Knowledge-Distilled Temporal Convolutional Networks for Transportation Mode Detection Using Edge-enabled Consumer Smartphone Sensors

Xiaofu Chen, Weizhi Meng, *Senior Member, IEEE*, and Wenjuan Li *Senior Member, IEEE*

***Abstract*—Nowadays, smartphones and their built-in sensors not only have become ubiquitous but also serve as invaluable tools for collecting a diverse array of data. When we navigate through the complexities of Transportation Mode Detection (TMD), two key challenges emerge as: a) the need to balance computational efficiency with model accuracy, and b) the importance to capture temporal dependencies in time-series data for accurate mode classification. To address these issues, we propose TRTCN, a novel algorithm designed to transportation mode detection via edge-enabled consumer smartphone sensors. This algorithm employs a Temporal Convolutional Network (TCN) and integrates data from lightweight sensors housed within the smartphone. In our proposed model, we employ knowledge distillation to construct a framework that comprises both a teacher model and a student model. The student model, represented as a Convolutional Neural Network (TRTCN-CNN), can help reduce the number of parameters required for the model, thereby decreasing the training time. A multi-Head attention mechanism is also incorporated to fine-tune the accuracy of the model's accuracy. The student model has exceptional performance, with an average precision of 88.32% on the SHL 2018 dataset and an astonishing 96.04% on the SHL 2021 dataset. Our work shows that the transportation mode detection can be enhanced via the existing edge-enabled consumer smartphone sensors.**

***Index Terms*—Edge Intelligence, Deep learning, Transportation mode detection, Temporal convolutional networks , Multi-head attention, Knowledge distillation, Consumer device**

I. Introduction

With the accelerated advancement of technology, smartphones and their integrated sensors are becoming increasingly prevalent in everyday life. These sensors can offer a vast quantity of information [1], [2], such as a user's mobility patterns, environmental data, etc. By using this kind of information, researchers have developed various methods to detect and predict users' modes of transportation, such as stationary, walking, biking, riding a motorcycle, taking a car, bus, metro, and railway [3]. However, accurate transportation mode detection from sensor data remains a challenge, particularly when dealing with large, diverse, and complex data [4].

X. Chen is with the SPTAGE Lab, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby 2800, Denmark.
W. Meng is with School of Computing and Communications, Lancaster University, United Kingdom, and with Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark. (Corresponding author) E-mail: weizhi.meng@ieee.org
W. Li is with the Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong SAR, China.

Transportation mode detection (TMD) is a critical area of research within the broader context of urban computing and smart cities. Utilizing various sensor data, typically collected from smartphones or wearable devices, TMD aims to identify the mode of transportation that an individual is using, such as walking, cycling, driving, or taking public transit. Accurate detection has far-reaching implications, offering valuable insights for urban planning, traffic management, and personalized travel recommendations. While significant advancements have been made in developing models that are both accurate and efficient, several critical challenges and limitations continue to hinder the wide adoption and the effectiveness of existing transportation mode detection solutions. These shortcomings are outlined as follows:

- **Balancing Model Complexity and Accuracy:** A common challenge in the field of transportation mode detection is striking the right balance between model complexity and predictive accuracy [5]. Many existing solutions either employ intricate models that demand significantly computational power, making them unsuitable for real-time deployment on devices with limited resources such as smartphones or smartwatches. These devices often lack enough computational capabilities toward more robust systems, thereby limiting the practicality of complex models for everyday usage. On the other hand, some approaches opt for simpler models to save computational costs, but this often comes at the expense of accuracy, rendering them less effective for real-world applications. Addressing this issue is crucial for the broader adoption of transportation mode detection technologies, especially for applications that require an immediate response and high reliability, such as healthcare monitoring, emergency response systems, and various consumer electronics.
- **Fine-grained Classification and Temporal Dependencies:** The inability of current models in the field to capture the temporal dependencies that exist within time-series data is one of the most prevalent issues [6]. This limitation is especially problematic when it comes to accurately identifying and differentiating transportation modalities that are (very) similar. For instance, the models frequently have difficulty distinguishing between walking and running–two activities that share similar patterns but differ in intensity and tempo. Also, the models encounter difficulties in accurately categorizing vehicular modes of transportation, such as train and subway travel, which

may share overlapping characteristics, such as speed and stop frequency, but are fundamentally distinct in other regards, such as route and environmental variables. These flaws not only reduce the model's overall accuracy but also limit its applicability in real-world scenarios where accurate identification of transportation modes is essential for a variety of applications, ranging from personal fitness monitoring to urban planning.

Recent advancements have leveraged machine learning techniques, especially deep learning, to enhance the accuracy of detection. For instance, convolutional neural network [7], long-short-term memory recurrent neural network [8] and variant of the temporal convolutional network [9]. However, the majority of existing methods are either too complex [10], [11], resulting in inefficiency in practical applications, or too imprecise [1] to satisfy the needs of practical applications. The main contributions of this paper are summarized as follows:

- We first develop and implement an innovative transportation mode detection approach, pioneering a novel model named *TwinResTCN (TRTCN)*. Our model can enhance the traditional Temporal Convolutional Network (TCN) by capturing temporal dependencies and handling diverse transportation patterns via edge-enabled consumer smartphone sensors.
- In addition, our method via edge intelligence also integrates knowledge distillation techniques. By distilling knowledge from our TRTCN model, we are able to transfer essential insights to a simpler Convolutional Neural Network (CNN)-based model. Thus, it not only streamlines the model but also ensures a high level of accuracy, adeptly addressing the prevalent trade-off between model simplicity and performance.
- We deploy multi-head self-attention mechanisms to weigh the significance of different features and time steps. This ensures that the model focuses on the most crucial aspects of the data, leading to improved recognition of transportation modes, especially in challenging scenarios where modes may appear similar.
- We evaluate the performance of our proposed TRTCN model on two large public datasets, namely the SHL 2018 dataset [12] and the SHL 2021 dataset [13]. Experimental results demonstrate that our TRTCN is much superior to baseline algorithms including MLP, CNN, RNN, LSTM [14], DeepConvLSTM [15], Decision Tree, TCMH [16] and T2Trans [9].

The structure of this paper is outlined as follows: In Section II, we survey related studies on transportation mode detection, emphasizing studies that utilize various sensor data in conjunction with machine learning or deep learning techniques. Section III delineates the architecture of the TRTCN algorithm and provides an in-depth discussion of the design intricacies of our proposed method. Also, this section details our application of the knowledge distillation technique within the framework. Section IV elaborates on the experimental setup and introduces the datasets used for evaluation. We then present and discuss the evaluation results. Section V concludes and encapsulates the essence of our research.

II. Related Work

A. GPS-Based and Multi-Sensor Approaches

In the research community, the earliest methods for transportation mode detection heavily relied on GPS data. For instance, Tao et al. [17] and Stenneth et al. [18] utilized Bayesian Belief Networks and Random Forest algorithms to classify various transportation modes based on GPS data. Although these models achieved promising accuracy, they suffered from high power consumption and were ineffective in GPS-denied environments. To mitigate these issues, Bjerre-Nielsen et al. [19] extended this approach by incorporating Wi-Fi and Bluetooth data alongside GPS. While this combination improved accuracy (89%) and reduced power consumption, these methods were still constrained by network availability and required external data sources.

