POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Online Knowledge Distillation-Based Neural Network Optimization Strategies: Application to Long-Range Capacitive Plate Indoor Positioning

Supervisors

Prof. Mihai T. LAZARESCU

Candidate

Feicheng ZHANG

March 2024

Summary

As the adoption of smart home technologies grows, so too does the need for accurate and cost-effective indoor positioning solutions. High-precision systems traditionally rely on beacons [1]. However, beacon-less alternatives [2], like visual positioning through high-definition cameras, present challenges including inconvenience and privacy concerns in residential settings. An emerging, viable solution is the use of capacitive sensors [3]. They can sense [4, 5], identify [6, 7], and localize [8, 9, 10, 11] persons indoor, but their sensitivity decreases steeply with the distance. In the case of long-range sensing, the technology is notably vulnerable to various environmental influences, including electromagnetic and electrostatic interference, humidity, and temperature fluctuations [12]. Regression neural networks used to analyze variations in data from capacitive plates mounted on walls have been shown to be both feasible and accurate for determining a person's location [13, 14]. Yet, the limited computational capacity of smart home embedded systems necessitates efficient model compression without compromising performance.

This thesis explores knowledge distillation techniques, with a focus on online distillation, and offers a comparative analysis with the outcomes of offline distillation. In the context of advancing indoor positioning technologies, particularly within smart home environments, is made a comprehensive investigation into optimizing neural network strategies through online knowledge distillation. The focus is on leveraging long-range capacitive plate sensors for high-precision indoor positioning, overcoming the limitations posed by the computational constraints of smart home embedded systems. Through an innovative application of online distillation techniques, the research explores the dynamic process of mutual learning between a student model and a teacher model, revealing significant improvements in model performance and efficiency.

Key Contributions and Findings

1. **Paradigm Shift to Online Knowledge Distillation**: Central to this research is the exploration of online knowledge distillation, a method that departs from traditional offline distillation by allowing concurrent learning and

knowledge exchange between the teacher and student models. This underscores the dynamic and interactive nature of learning processes in neural networks, highlighting the potential for real-time performance enhancements and the acquisition of more nuanced “domain-specific knowledge” that static methods may overlook.

2. **Innovative Optimization Strategies**:
    - **Adaptive Weights Mechanism**: The thesis introduces a novel adaptive weights mechanism, crucial for balancing the influence of distilled knowledge and inherent training data on the student model’s learning trajectory. This mechanism dynamically adjusts the weights of distilled knowledge in the loss function, optimizing the learning process based on the student model’s progress and ensuring a balanced and effective knowledge transfer.
    - **Pre-Distillation Training**: Another innovative strategy explored is pre-distillation training, where the teacher model undergoes preliminary training to stabilize and improve the quality of distilled knowledge. This approach mitigates potential misleading guidance from untrained models at the onset of training, enhancing the overall stability and effectiveness of the distillation process.
    - **Co-Training Using Kalman Filter**: The use of a Kalman filter to post-process the output of the teacher model represents a significant advance. Trained to model human walking dynamics and treating the teacher’s output as observational noise, this method integrates prior knowledge about human motion and enriches the distilled knowledge with information about smoothness and other characteristics of human indoor trajectories.

Key Results

Through rigorous experimental validation, the research demonstrates that the use of online distillation methods with a controller leads to smaller validation errors on the validation set for both types of models compared to traditional methods (for TCN and CPS models, the validation errors decreased by 14 % and 8 %, respectively).After further tuning the value of η[7.1], the validation error was reduced by 35.1 % and 20 %,for TCN and CPS modelsr espectively, compared to the offline method. The application of a Pre-training strategy further reduced the validation error on the validation set by 4.8 %. Ultimately, by applying domain knowledge using a Kalman filter, the optimized online distillation methods shown much improved generalization capability on two completely new test sets compared to offline distillation, with test errors reduced by 20 % and 9 % on the two new test sets, respectively. Finally, these results highlight the effectiveness of these optimization strategies in enhancing the generalization capabilities of the student model, which is a critical aspect for the practical application of indoor positioning systems.

Conclusion and Future Research Directions

This detailed exploration and the consequential findings of the thesis lay a robust foundation for future work in the field of knowledge distillation, particularly emphasizing the importance of dynamic learning processes and the integration of prior knowledge for improving neural network models in complex applications such as indoor positioning.

The potential of online knowledge distillation in enhancing neural network models for indoor positioning applications has been demonstrated. The successful integration of adaptive weights, pre-distillation training, and a Kalman filter suggests benefits from further exploration of incorporating more diverse and accurate domain knowledge and other optimization techniques to refine online distillation methods for most applications.

Acknowledgements

I wish to convey my heartfelt appreciation and deepest gratitude to all those who have contributed significantly to the completion of my thesis. Their guidance, support, and encouragement have been vital in my academic journey.

I would like to express my sincere gratitude to Professor Mihai Teodor Lazarescu, my supervisor, for providing me with the opportunity to conduct this research under his guidance and within his research group. His insight, encouragement, and expertise have been invaluable to my studies and research.

I am deeply grateful to Giorgia Subbicini, my co-supervisor, for her unwavering guidance, boundless patience, and willingness to share her vast knowledge throughout the research process. Her insightful feedback and expertise were crucial in shaping both my research journey and my professional growth.

