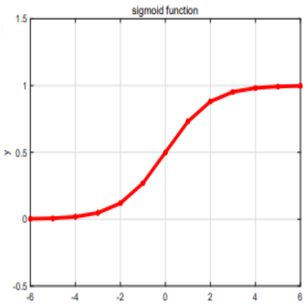
$$y_1 = w_1 x_1 + w_2 x_2 + b$$

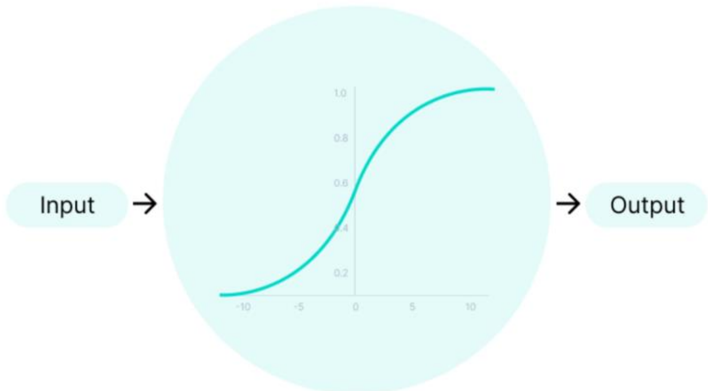


With activation function

$$y_2 = w_1 x_1 + w_2 x_2 + b$$
$$y = \sigma(y_2)$$







Activation Function

Source: [3]

Popular types of activation functions

- *Binary Step*
- *Linear*
- *Sigmoid*
- *Tanh*
- *ReLU*
- *Leaky ReLU*
- *Parameterised ReLU*
- *Exponential Linear Unit*
- *Swish*
- *Softmax*

Random Search – Tuning of Hyperparameters

```
def build_model(hp):
    model = keras.Sequential()
    for i in range(hp.Int('num_layers',2,4)):
        model.add(layers.Dense(units=hp.Int('units_' + str(i),
                                            min_value=45,
                                            max_value=350,
                                            step=32),
                               activation='sigmoid'))
    model.add(layers.Dense(1, activation='linear'))
    model.compile(
        optimizer=keras.optimizers.Adam(
            hp.Choice('learning_rate', [1e-4, 1e-5])),
        loss='mean_absolute_error',
        metrics=['mean_absolute_error'])
    return model

tuner = RandomSearch(
    build_model,
    objective='val_mean_absolute_error',
    max_trials=5,
    executions_per_trial=3,
    directory='project',
    project_name='Emobilitytune')

tuner.search_space_summary()

tuner.search(xTrain, yTrain,
             epochs=50,
             batch_size=80,
             validation_data=(xTest, yTest))

tuner.results_summary()
```

- **Output:**

```
Trial 361 Complete [00h 04m 00s]
val_mean_absolute_error: 0.1795041263103485

Best val_mean_absolute_error So Far: 0.1759001612663269
Total elapsed time: 00h 09m 33s

Search: Running Trial #362

Value             |Best Value So Far |Hyperparameter
3                 |3                 |num_layers
301               |141               |units_0
333               |237               |units_1
45                |333               |units_2
1e-05             |0.0001            |learning_rate
```

- Random combinations of the hyperparameters are used to find the best solution
- Set values in ranges
- Results from the previous iteration are not used to decide the next hyperparameter value candidates- Bayesian Optimization

Final Evaluation

Prediction Testing

- Testing on blind data set and generalization performance
- Test Data: Recording 28

Collecting Ground Truth and final Evaluation

- Validation against ground truth for actual performance
- Error metrics: ADE, FDE, AHE

References

[1] https://medium.com/analytics-vidhya/comparison-of-hyperparameter-tuning-algorithms-grid-search-random-search-bayesian-optimization-5326aaef1bd1

[2] https://www.ibm.com/cloud/learn/neural-networks#:~:text=Neural%20networks%20reflect%20the%20behavior,machine%20learning%2C%20and%20deep%20learning.

https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b

[3] https://www.v7labs.com/blog/neural-networks-activation-functions

[4] Heru Wahyu Herwanto, Anik Nur Handayani, Aji Prasetya Wibawa, Katya Lindi Chandrika, Kohei Arai, "Comparison of Min-Max, Z-Score and Decimal Scaling Normalization for Zoning Feature Extraction on Javanese Character Recognition," IEEE, 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE), October 2021.

[5] H.S. Hota , Richa Handa and A.K. Shrivas, "Time Series Data Prediction Using Sliding Window Based RBF Neural Network," International Journal of Computational Intelligence Research, Volume 13, Issue 5, pp. 1145-1156, 2017.

[6] Yun Xu, and Royston Goodacre, "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," Springer, Journal of Analysis and Testing, October 2018.

[7] Qipei Li, Ming Yan, and Jie Xughg, "Optimizing Convolutional Neural Network Performance by Mitigating Underfitting and Overfitting," IEEE/ACIS 19th International Conference on Computer and Information Science (ICIS), Shanghai, China, June 2021.

[8] Yingying Wang, Yibin Li, Yong Song, and Xuewen Rong, "The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition," MDPI Applied Sciences, China, March 2020.

Thank you