These GPS-based methods laid the foundation for transportation mode detection but highlighted the need for more power-efficient solutions. Our approach addresses this gap by relying solely on low-power, built-in smartphone sensors like accelerometers and gyroscopes, which do not require external signals such as GPS or Wi-Fi signals, making the method more suitable for real-world and edge computing scenarios.

B. Sensor-Based and Lightweight Machine Learning Methods

To tackle the high power consumption of GPS-based methods, many researchers turned to lightweight sensor data such as accelerometers. Hemminki et al. [20] and Ashqar et al. [21] leveraged hierarchical classifiers on accelerometer data to detect transportation modes. These methods reduced power usage but faced challenges with computational efficiency and robustness, as sensor noise and environmental variations could greatly degrade performance. In contrast, Sağbaş and Balli [22] achieved an accuracy rate of 99.4% using the Random Forest algorithms on smartphone sensor data, demonstrating that sensor-based approaches can reach high accuracy without excessive power costs.

However, these traditional machine learning models often rely heavily on hand-crafted features, which can be computationally expensive and may not generalize well across different environments or user behaviors. This limitation motivates our proposal of a deep learning-based approach, leveraging temporal convolutional networks (TCNs) to automatically extract features from sensor data, thus reducing the reliance on manually engineered features and enhancing model adaptability.

C. Deep Learning Methods for Temporal and Multi-Modal Data

The application of deep learning has significantly improved the transportation mode detection by better capturing temporal dependencies in sensor data. Vu et al. [23] introduced recurrent neural networks (RNNs) for feature learning from accelerometer data, while Dabiri et al. [7] employed convolutional neural networks (CNNs) to learn features from multi-channel GPS segments. Both studies demonstrated the potential of deep learning in transportation mode detection, but they were still limited to experimental settings, and the models often lacked efficiency for real-time, edge-based deployment.

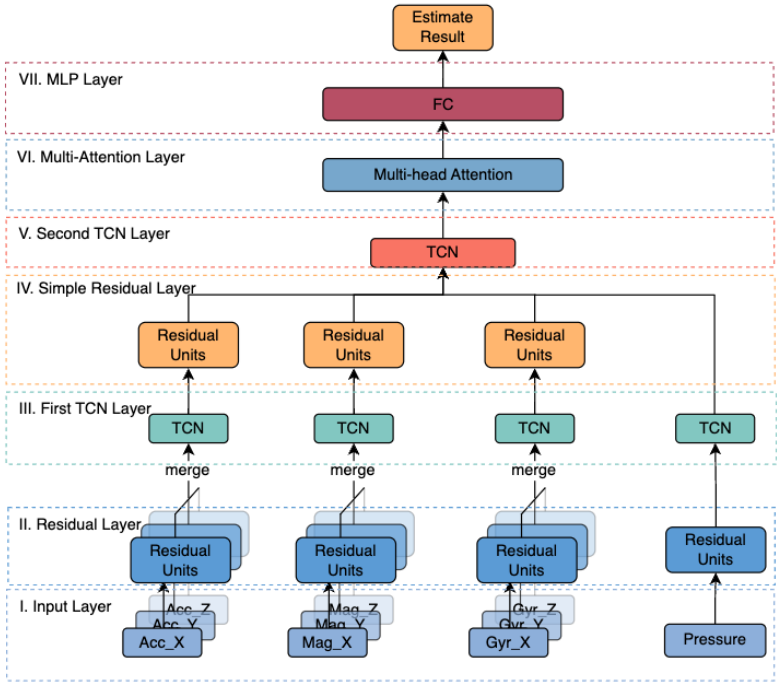

Fig. 1. Overall architecture of the TRTCN model (with edge intelligence support)

Hong et al. [24] combined LSTM networks with GPS sensor data, achieving good accuracy and computational efficiency. Alferaidi et al. [25] extended this by applying a distributed CNN-LSTM model for IoT-based vehicle intrusion detection, reaching a high accuracy rate of 99.7%. While these models perform well, they primarily focus on external data (e.g., GPS) or specific scenarios (e.g., vehicle detection), which may not be applicable to general smartphone sensor data in diverse environments. Li et al. [26] introduced an enhanced Spatial-Temporal Recurrent Neural Network (SCNN) framework. This framework serves as the cornerstone of an innovative hybrid spatial-temporal model for lane detection. Cardoso-Pereira et al. [27] introduced polar ordinal patterns with amplitude information (POPAyI), a strategy that bases its design on the ordinal pattern (OP) transformation applied to mobility-related time series. In [28], the authors developed a robust transportation mode detector based on a convolution neural network (CNN) via a feature modeling technique, feature fusion and cloning techniques.

In contrast, our work proposes the use of Temporal Convolutional Networks (TCNs) to capture temporal dependencies in a causal and feed-forward manner, making it computationally efficient and suitable for edge-enabled smartphones. By utilizing only lightweight smartphone sensors (e.g., accelerometers, gyroscopes), we are able to reduce the computational and the energy costs associated with more complex or external data sources like GPS.

D. Hybrid Models and Multi-Head Attention Mechanisms

Several hybrid models have been proposed that combine the strengths of traditional machine learning and deep learning approaches. For example, Francisco et al. [15] designed a hybrid model that integrates convolutional layers with LSTM networks for robust feature learning, while Chen et al. [29] employed a hybrid approach that merges feed-forward neural networks with LSTM layers. These models effectively captured both spatial and temporal dependencies in sensor data but introduced higher computational complexity, which may limit their deployment on edge devices.

To further enhance the performance, researchers have started incorporating attention mechanisms. Kalatian et al. [10] employed residual networks with Wi-Fi signals, and Wang et al. [30] included sound data in their models to augment detection capabilities. Although these methods demonstrated improved accuracy, they introduced additional challenges, such as increased power consumption and reliance on network availability.

In this work, our proposed approach addresses these limitations by integrating a multi-head attention mechanism into the TRTCN model, allowing it to selectively focus on relevant temporal features while maintaining computational efficiency. This enables our model to make a balance between accuracy and resource consumption, making it ideal for edge-enabled devices that have only limited computational power.

III. Our Approach

A. Methodology Overview

Our proposed deep learning algorithm for the transportation mode detection consists of three main steps: data preprocessing, TRTCN model training, and transportation mode prediction. For the workflow, all datasets are first transformed into uniform matrices during the data preprocessing phase and

TABLE I
SUMMARY OF ADVANCED TECHNIQUES FOR TRANSPORTATION MODE DETECTION

Technique	Key Features	Advantages
GPS-based methods [17], [18]	Bayesian Belief Networks, Random Forest	High accuracy in open environments
Wi-Fi/Bluetooth integration [19]	Combines Wi-Fi, Bluetooth, and GPS data	Battery efficient, accurate
Accelerometer-based methods [20], [21]	Hierarchical classifiers for sensor data	Lower power usage, high accuracy
Random Forest on sensor data [22]	Random Forest algorithm with smartphone sensors	Very high accuracy (99.4%)
Semi-supervised deep learning [31]	Deep Convolutional Autoencoder (SECA)	Handles unlabeled data, reduces handcrafted features
Hybrid deep learning models [15], [29]	Combines CNN, LSTM with traditional machine learning	Captures both spatial and temporal features
Wi-Fi and sound-based models [10], [30]	Uses Wi-Fi and sound data for detection	Enhanced accuracy in non-traditional environments
Temporal Convolutional Networks (TCN) [32]	Causal and feed-forward architecture	Captures long-range dependencies, real-time predictions

then fed into the input layer. In the model training step, the preprocessed data is used as training data, and test data is fed into the TRTCN model to optimize the trainable parameters in the TRTCN model. Consideration is given to the identification of barometric pressure and three other inertial sensors: linear acceleration, gyroscope, and magnetometer. Due to the undetermined posture and orientation of the smartphone, the X, Y, and Z axes of each sensor will be combined.