Table of Contents

List of Tables

List of Figures

Acronyms

ML
: Machine Learning

NN
: Neural Network

TCN
: Temporal Convolutional Neural Network

CPS
: Capsule Neural Network

MSE
: Mean Squared Error

LSTM
: Long Short-Term Memory Neural Network

AIL
: Attentive Imitation Loss

MDL
: Deep Mutual Learning

Chapter 1

Introduction

1.1 Background

As the adoption of smart home technologies grows, so too does the need for accurate and cost-effective indoor positioning solutions. High-precision systems traditionally rely on beacons; however, beacon-less alternatives, like visual positioning through high-definition cameras, present challenges including inconvenience and privacy concerns in residential settings. An emerging, viable solution is the use of long-range capacitive sensors. Leveraging regression neural networks to analyze variations in data from capacitive plates mounted on walls has been demonstrated to be both feasible and highly precise for determining a person's location. Yet, the limited computational capacity of smart home embedded systems necessitates efficient model compression without compromising performance.

1.2 Motivation

Knowledge distillation represents a pivotal model compression technique designed to transfer the insights of a large, complex model (termed the "teacher model") to a smaller, simpler model (referred to as the "student model"), thereby enhancing the performance of the latter while maintaining efficiency and a lightweight footprint. This methodology is particularly beneficial for applications requiring model deployment in resource-constrained settings, such as mobile devices and embedded systems. Typically conducted post the training of the teacher model, the process begins with the training of a large, complex model using extensive data to achieve high accuracy. Subsequently, the knowledge of the teacher model is extracted either by employing its output (e.g., the output of the softmax layer) as a target or by extracting certain features from the model's intermediate layers. Finally, this knowledge is used to train a smaller, simpler model (the student model). The training of the student model relies not only on the original label data but also on the outputs of the teacher model, enabling the student to learn subtle features and probability distributions from the teacher. This approach, referred to as offline distillation, has its effectiveness and mechanisms well articulated in the literature.

However, a recent study [15] introduces an intriguing concept of training two models of similar or differing scales simultaneously, employing a shared error as part of the loss function. This shared error is computed in a manner nearly identical to the distillation error used in offline distillation. Remarkably, in classic image recognition tasks, the co-learning approach has demonstrated a significant improvement in the accuracy of the smaller model compared to when it is trained independently. This could be considered a form of online distillation, a method that, although only experimentally validated in classification neural networks, has shown promise. Given the proven applicability of offline distillation to regression tasks, exploring the potential of this online approach warrants further investigation.

1.3 Overview

This thesis primarily revolves around the application of online distillation to specific regression tasks, such as how to address numerical instability in online distillation within regression tasks, how to configure the weight between distilled knowledge and the model's own loss, and how to apply some prior knowledge during the training process to enhance the model's performance.

The thesis is structured into several chapters as follows:

- Chapter 1 introduces the thesis and outlines the motivation behind the research.
- Chapter 2 covers general concepts of ML and Neural Networks
- Chapter 3 presents the working mechanisms of knowledge distillation, along with the necessary adjustments made to adapt it to our task, and the specific implementation of online distillation.
- Chapter 4 introduces our experimental environment and configuration.
- Chapter 5 presents our dataset generation framework and feature selection.
- Chapter 6 details the architecture of our models, including a exposition of their mechanisms and the rationale behind selection.
- Chapter 7 explores methods for improving the efficacy of online distillation and modifications to the training process.
- Chapter 8 presents the experimental results and evaluates the efficacy of proposed approachs.
- Chapter 9 concludes the thesis work and provides an outlook on future research directions.

Chapter 2

Machine Learning

2.1 Introduction

Machine Learning (ML) aims at crafting algorithms capable of autonomously identifying patterns within incoming data, bypassing the need for explicit coding [16]. This capability enables ML algorithms to address a broad spectrum of complex issues, including but not limited to, recognizing speech, processing natural language, and identifying images. The foundational structure of ML approaches generally consists of three essential elements: the input data, a learning algorithm, and a subsequently trained model. The learning algorithm's role is to educate the model using the input data, empowering it to render predictions or decisions concerning new data.

2.2 Neural Networks

Neural networks[17] are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

At its simplest, a neural network consists of layers of nodes, or "neurons," with each layer designed to perform specific types of transformations on its inputs. These transformations are determined by the strength of connections between neurons, which are adjusted during training. The process of adjusting these connections is known as "learning." The typical structure as show in the Figure2.1

Neural networks are particularly powerful in handling complex tasks like image and speech recognition, language translation, and playing complex games like Go or Chess. This is because they can automatically and adaptively learn spatial hierarchies of features from data, a task that is challenging with traditional machine learning techniques.