To capture intricate temporal dependencies and enable real-time predictions, the model employs multiple ResNet blocks for each sensor channel immediately after the data preprocessing stage. These ResNet blocks are designed to extract hierarchical features while maintaining the computational efficiency. The outputs from selected groups of ResNet blocks are then concatenated and passed through the Temporal Convolutional Networks (TCN) [33], which provide causal characteristics and multi-scale feature extraction capabilities.

To further refine these features and enlarge the receptive field, a simplified ResNet block is applied post-TCN, followed by another TCN layer. This architecture not only allows for the direct transmission of underlying features across layers but also enhances the model's gradient propagation capability.

Then, the model incorporates a Multi-Head Attention mechanism to capture long-range dependencies in the time-series data. Finally, the processed feature representations are passed through a Shortcut Multi-Layer Perceptron (ShortcutMLP) comprising multiple fully connected layers. The model then employs a Softmax function to produce the final transportation mode estimation. The definitions is shown in Table 1.

B. TRTCN Model

In this part, we describe our proposed TRTCN model in detail. The overall architecture is shown in Figure 1.

1) `Multimodal Input Layer`: The model receives preprocessed sensor data through the input layer–defined as Tensor $A_{n,d,k}$, where n is the number of the total samples, d is the length of the selected sliding window, and k is the total element number of all sensors. This work considers the representation of many axes, namely linear acceleration (X, Y, Z), gyroscope (X, Y, Z), magnetometer (X, Y, Z), and barometric pressure, denoted by the variable $k = 10$. The variable "d" denotes the length of the chosen sliding window, which consists of 500 samples. Each sample corresponds to a sampling period of 5 seconds, with a sampling frequency of 100Hz. Subsequently, the tensor $A_{n,d,10}$ undergoes a transformation, resulting in the creation of ten tensors denoted as $A_{n,l,1}$. These tensors are subsequently inputted into the residual layer.
2) `Residual Layer`: This layer is comprised of a total of ten residual units [34]. The residual layer, being the initial component of feature learning on the sensor input, significantly impacts the overall trend of feature learning inside the network. One of the benefits associated with the utilization of residual networks is the potential to enhance both the training efficiency and the accuracy of the model. Each individual in the group is provided with a tensor derived from the input layer: $A_{n,d,1}$. The operations of convolutional and max-pooling [35] are defined as follows. The chosen activation function has been demonstrated to significantly speed up the training process [36]. In this

TABLE II
DEFINITIONS OF VARIABLES AND SYMBOLS

Definition	Description
$A_{n,d,k}$	Input layer tensor
n	The number of the total samples
d	The length of the selected sliding window
k	The number of the total all sensors
$\hat{y}_0, \ldots, \hat{y}_T$	Temporal Convolutional Network output sequence
$x_0, \ldots, x_t$	Temporal Convolutional Network input sequence
o	The residual block formulate
Q	Multi-Self Attention function query
K	Multi-Self Attention function key
V	Multi-Self Attention function value
d_q	The projection of Multi-Self Attention function query
d_k	The projection of Multi-Self Attention function key
d_v	The projection of Multi-Self Attention function value
T	Knowledge distillation temperature parameter
q_i	Knowledge distillation soften probabilities

case, max-pooling is employed to down-sample the input, with both the kernel size and stride set to 3.

It is worth noting that our architectural integrates two specialized forms of residual layers: the Simple ResNet Layer and the Standard ResNet Layer, as illustrated in Figures 2 and 3.

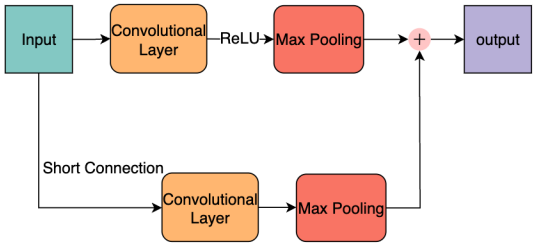

Fig. 2. The Simple Residual Layer

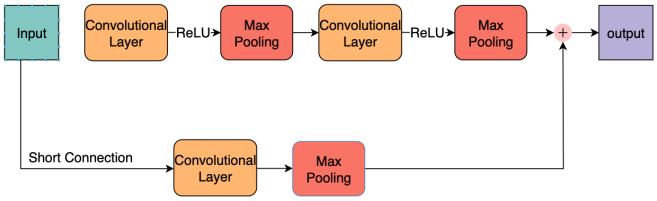

Fig. 3. The Residual Layer

3) `Temporal Convolutional Networks (TCN) Layer:` This TCN layer [33] is designed to address sequence modeling tasks while adhering to the principle of causality. Specifically, a TCN aims to predict an output sequence $\hat{y}_0, \ldots, \hat{y}_T$ based on an input sequence $x_0, \ldots, x_T$ in a way that each output $\hat{y}_t$ only depends on the observed inputs $x_0, \ldots, x_t$.

$$\hat{y}_0, \ldots, \hat{y}_T = f(x_0, \ldots, x_T) \tag{1}$$

The architecture of TCN combines a 1D Fully Convolutional Network (FCN) with causal convolutions. This ensures that the output at each time step is influenced only by the current and past inputs, thereby preserving causality.

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{2}$$

To extend the receptive field without increasing the network depth or filter size excessively, TCNs employ dilated convolutions. The dilation factor d is increased exponentially with the depth of the network.

In addition to these, TCNs incorporate residual connections [34] to stabilize the training of deep networks. The residual block is formulated as:

$$o = \text{Activation}(x + F(x)) \tag{3}$$

This architecture allows TCNs to be both deep and have a long effective history, making them suitable for a wide range of sequence modeling tasks.

4) `Multi-Head Attention Layer:` In this layer, we aim to enhance the attention mechanism by employing multiple attention heads. Instead of utilizing a single set of d_{model}-dimensional queries, keys, and values, we opt for projecting them into h different subspaces. These projections are of dimensions d_q, d_k, and d_v for queries, keys, and values, respectively.

$$\text{Multi}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

The projection matrices $W_i^Q \in R^{d_{\text{model}} \times d_q}$, $W_i^K \in R^{d_{\text{model}} \times d_k}$, $W_i^V \in R^{d_{\text{model}} \times d_v}$, and $W^O \in R^{hd_v \times d_{\text{model}}}$ are learned parameters. The architecture of Multi-Head Attention Layer is described in Figure 4.