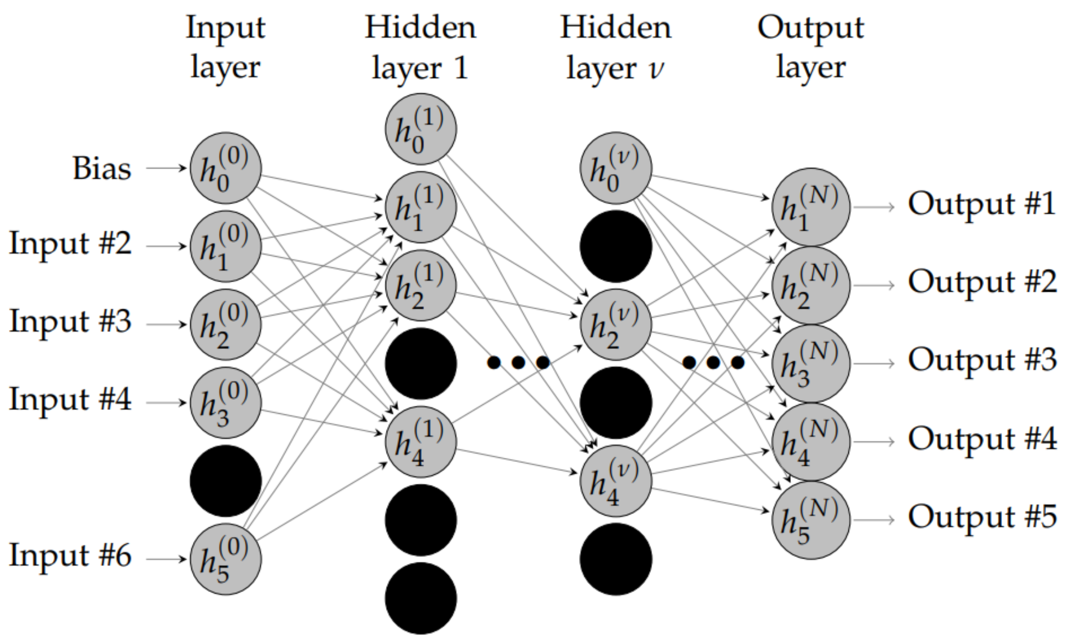


Figure 2.1: Typical Structure of NN [18]

The relationship between neural networks and machine learning is that neural networks provide a powerful tool for machine learning tasks. While machine learning is a broad field that encompasses various techniques for enabling machines to learn from data, neural networks specifically refer to a set of algorithms inspired by the structure and function of the brain's neural networks. They can learn and model complex relationships between inputs and outputs or patterns within data, making them a cornerstone of many modern artificial intelligence (AI) applications.

2.2.1 Common Activation Functions in the Neuron

Activation functions are a fundamental aspect of neural networks, enabling these models to capture complex and non-linear relationships within the data. By integrating non-linear properties into the network, activation functions allow neural networks to learn and perform tasks that are far beyond the capabilities of linear models. The choice of activation function can significantly affect the performance and convergence speed of a neural network. Below, we delve into some of the most pivotal activation functions in neural network design:

- **Sigmoid Function:** The sigmoid activation function, also known as the logistic function, is mathematically defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. It maps any input value to a range between 0 and 1, making it particularly useful for models where we need to interpret the output as a probability. Despite its historical popularity, especially in binary classification problems, its usage has declined due to issues such as vanishing gradients, where the function saturates at 0 or 1, leading to extremely small gradients during backpropagation and hence slow training.

- **Hyperbolic Tangent Function (tanh):** The tanh function scales the output of the sigmoid function to range between -1 and 1, defined as $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. This centring of the output range around zero allows for higher efficiency in learning, as it tends to make the mean of the activations closer to zero, and thus helps in speeding up convergence. Like the sigmoid, it can suffer from vanishing gradients for large positive or negative inputs.

- **Rectified Linear Unit (ReLU) [19]:** Defined as $f(x) = \max(0, x)$, ReLU has become exceptionally popular due to its simplicity and efficiency. It activates a neuron only if the input is positive and deactivates it otherwise. This piecewise linear function helps to alleviate the vanishing gradient problem, allowing models to learn faster and perform better. However, it is not without its issues; the "dying ReLU" problem occurs when inputs are negative, and neurons stop firing altogether, potentially causing portions of the network to become inactive during training.

- **Leaky Rectified Linear Unit (Leaky ReLU) [20]:** To mitigate the dying ReLU problem, Leaky ReLU introduces a small slope for negative values, thereby allowing a small, non-zero gradient when the unit is not active. Mathematically, it is defined as $f(x) = x$ if $x > 0$, otherwise αx where α is a small constant, such as 0.01. This adjustment ensures that all neurons have the opportunity to activate, maintaining a gradient flow through the network and improving the network's capacity to learn.

- **Softmax Function:** The softmax function is crucial for multi-class classification problems. It converts a vector of values into a probability distribution, where each value's exponentiated form is divided by the sum of exponentiated values of all elements in the vector. For input vector x_i, it is defined as $\frac{e^{x_i}}{\sum_j e^{x_j}}$. The softmax function ensures that the output values are in the range (0,1) and that they sum up to 1, making it possible to interpret the outputs as probabilities. The softmax is typically applied in the output layer of a classifier, providing a clear, probabilistic output across multiple classes.

Understanding the intricacies of these activation functions is crucial for designing effective neural network architectures. The choice of activation function can dramatically influence the learning dynamics and performance of the network, making it an essential consideration in the development of deep learning models.

2.3 Supervised Learning

Supervised learning, a cornerstone of machine learning, employs labeled datasets to train algorithms. In this paradigm, every training instance is paired with an output label, serving as a guide for the algorithm to learn the mapping between inputs and desired outputs. This method is instrumental in applications where the prediction of precise outcomes is crucial.

2.3.1 Dataset

The dataset comprises input features along with corresponding target labels. For instance, in a spam detection system, emails would be input features, and their labels would be “spam” or “not spam.”

2.3.2 Applications

- **Classification**: This involves categorizing input data into predefined classes. A quintessential example is a facial recognition system that identifies individuals within images.
- **Regression**: Here, the goal is to predict a continuous quantity. Predicting stock prices based on historical data exemplifies regression.

2.3.3 Challenges

A primary challenge in supervised learning is overfitting, where the model learns the noise in the training data instead of the actual signal. Techniques like cross-validation and regularization are employed to mitigate this issue.