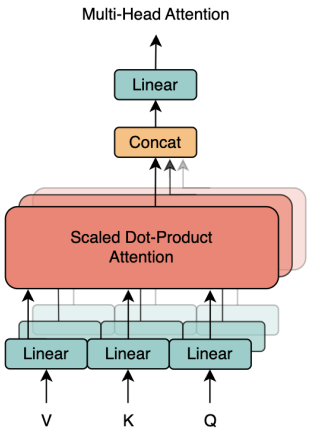

Fig. 4. Architecture of the Multi-Head Attention Layer

5) `MLP Layer:` This layer serves as the final stage for feature extraction and classification. It is composed of five fully connected layers, enhanced with shortcut connections for improved performance. To mitigate the risk of overfitting [37], a Dropout technique [38] is adopted in the layer. The output from the i^{th} fully connected layer can be mathematically represented as:

$$o_i = \text{Activation}(W \cdot x + b) \tag{6}$$

where W denotes the weight matrix of the hidden layer, and b is the bias term. The MLP Layer architecture is shown in Figure 5.

In the proposed TRTCN model, the dimensions for each hidden layer are set to 256, 512, 1024, 1024 and the number of classes, respectively. The activation functions employed are *ReLU* for the first four layers and *Softmax* for the final output layer. Then, Dropout

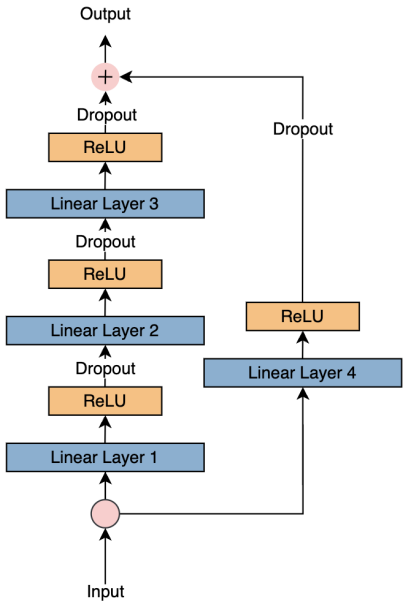

Fig. 5. Architecture of the Multi-Head Attention Layer

is applied to all hidden layers except for the last one, with a dropout rate of 0.2 in order to prevent overfitting.

Our TRTCN model also incorporates a shortcut connection between the input and the output of the third fully connected layer, which can be concatenated before passing through the fourth fully connected layer. This shortcut connection aims to achieve easier optimization and better performance.

C. Knowledge Distillation

Knowledge distillation [39] is a technique aimed at model compression, where a smaller model, referred to as the *student model*, is trained to mimic the behavior of a larger, pre-trained model, known as the *teacher model*. The core idea is to soften the probability distribution of the teacher model and convey more information than just the hard labels.

The softened probabilities are computed using the softmax function with a temperature parameter T, which is usually set to 1 for normal operation. The equation for the softened probabilities q_i is given by:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{7}$$

During the training phase of the student model, two loss functions are combined: one that minimizes the difference between the student's and teacher's softened logits, and another that minimizes the difference between the student's logits and the true labels. The combined loss function can be represented as below:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{soft}} + (1-\alpha) \cdot \mathcal{L}_{\text{hard}} \tag{8}$$

Here, $\mathcal{L}_{\text{soft}}$ is the Kullback-Leibler divergence between the softened logits of the student and teacher models, and $\mathcal{L}_{\text{hard}}$ is the cross-entropy loss between the student's logits and the true labels. α is a hyperparameter that controls the trade-off between the two objectives.

The gradients of the loss with respect to the logits z_i of the student model can be approximated as:

$$\frac{\partial \mathcal{L}}{\partial z_i} \approx \frac{1}{T^2}(z_i - v_i) \tag{9}$$

where v_i indicate the logits from the teacher model. This approximation holds true when the temperature T is high compared to the magnitudes of the logits.

This method offers a balance between paying attention to logits that are significantly more negative than the average and those that are around the average, providing an effective way to transfer knowledge from the teacher model to the student model. Within our proposed, the student model is a modified CNN specifically designed to imitate the instructor model.

The architectural design consists of a series of convolutional layers, followed by max-pooling layers, and culminating in a fully connected layer that is responsible for predicting the class. The primary purpose of the convolutional layers is to capture and extract feature representations from the input data, whereas the max-pooling layers serve to minimize the dimensionality of the feature maps. The completely linked layer produces logits that correspond to eight distinct classifications.

Throughout the forward pass, the input tensor experiences a sequence of operations, such as activation functions and dimensionality reduction, ultimately resulting in the logits generated by the fully connected layer.

For the distillation process, we employ a custom loss function that combines the Kullback-Leibler divergence between the softened logits of the student and teacher models, and the cross-entropy loss between the student's logits and the true labels. The equation for the distillation loss is given by:

$$\begin{aligned}\text{Loss} = \text{KLD}\left(\frac{\log(\text{SM}(y/T))}{\text{SM}(\text{TS}/T)}\right) \\ \times (T^2 \times 2.0 \times \alpha) + \text{CE}(y, \text{L}) \times (1-\alpha)\end{aligned} \tag{10}$$

IV. Experimental Evaluation

In this section, we design and conduct a set of experiments to evaluate the performance of our TRTCN model on public datasets such as the SHL 2018 dataset and SHL 2021 dataset. More specifically, a comprehensive introduction is provided about the two datasets with an in-depth explanation of the data processing procedure. We then examine the performance of TRTCN in comparison with the baseline.

A. Dataset

- `SHL 2018 dataset`: The dataset was obtained with a period of seven months in 2017 by a group of three volunteers located in the United Kingdom. There are eight different transportation modes that are categorized and identified during their daily traffic transfers. Every sample in the dataset includes measurements of ambient light, temperature, GPS location, Wi-Fi signal strength, and motion data. These samples were obtained based

on the Huawei Mate 9 cellphones that were positioned in various locations, e.g., within a bag, held in hand, secured to the chest, or placed in a pocket.

The motivation for selecting the SHL2018 dataset lies in its comprehensive multi-modal sensor data and its diverse transportation mode categories, which align well with the goals of our research. This dataset allows us to evaluate the performance of our TRTCN model in detecting different transportation modes under various real-world scenarios. The dataset also provides a good balance between data richness and practical applicability, enabling us to focus mainly on low power consumption sensors. We evaluate the TRTCN model by utilizing only lightweight sensor data, including accelerometer, gyroscope, magnetometer and baroceptor. In light of prioritizing the practical applications, the evaluation performance is limited to the sensor data collected by hand and in one's pocket. A total of 272 hours of sensor data were chosen for the purpose of training and testing the TRTCN model. After preprocessing the dataset, we divided it into the training part (80%) and the test part (20%). The data were collected by a single volunteer over a period of four months. The sampling rate for all the sensor data was 100Hz. In order to capitalize on the temporal relationships presented in our trials, we reorganized the SHL data in chronological order.

- `SHL 2021 dataset`: The dataset comprises 750 hours of multi-modal sensor data for studying human locomotion across eight activities: Car (88h), Bus (107h), Train (115h), Subway (89h), Walk (127h), Run (21h), Bike (79h), and Still (127h). With each participant carrying four smartphones, the effective data volume amounts to 3000 hours.

  The motivation for including the SHL2021 dataset is its larger size and more extensive set of sensor placements, which provide a more challenging benchmark for evaluating our model's scalability and generalization capabilities. This dataset allows us to test the robustness of our model across a wider range of transportation modes and human locomotion activities. By focusing specifically on the torso data from User 2, we maintain consistency in sensor placement and data quality while ensuring that the model's performance is tested on realistic and varied locomotion scenarios. We divided the dataset into training and testing sets, allocating 80% for training and 20% for testing purposes.

B. Data Preprocessing

To ensure the quality and reliability of the data fed into our model, we employ a series of preprocessing steps. These steps aim to clean, normalize, and segment the data, thereby enhancing the model's training and prediction accuracy. Below are the specific methods used:

1) `Column Selection`: Initially, we filter the raw dataset to only include the columns that are relevant to our study. This reduces the dimensionality of the data and focuses the model's attention on the most pertinent features.
2) `Dirty Data Removing`: Given the large volume of the dataset, it is crucial to remove any samples with incomplete or missing values. This step ensures that only complete and accurate data points are used for model training.
3) `Normalization`: To standardize the range of the sensor data, we apply Z-Score normalization to each element of the sensor vector data. The formula for Z-Score normalization [40] is given by:

$$x' = \frac{x - \mu}{\sigma} \tag{11}$$

where μ is the average and σ is the standard deviation of each element in the sensor data.
4) `Sliding Window-Based Segmentation`: The choice of sequence length for sensor data is crucial for accurate pattern recognition. Too short a sequence length can lead to recognition errors, while too long a sequence can result in computational overhead and prediction delays. To balance accuracy and computational efficiency, we use a fixed-length sliding window to segment the sensor data into short-term sequences. Specifically, we set the window length to 500 based on empirical experiments.

The objective of deploying these preprocessing procedures is to generate a dataset that is clean, normalized, and segmented. This dataset will facilitate the training and prediction of models with accuracy and efficiency.

C. Baselines and Metrics

To provide a comprehensive evaluation of our proposed model, we compare it with several baseline algorithms. These include classic machine learning algorithms, deep learning algorithms, and state-of-the-art methods in transportation mode recognition. Specifically, the baselines are MLP, CNN [7], RNN [8], LSTM [14], DeepConvLSTM [15], Decision Tree, TCMH [16] and T2Trans [9]. In particular, MLP is part of our proposed model and serves as a baseline for comparison. CNN is also part of our proposed model and serves as the student model in the knowledge distillation process. Both T2Trans and TCMH employ temporal convolutional networks for the purpose of transportation mode detection. T2Trans is a suggested method set to be introduced in 2022, while TCMH is scheduled for release in 2023. The detailed parameters of all baselines are listed in Table II.

To assess the performance of our proposed model and the baseline algorithms, we employ four evaluation metrics, such as model accuracy, precision, recall, and F1 Score. The F1 Score is defined as follow:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

By using these metrics, we aim to provide a thorough evaluation of the model's ability to accurately recognize different transportation modes.

TABLE III
THE DETAILED PARAMETERS OF BASELINES

Name	Architecture
MLP	FC (128)-FC (256)-FC (512)-FC (1024)-Softmax
CNN	Each Element [C(128)-P(2)-C(256)-P(2)]-C(32)-P(2)-MLP-Softmax
LSTM	LSTM(128)-MLP-Softmax
DeepConvLSTM	C(64)-C(64)-C(64)-C(64)-LSTM(128)-Softmax
DT	min. num. of instances per leaf: 2
T2Trans	input-TCN(16)-TCN(64)-C(64)-MLP(128, 256, 512, 1024)-Softmax
TCMH	input-TCN(16)-MHA(5)-CNN(64)

TABLE IV
THE CONFIGURATION OF PC

Name	Detail
CPU	Intel(R) Xeon(R) CPU @ 2.20GHz
Memory	12.67GB
GPU	Tesla T4-PCIE-15GB
Operating System	Ubuntu 22.04.2 LTS
Python Environment	3.10.12
Development Framework	Pytorch

D. Experimental Settings

In this work, the TRTCN model and the CNN student model were trained utilizing the Pytorch deep learning framework, applied to the preprocessed SHL 2018 dataset. Here, the optimization process adopted the Adam optimizer [41], accompanied by a cross-entropy loss function.

The learning rate was set to 0.001, beta1=0.9, beta2=0.999. The model was trained for 100 epochs with a batch size of 512 samples. For the knowledge distillation process in the CNN student model, the architecture employs a simple yet effective approach, integrating two convolutional layers with max-pooling and a fully connected layer, a temperature of 8 and an alpha of 0.3 were utilized. To mitigate the risk of overfitting, the sequence of data feeding was randomized. The experimental equipment has GPU support. The specific hardware configuration of a node is delineated in Table III.

E. Accuracy Results

The experimental results on the SHL 2018 dataset and SHL 2021 dataset were visually presented in Figure 6, Figure 7, Table IV, and Table V, respectively.

It is found that our proposed TRTCN model can outperform all baseline methods. The experiments on the SHL 2018 dataset and SHL 2021 dataset have reported an accuracy rate of 88.32% and 96.04%, respectively, for the TRTCN-CNN model. With the exception of Decision Tree (DT) model, all baseline models on the SHL 2018 dataset exhibited an accuracy rate beyond 60%. Similarly, on the SHL 2021 dataset, all baseline models achieved an accuracy rate surpassing 80%. Nevertheless, the existing baselines have little capability in accurately discerning patterns between trains and subways. This deficiency may arise from inadequate utilization of feature pressure, hence hindering the baselines' ability to acquire advanced features or temporal relationships.

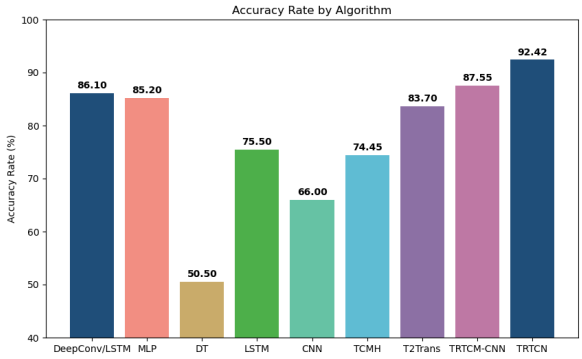

Fig. 6. The Transportation Recognition Accuracy Using Different Classification Algorithmns On The SHL 2018 dataset

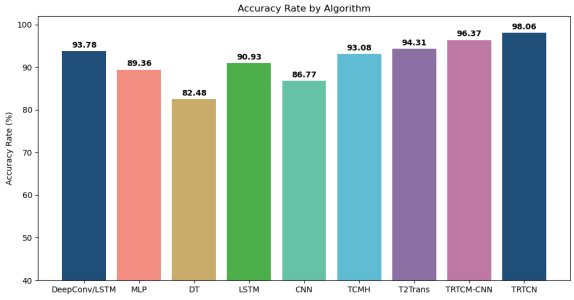

Fig. 7. The Transportation Recognition Accuracy Using Different Classification Algorithmns On The SHL 2021 dataset

Furthermore, the metrics of precision, recall, and F1 score of the TRTCN-CNN model are presented in Table VI and Table VII. Figure 8 and Figure 9 present the confusion matrix of the TRTCN-CNN model applied to the SHL 2018 dataset and the SHL 2021 dataset, respectively. In Figure 6, the DeepConvLSTM exhibits the second-highest discriminative performance. Similarly, in Figure 7, the T2Trans model also ranks second in terms of discriminative capability, whereas the DT model demonstrates the least discriminative power across both figures. Notably, Figures 8 and 9 highlight a challenge in distinguishing between the train and subway modes, indicating a potential area for enhancement in subsequent research endeavors.