2.4 Unsupervised Learning

Unsupervised learning explores data without preassigned labels, uncovering hidden patterns and structures. It’s pivotal in scenarios where the underlying distributions of data are unknown or when discovering intrinsic groupings within the data is the goal.

2.4.1 Dataset

The dataset consists solely of input data without any associated labels. An example could be a collection of customer shopping habits data, without any indication of market segments.

2.4.2 Applications

- **Clustering**: It aims to group a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Market segmentation, where customers are grouped based on purchasing behavior, is a classic example.

- **Dimensionality Reduction**: This process reduces the number of random variables under consideration, by obtaining a set of principal variables. Techniques like PCA (Principal Component Analysis) help in reducing the feature space, thus simplifying models without a significant loss of information.

- **Association Rule Learning**: It discovers interesting relations between variables in large databases. An iconic example is the "beer and diapers" story from market basket analysis, where purchases of diapers were found to be associated with purchases of beers.

2.4.3 Challenges

Evaluating the effectiveness of an unsupervised learning model can be challenging due to the absence of a clear benchmark or target. Moreover, determining the optimal number of clusters in clustering problems or the right number of dimensions to retain in dimensionality reduction requires heuristic approaches or domain expertise.

2.5 Comparison and Integration

While supervised learning models excel at predictive tasks with clear objective metrics for evaluation, unsupervised learning models thrive in exploratory data analysis, revealing underlying patterns and associations without predetermined labels. The choice between supervised and unsupervised learning depends on the nature of the problem at hand, the type of data available, and the specific goals of the analysis.

In practice, these two approaches are not mutually exclusive and can be combined in what's known as semi-supervised learning, where a small amount of labeled data guides the learning process in a larger unlabeled dataset. Another approach is reinforcement learning, where an agent learns to make decisions by performing actions and receiving feedback from the environment.

In this work, we frame our regression problem within the context of supervised learning.

2.6 Model Evaluation

Model evaluation plays a pivotal role in Machine Learning (ML) by enabling the assessment of a model's effectiveness and its capacity to generalize to data it hasn't encountered before. This section delves into two prevalent strategies for evaluating models: the single test method and cross-validation.

- The single test method is a straightforward strategy that involves dividing the dataset into three segments: training, validation, and testing sets, usually following a 70% (training), 10% (validation), and 20% (testing) split. The training set is used to educate the model, the validation set assists in fine-tuning hyperparameters and mitigating overfitting, and the testing set serves to gauge the model's efficacy. Although this method is straightforward and computationally light, its reliability might waver with small or skewed datasets.
- Cross-validation stands out as a more comprehensive method, overcoming the drawbacks of the single test method through its two primary formats: k-fold cross-validation applied to the entire dataset or restricted to the training data only.
  - In the first format, the dataset is partitioned into k equal parts. The model undergoes training and evaluation k times, with each iteration using a different part as the test set while the remaining parts form the training set. This cycle ensures every data point is used in both training and testing, yielding a thorough assessment of the model's capabilities. The results from each iteration are averaged to estimate the model's overall performance accurately.
  - The second format implements k-fold cross-validation on the training dataset(as show in the Figure 2.2) keeping a separate holdout set for the final evaluation. This method mirrors the first in its execution but confines the cross-validation to the training set, with the holdout set dedicated to assessing performance on new data. While this approach is less prevalent, it offers an extra layer of validation for specific scenarios, ensuring the model's performance is tested against completely unseen data.

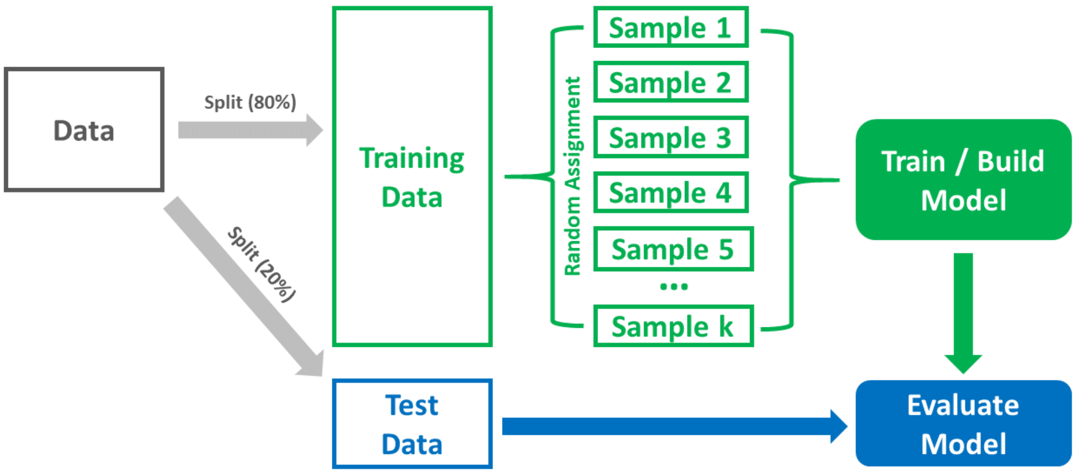


Figure 2.2: K-Fold cross-validation on the training set [21]

Chapter 3

Knowledge Distillation

3.1 Operation

Knowledge distillation is a technique used in machine learning to transfer knowledge from a larger, more complex model (teacher) to a smaller, more efficient one (student). This process aims to improve the performance of the student model while maintaining or reducing computational resources. There are several approaches to knowledge distillation, including generic response-based, feature-based, and relation-based knowledge distillation.

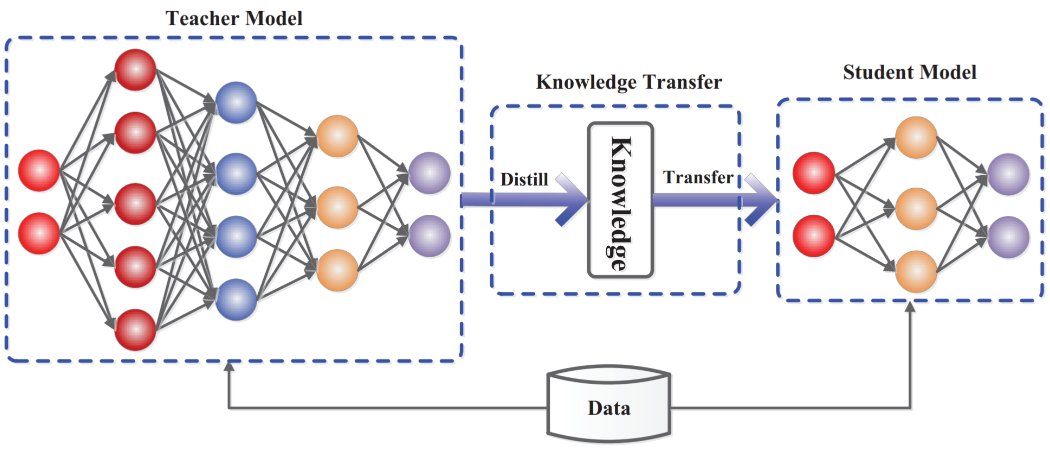


Figure 3.1: The generic teacher-student framework for knowledge distillation [22]

- Generic Response-Based Knowledge Distillation: This is the most straightforward approach, where the student model learns to mimic the output (or response) of the teacher model. The objective is to produce similar predictions, effectively learning the teacher's decision boundaries. This method often

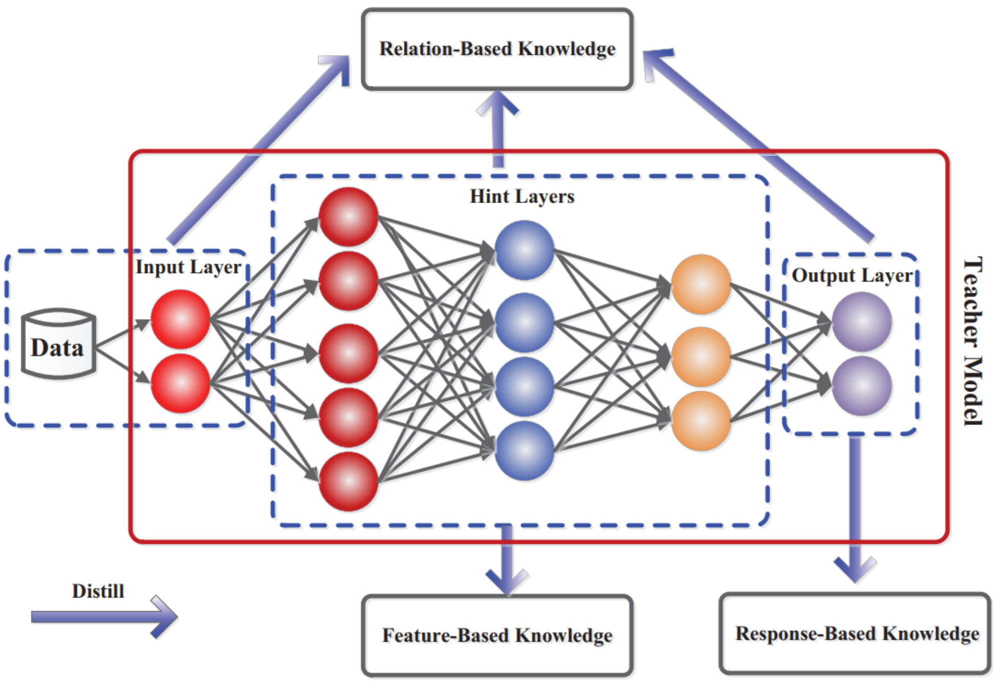


Figure 3.2: The schematic illustrations of sources of response-based knowledge, feature-based knowledge and relation-based [22]

involves softening the teacher's outputs with a temperature parameter to provide more information about the class probability distribution.

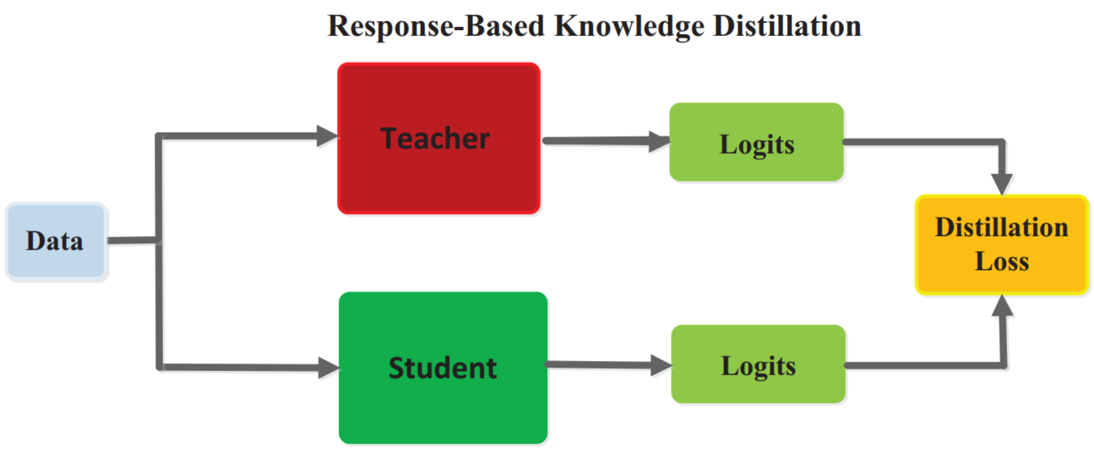


Figure 3.3: The generic response-based knowledge distillation [22]