F. Ablation Studies

Figure 10 illustrates the effectiveness of incorporating Simple Residual Layer and Multi-Head Attention in our student model with the SHL 2018 dataset. It is evident that the model with both Simple Residual Layer and Multi-Head Attention achieves the highest accuracy of 88.46%, demonstrating the synergistic benefits of combining these two techniques. The model with Simple Residual Layer alone also shows a notable

TABLE V
THE PRECISIONS OF DIFFERENT TRANSPORTATION MODE RECOGNITION USING DIFFERENT ALGORITHMS ON THE SHL 2018 DATASET

	DCL	**MLP**	**DT**	**LSTM**	**CNN**	**TCMH**	**T2Trans**	**TRTCN-CNN**	**TRTCN**
Still	86.42%	86.21%	53.57%	76.41%	58.25%	77.64%	84.23%	88.46%	90.51%
Walk	90.12%	88.20%	55.11%	79.28%	85.24%	89.56%	93.64%	92.27%	94.43%
Run	98.02%	96.93%	68.68%	90.41%	97.35%	96.98%	98.55%	98.68%	99.25%
Bike	86.61%	86.70%	48.23%	73.06%	67.80%	90.21%	91.61%	88.67%	90.73%
Car	90.17%	91.13%	59.03%	81.39%	63.95%	79.73%	88.73%	91.38%	92.53%
Bus	79.48%	81.16%	36.12%	71.28%	55.13%	69.59%	74.21%	84.54%	86.12%
Train	75.32%	76.61%	43.88%	67.52%	44.93%	57.08%	72.98%	79.40%	81.51%
Subway	75.10%	73.63%	42.27%	64.48%	42.55%	58.78%	69.22%	77.30%	79.36%

Note: DCL is short for DeepConvLSTM. TRTCN represents the teacher model of the TRTCN-CNN after knowledge distillation. TRTCN-CNN represents the CNN model after knowledge distillation of the TRTCN model.

TABLE VI
THE PRECISIONS OF DIFFERENT TRANSPORTATION MODE RECOGNITION USING DIFFERENT ALGORITHMS ON THE SHL 2021 DATASET

	DCL	**MLP**	**DT**	**LSTM**	**CNN**	**TCMH**	**T2Trans**	**TRTCN-CNN**	**TRTCN**
Still	97.10%	96.15%	89.31%	94.38%	86.57%	92.48%	91.67%	96.74%	97.15%
Walk	95.83%	93.96%	85.47%	94.53%	88.43%	97.01%	97.78%	96.69%	97.64%
Run	95.73%	94.92%	97.44%	90.60%	43.11%	98.33%	97.32%	95.73%	97.85%
Bike	98.17%	97.67%	95.85%	92.78%	95.39%	97.38%	96.97%	99.47%	99.46%
Car	94.38%	93.72%	81.69%	91.48%	91.96%	94.38%	92.87%	97.49%	98.33%
Bus	92.04%	89.90%	70.63%	86.53%	80.31%	91.72%	90.33%	93.33%	95.73%
Train	90.50%	89.22%	81.71%	89.93%	89.43%	91.19%	90.77%	94.74%	96.25%
Subway	90.58%	86.64%	77.94%	88.58%	86.87%	89.39%	91.31%	95.35%	96.18%

Note: DCL is short for DeepConvLSTM. TRTCN represents the teacher model of the TRTCN-CNN after knowledge distillation. TRTCN-CNN represents the CNN model after knowledge distillation of the TRTCN model.

TABLE VII
PRECISIONS,RECALLS AND F1-SCORES OF THE TRTCN-CNN MODEL ON THE SHL 2018 DATASET

	Precision	Recall	F1-Score
Still	88.96%	87.96%	88.46%
Walk	92.17%	92.37%	92.27%
Run	99.92%	97.47%	98.68%
Bike	89.01%	88.33%	88.67%
Car	89.47%	93.38%	91.38%
Bus	83.45%	85.65%	84.54%
Train	84.21%	75.10%	79.40%
Subway	75.34%	79.36%	77.30%

TABLE VIII
PRECISIONS,RECALLS AND F1-SCORES OF THE TRTCN-CNN MODEL ON THE SHL 2021 DATASET

	Precision	Recall	F1-Score
Still	95.70%	97.80%	96.74%
Walk	97.33%	96.05%	96.69%
Run	98.25%	93.33%	95.73%
Bike	100%	98.95%	99.47%
Car	95.80%	99.23%	97.49%
Bus	94.95%	91.77%	93.33%
Train	94.33%	95.14%	94.74%
Subway	95.73%	94.97%	95.35%

improvement, reaching an accuracy rate of 84.2%, meanwhile the model with the Multi-Head Attention alone attains an accuracy rate of 83.75%. The baseline model, without any of these additions, lags behind with an accuracy rate of 82.8%.

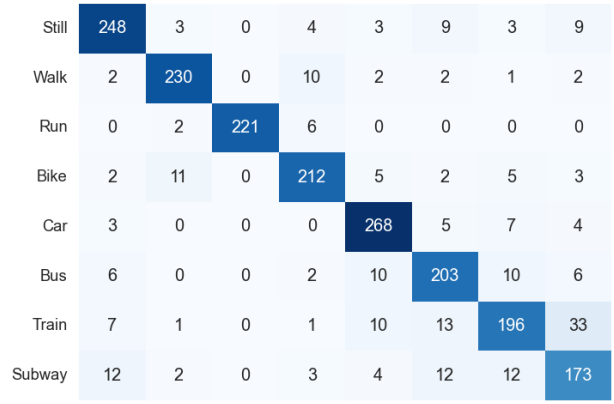

Fig. 8. The Confusion Matrix Of The TRTCN-CNN On The SHL 2018 dataset

These results clearly validate the efficacy of using both Simple Residual Layer and Multi-Head Attention in enhancing the model's performance.