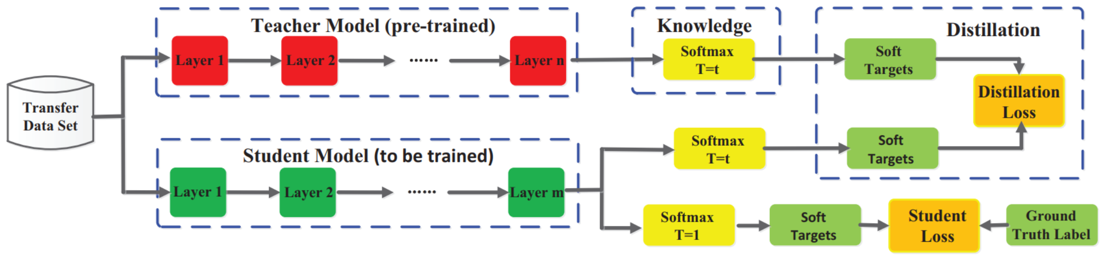


Figure 3.4: The specific architecture of the benchmark knowledge distillation [22]

- Feature-Based Knowledge Distillation: Instead of focusing on the final output, feature-based knowledge distillation involves transferring intermediate representations (features) learned by the teacher model to the student. The rationale is that these intermediate features contain rich information about the data that can help the student model learn better representations and improve its performance. The student model is trained to minimize the difference between its own feature representations and those of the teacher.

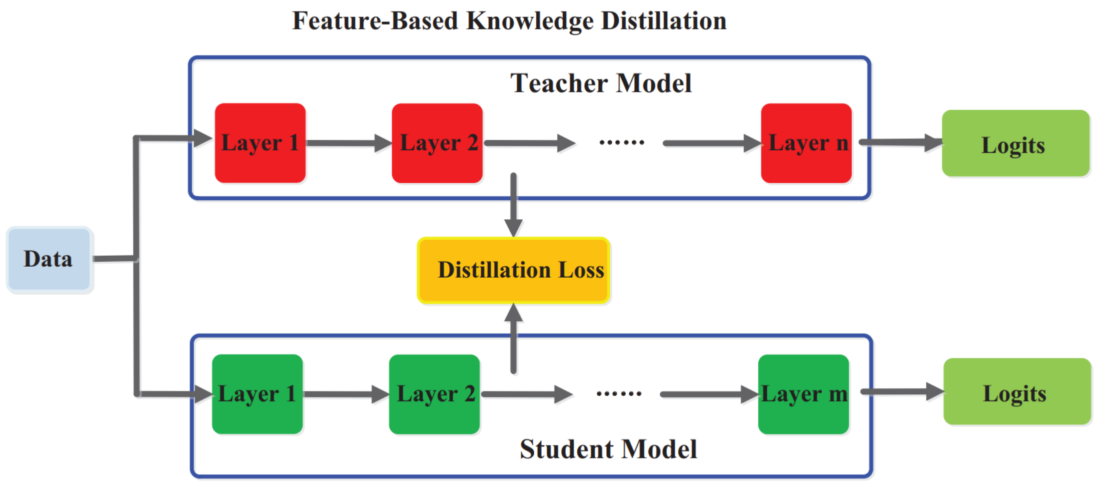


Figure 3.5: The generic feature-based knowledge distillation [22]

- Relation-Based Knowledge Distillation: This approach takes knowledge distillation a step further by focusing on the relationships between data points or features within the data, as learned by the teacher model. It aims to teach the student model to replicate these relationships or interactions, such as the similarity between different data points. This method often involves constructing a relational graph or using other techniques to model the data relationships captured by the teacher, and then training the student model to mimic these.

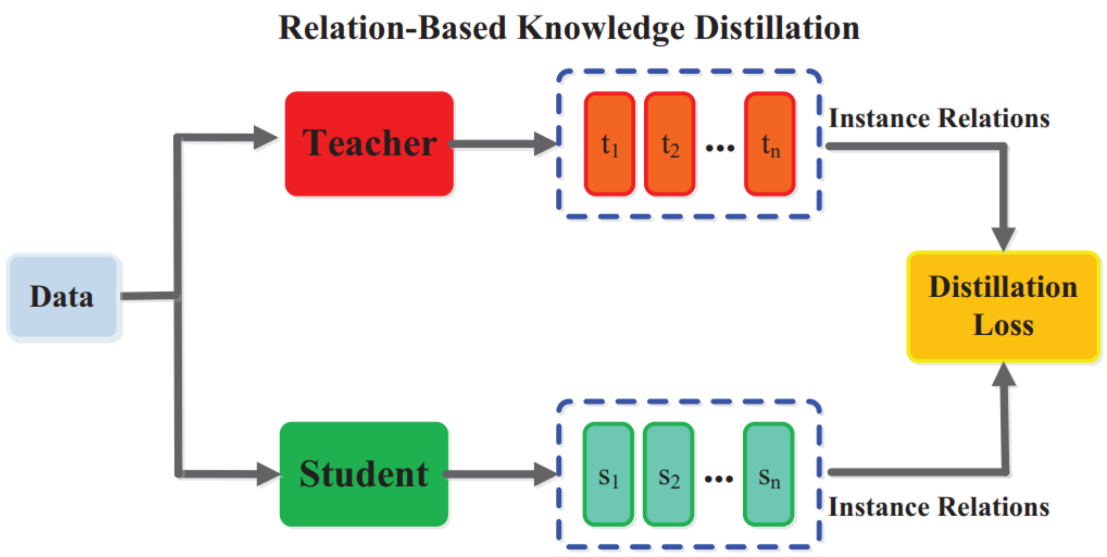


Figure 3.6: The generic instance relation-based knowledge [22]

Each of these methods offers a different mechanism for leveraging the knowledge encapsulated in a teacher model, with the goal of improving the efficiency and performance of the student model in a more nuanced way than simply replicating the final output.

3.2 Application in Our Scenario

In the context of our specific regression task, which lacks intermediary layers and explicit relational structures, an intuitive and direct approach emerges as the most suitable for our objectives: response-based knowledge distillation. This methodology emphasizes the transfer of knowledge from a sophisticated, often cumbersome, teacher model to a more compact and efficient student model by aligning the student's output predictions with those of the teacher. Response-based distillation focuses solely on the final output responses of the models involved. This simplicity aligns perfectly with our scenario, where the absence of intermediate layers or explicit relationships between inputs eliminates the need for complex distillation techniques that leverage such structures. By adopting response-based knowledge distillation, we can efficiently capture the nuanced output behavior of the teacher model, thereby endowing the student model with the ability to approximate the regression task with a high degree of accuracy.

Given the current landscape where mainstream applications are predominantly centered around recognition tasks, the mechanism of action for dark knowledge based on soft labels is quite clear. However, in the domain of knowledge distillation, experimental research on regression tasks is somewhat limited. Fortunately, the paper [23] offers considerable insights, proposing several methods that have been proven to be effective.

3.2.1 Regularization Loss for Regression

The regularization loss for regression tasks [23], aiming to minimize the prediction error of the student model, is given by:

$$L_{reg} = \frac{1}{n} \sum_{i=1}^{n} \min \left(\|p_S - p_{gt}\|^2, \|p_S - p_T\|^2 \right) \tag{3.1}$$

Here, L_{reg} is the regularization loss, p_S and p_T are the predictions of the student and teacher models respectively, and p_{gt} represents the ground truth. This formula encourages the student model to closely follow the teacher's predictions when they are reliable.

3.2.2 Attentive Imitation Loss

In the distillation process for regression tasks, the quality of the distilled knowledge is of paramount importance. This is because, in comparison to recognition tasks, the form in which dark knowledge exists in regression tasks is more "uniform." If the distilled knowledge provided by the teacher model is of low quality, it can significantly impact the effectiveness of knowledge distillation. Therefore, assessing the quality of knowledge output by the teacher model is crucial in our scenario.

The attentive imitation loss [23], which adjusts the importance of the imitation component based on the teacher's performance, is formulated as

$$L_{reg} = \frac{1}{n} \sum_{i=1}^{n} \alpha \|p_S - p_{gt}\|^2 + (1-\alpha)\Phi_i \|p_S - p_T\|^2 \tag{3.2}$$

$$\Phi_i = 1 - \frac{\|p_T - p_{gt}\|^2}{\eta} \tag{3.3}$$

where Φ_i is a weighting factor and η is a normalization parameter calculated as the difference between the maximum and minimum errors of the teacher's predictions across the dataset.

The AIL mechanism employs a normalized Φ to assess the quality of distilled knowledge during the training process. This assessment is based on the ratio of the error in the teacher network's current prediction to the "global" maximum error observed in the teacher network, thereby determining the reference value of the current distilled knowledge. This approach facilitates a smooth de-weighting of the distillation process for the current iteration.

3.3 From Offline to Online

Typically, in the process of knowledge distillation, the teacher network is pre-trained and its outputs are utilized as a form of dark knowledge for distillation with a student network, which was our initial research direction. However, training the student and teacher networks together and allowing them to mutually distill knowledge from each other yielded better results in classic image classification tasks, compared to the approach where the student network unilaterally receives outputs from the teacher network for knowledge distillation. This method of jointly training two models of different capacities and enabling mutual distillation between them is referred as "Mutual Learning" [15]

$$D_{KL}(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right). \tag{3.4}$$