The algorithm was initially evaluated using various architectures for feature extraction, and it was observed that Residual Networks (ResNet) and its simpler variant–Simple ResNet, provided significant improvements. Motivated by these empirical results / observations, we incorporated ResNet blocks, which are renowned for their ability to capture intricate

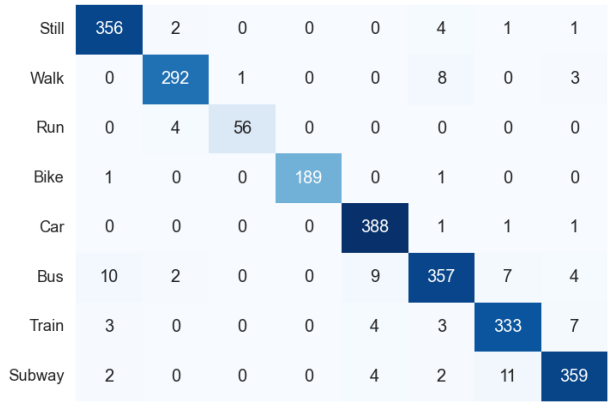

Fig. 9. The Confusion Matrix Of The TRTCN-CNN On The SHL 2021 dataset

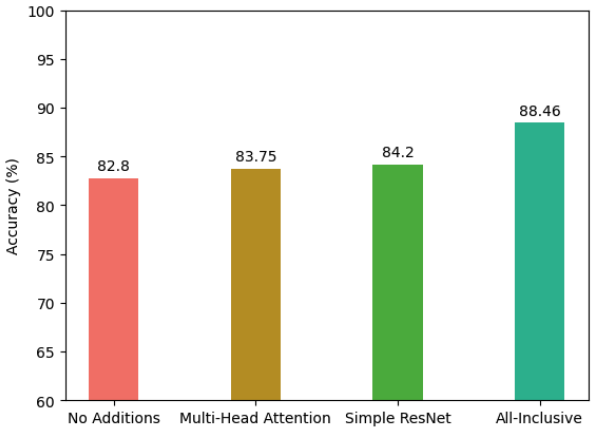

Fig. 10. Comparison of Different Additions

features through deep architectures. The residual connections in these blocks mitigate the vanishing gradient problem, enabling the model to effectively learn both low-level and high-level features. This is particularly beneficial for time-series data, where capturing both granular and abstract features is crucial for accurate categorization. On the other hand, the Simple ResNet blocks are designed to be computationally less demanding while retaining essential characteristics, and offering a scalable and efficient model.

After observing the efficacy of Multi-Head Self-Attention in preliminary experiments, we integrated it into the architectural design to further investigate its benefits. Unlike traditional attention mechanisms that uniformly weigh all elements in a sequence, the multi-head variant allows the model to focus selectively on different segments. This is particularly advantageous for time-series classification, as it enables the model to consider varying levels of importance across different time frames. The Multi-Head Self-Attention mechanism effectively distributes its focus, capturing a wide array of features and thereby deepening the model's understanding of temporal relationships.

G. Computational Complexity and Efficiency

To provide meaningful insights into the computational efficiency of our proposed TRTCN-CNN model, we quantify the key metrics such as *training accuracy*, *inference time*, and *parameter size*. These factors are critical for balancing model complexity with performance, especially for real-world, resource-constrained scenarios.

Our TRTCN-CNN model achieves an impressive F1-Score of 96.04%, surpassing the performance of several advanced algorithms in the literature such as DeepConvLSTM, MLP, LSTM, CNN, TCMH, and T2Trans. A notable feature of our proposed model is its efficient use of parameters—specifically, it employs only 486,280 parameters, which is significantly lower than models like DCL, which utilizes 2,850,184 parameters for an F1-Score of 93.78%. This reduction of approximately six times fewer parameters not only reduces memory consumption but also decreases computational demands, enhancing the model's applicability on edge devices.

Regarding inference time, the TRTCN-CNN model demonstrates remarkable efficiency, requiring only 34.25 seconds for inference, which is significantly faster than models like TCMH that needs 459.1 seconds. This substantial reduction in computational time positions the TRTCN-CNN model as highly suitable for real-time applications, a crucial requirement for transportation mode detection in edge computing environments (with IoT devices).

As illustrated in Figure 11, the TRTCN-CNN model strikes an optimal balance between parameter efficiency and performance. Despite utilizing a reduced number of parameters, it delivers superior F1-Score performance compared to other relevant models, making it particularly well-suited for deployment in environments with limited computational resources and memory, such as mobile or IoT devices.

V. Limitation

Despite the promising results achieved by the proposed TRTCN-CNN model, there are several limitations that should be acknowledged. First, the model relies heavily on specific sensor data, particularly accelerometer, gyroscope, and magnetometer readings. This dependence on sensor-specific data may limit the generalizability of the model to other transportation mode detection scenarios where different types of sensors or additional environmental data (e.g., GPS or Wi-Fi) may be needed. Additionally, the evaluation was conducted using datasets that feature data collected under relatively controlled conditions, which may not fully capture the variability and noise found in more diverse, real-world environments.

Another potential limitation is the computational complexity of the TRTCN model, especially when applied to resource-constrained devices. While we have optimized the model for edge devices, the implementation may still be challenging for devices with minimal computational power. Future research could focus on further model compression techniques, such as quantization or pruning, to enhance its efficiency. Moreover, expanding the dataset to include more users [42], varied environments, and additional transportation modes could further improve the model's robustness and adaptability to real-world scenarios.

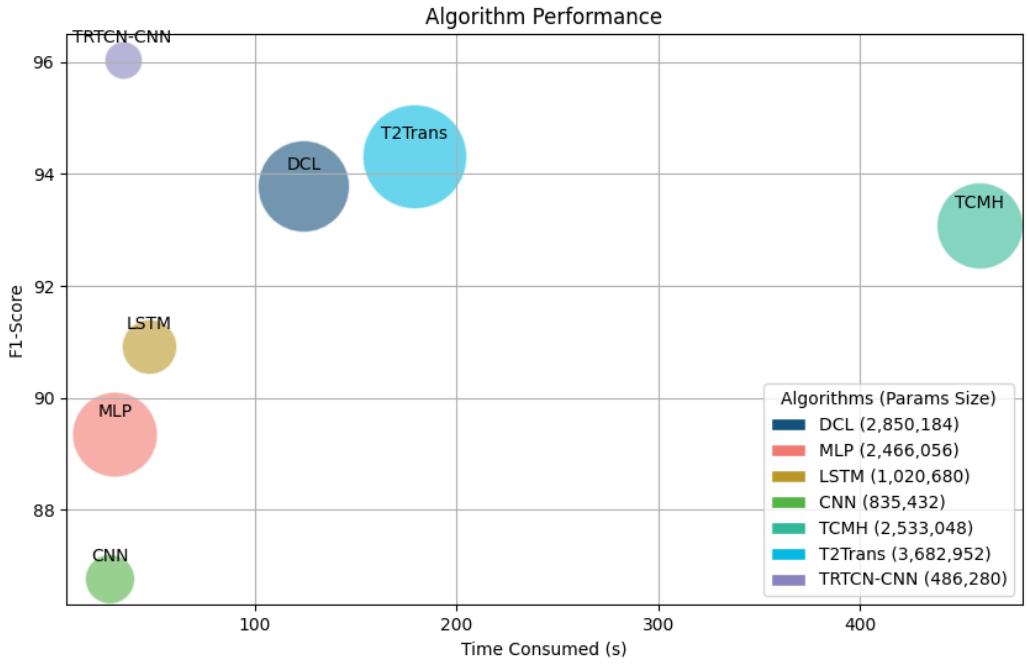

Fig. 11. Comparative Analysis of Algorithmic Performance: F1-Score vs. Time Consumption with Parameter Size Indication On The SHL 2021 dataset

VI. Conclusion

In this work, we focused on transportation mode detection and proposed an enhanced Temporal Convolutional Networks (TCNs) model, called TRTCN-CNN via edge intelligence and consumer smartphone sensors. It adeptly captures the long-range temporal dependencies in multi-sensor data harvested from smartphones. The architecture leverages causal dilated convolutions, thereby facilitating the ability of parallel computation across all temporal steps and substantially expediting the training process. The model demonstrated exemplary performance, achieving an average precision rate of 88.32% on the SHL-Challenge Dataset (2018) and an impressive 96.04% on the SHL-Dataset (2021). Also, our approach incorporated residual networks and the multi-Head attention mechanisms to further enhance its feature extraction and decision-making capabilities. Furthermore, we employed knowledge distillation techniques to train a much more parameter-efficient model that not only accelerates the training process but also improves the accuracy, making it highly suitable for deployment on edge-enabled consumer devices. Our results indicated that our proposed TRTCN-CNN outperformed other baseline and state-of-the-art schemes.

References

[1] A. Efthymiou, E. N. Barmpounakis, D. Efthymiou, and E. I. Vlahogianni, "Transportation mode detection from low-power smartphone sensors using tree-based ensembles," *Journal of Big Data Analytics in Transportation*, vol. 1, pp. 57–69, 2019.

[2] B. Lampe and W. Meng, "A survey of deep learning-based intrusion detection in automotive applications," *Expert Syst. Appl.*, vol. 221, p. 119771, 2023.

[3] B. Alotaibi, "Transportation mode detection by embedded sensors based on ensemble learning," *IEEE Access*, vol. 8, pp. 145 552–145 563, 2020.

[4] C. Carpineti, V. Lomonaco, L. Bedogni, M. Di Felice, and L. Bononi, "Custom dual transportation mode detection by smartphone devices exploiting sensor diversity," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2018, pp. 367–372.

[5] P. Vinayaraj and K. Mede, "Multi-branch deep learning based transport mode detection using weakly supervised labels," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, pp. 525–530, 2022.

[6] F. Yu, L. Wang, Q. Jiang, Q. Yan, and S. Qiao, "Self-attention-based short-term load forecasting considering demand-side management," *Energies*, vol. 15, no. 12, p. 4198, 2022.

[7] S. Dabiri and K. Heaslip, "Inferring transportation modes from gps trajectories using a convolutional neural network," *Transportation research part C: emerging technologies*, vol. 86, pp. 360–371, 2018.

[8] H. Liu and I. Lee, "End-to-end trajectory transportation mode classification using bi-lstm recurrent neural network," in *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. IEEE, 2017, pp. 1–5.

[9] P. Wang and Y. Jiang, "Transportation mode detection using temporal convolutional networks based on sensors integrated into smartphones," *Sensors*, vol. 22, no. 17, p. 6712, 2022.

[10] A. Kalatian and B. Farooq, "A semi-supervised deep residual network for mode detection in wi-fi signals," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 2, pp. 167–180, 2020.

[11] W. Li, T. Gleerup, J. Tan, and Y. Wang, "A security enhanced android unlock scheme based on pinch-to-zoom for smart devices," *IEEE Trans. Consumer Electron.*, vol. 70, no. 1, pp. 3985–3993, 2024.

[12] H. Gjoreski, M. Ciliberto, L. Wang, F. J. O. Morales, S. Mekki, S. Valentin, and D. Roggen, "The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices," *IEEE Access*, vol. 6, pp. 42 592–42 604, 2018.

[13] S. D. Team, "Shl dataset," 2021, accessed: 2023-03-12. [Online]. Available: http://www.shl-dataset.org/download/

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[16] S. Cheng and Y. Liu, "Research on transportation mode recognition based on multi-head attention temporal convolutional network," *Sensors*, vol. 23, no. 7, p. 3585, 2023.

[17] T. Feng and H. J. Timmermans, "Transportation mode recognition using gps and accelerometer data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 118–130, 2013.

[18] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and gis information," in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2011, pp. 54–63.

[19] A. Bjerre-Nielsen, K. Minor, P. Sapieżyński, S. Lehmann, and D. D. Lassen, "Inferring transportation mode from smartphone sensors: Evaluating the potential of wi-fi and bluetooth," *PloS one*, vol. 15, no. 7, p. e0234003, 2020.

[20] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM conference on embedded networked sensor systems*, 2013, pp. 1–14.

[21] H. I. Ashqar, M. H. Almannaa, M. Elhenawy, H. A. Rakha, and L. House, "Smartphone transportation mode recognition using a hierarchical machine learning classifier and pooled features from time and frequency domains," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 244–252, 2018.

[22] E. A. Sağbaş and S. Ballı, "Transportation mode detection by using smartphone sensors and machine learning," 2016.

[23] T. H. Vu, L. Dung, and J.-C. Wang, "Transportation mode detection on mobile devices using recurrent nets," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 392–396.

[24] K. C. Koo, K. S. Lee, S. Kim, C. Min, G. R. Min, Y. H. Lee, W. K. Han, K. H. Rha, S. J. Hong, S. C. Yang, *et al.*, "Long short-term memory artificial neural network model for prediction of prostate cancer survival outcomes according to initial treatment strategy: development of an online decision-making support system," *World journal of urology*, vol. 38, pp. 2469–2476, 2020.

[25] A. Alferaidi, K. Yadav, Y. Alharbi, N. Razmjooy, W. Viriyasitavat, K. Gulati, S. Kautish, G. Dhiman, *et al.*, "Distributed deep cnn-lstm model for intrusion detection method in iot-based vehicles," *Mathematical Problems in Engineering*, vol. 2022, 2022.

[26] J. Li, C. Ma, Y. Han, H. Mu, and L. Jiang, "Enhanced scnn-based hybrid spatial-temporal lane detection model for intelligent transportation systems," *IEEE Access*, vol. 12, pp. 40 075–40 091, 2024.

[27] I. Cardoso-Pereira, J. B. B. Neto, A. C. Viana, A. A. F. Loureiro, and H. S. Ramos, "Popayi: Muscling ordinal patterns for low-complex and usability-aware transportation mode detection," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17 170–17 183, 2024.

[28] M. G. R. Alam, M. Haque, M. R. Hassan, S. Huda, M. M. Hassan, F. L. Strickland, and S. A. AlQahtani, "Feature cloning and feature fusion based transportation mode detection using convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4671–4681, 2023.

[29] Z. Chen, L. Zhang, Z. Cao, and J. Guo, "Distilling the knowledge from handcrafted features for human activity recognition," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4334–4342, 2018.

[30] L. Wang and D. Roggen, "Sound-based transportation mode recognition with smartphones," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 930–934.

[31] S. Dabiri, C.-T. Lu, K. Heaslip, and C. K. Reddy, "Semi-supervised deep learning approach for transportation mode identification using gps trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 1010–1023, 2019.

[32] P. Hewage, A. Behera, M. Trovati, E. Pereira, M. Ghahremani, F. Palmieri, and Y. Liu, "Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station," *Soft Computing*, vol. 24, pp. 16 453–16 482, 2020.

[33] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *2011 IEEE international conference on signal and image processing applications (ICSIPA)*. IEEE, 2011, pp. 342–347.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[37] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of chemical information and computer sciences*, vol. 35, no. 5, pp. 826–833, 1995.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[40] C. Saranya and G. Manikandan, "A study on normalization techniques for privacy preserving data mining," *International Journal of Engineering and Technology (IJET)*, vol. 5, no. 3, pp. 2701–2704, 2013.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] B. Lampe and W. Meng, "can-train-and-test: A curated CAN dataset for automotive intrusion detection," *Comput. Secur.*, vol. 140, p. 103777, 2024.