Here, $D_{KL}(P||Q)$ quantifies the divergence between the predicted probability distributions P of network Θ_1 and Q of network Θ_2, facilitating the mutual learning process by aligning their predictions [15]

$$L_{\Theta_1} = L_{C_1} + \lambda D_{KL}(P_2||P_1). \tag{3.5}$$

The total loss L_{Θ_1} for network Θ_1 combines its supervised learning loss L_{C_1} with the KL divergence to the predictions of network Θ_2, weighted by λ, to guide Θ_1 towards better generalization by learning from Θ_2 [15]

$$L_{\Theta_2} = L_{C_2} + \lambda D_{KL}(P_1||P_2). \tag{3.6}$$

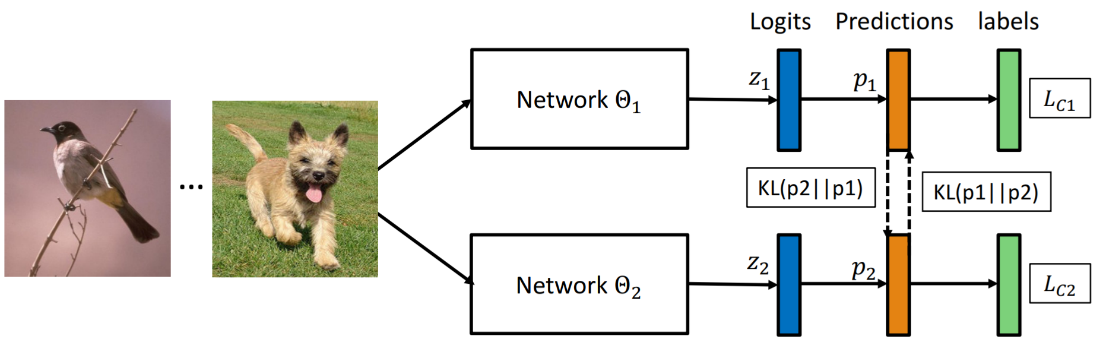


Figure 3.7: The specific architecture of the mutual learning [15]

Similarly, L_{Θ_2} for network Θ_2 merges its own supervised loss L_{C_2} with the KL divergence from Θ_1's predictions, allowing Θ_2 to also benefit from the insights gained by Θ_1, thus embodying the essence of mutual learning.

3.3.1 Deep Mutual Learning Algorithm

The KL divergence between the two models, as formally described, represents the distillation of errors using the distilled knowledge previously mentioned. Assuming these two models comprise one larger and one smaller model, according to the logic of this algorithm [15], the smaller model acquires distilled knowledge from the results of the larger model, while the larger model also obtains distilled knowledge from the smaller model to complete its iteration. This process allows both the larger and smaller models to be trained together, rather than employing the conventional approach of training and distilling knowledge from a pre-trained larger model to a smaller model.

The Deep Mutual Learning (DML) algorithm, Algorithm 1, is devised to bolster the performance of neural networks through a collaborative learning approach amongst an ensemble of student networks. Contrary to the traditional teacher-student paradigm where a pre-trained, larger model (teacher) facilitates the training of a smaller model (student), DML enables multiple student networks to concurrently learn from each other.

Algorithm 1 Deep Mutual Learning

```
 1: Input: Training set $X$, label set $Y$, learning rates $\gamma_1^t$, $\gamma_2^t$
 2: Initialize: Models $\Theta_1$ and $\Theta_2$ to different initial conditions.
 3: repeat
 4:     $t = t + 1$
 5:     Randomly sample data $x$ from $X$.
 6:     1: Update the predictions $p_1$ and $p_2$ of $x$ by (1) for the current mini-batch
 7:     2: Compute the stochastic gradient and update $\Theta_1$:
 8:         $\Theta_1 \leftarrow \Theta_1 - \gamma_1^t \frac{\partial L_{\Theta_1}}{\partial \Theta_1}$
 9:     3: Update the predictions $p_1$ and $p_2$ of $x$ by (1) for the current mini-batch
10:     4: Compute the stochastic gradient and update $\Theta_2$:
11:         $\Theta_2 \leftarrow \Theta_2 - \gamma_2^t \frac{\partial L_{\Theta_2}}{\partial \Theta_2}$
12: until convergence
```

Algorithm input:

- *Training set X*: The dataset utilized for training.
- *Label set Y*: The corresponding labels for the training dataset.
- *Learning rates γ_1^t and γ_2^t*: The learning rates for the models, which could vary over time, indexed by t.

Algorithm initialization:

- *Models* Θ_1 *and* Θ_2: Two neural network models are initialized under different starting conditions to ensure diversity in the learning perspectives from the onset.

Algorithm Repeat Until Convergence:

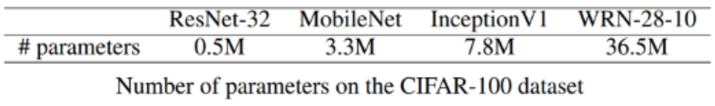
1. Increment the time step by 1, progressing to the next iteration.
2. Randomly select data x from the training set X.
3. For the current mini-batch, compute the predictions p_1 and p_2 using step **1** in the line 6, involving forward propagation through each model based on their parameters at time t.
4. Compute the gradient of the combined loss function step **2** in the line 7 with respect to Θ_1 and update its parameters using this gradient and the learning rate γ_1^t. This step promotes Θ_1 not only to fit the training data but also to align its predictions with those of Θ_2.
5. Similar to step **3** in the line 9, recalculate predictions for the mini-batch to reflect the updated parameters of Θ_1.
6. Similarly, update Θ_2 by computing the gradient of the combined loss function step **4** in the line 10 with respect to Θ_2 and using this gradient and the learning rate γ_2^t. This ensures Θ_2 also learns from both the label data and the predictions of Θ_1.

The steps are repeated until both models converge to a state where their parameters cease to undergo significant changes, indicating that they have effectively learned to predict the training data accurately while aligning their predictions as closely as possible with each other.

The DML strategy fosters a collaborative learning environment by updating each model based on its accuracy on the training data and its alignment with the other model's predictions. This mutual improvement process is distinct from traditional distillation, as it allows both models to evolve and enhance together, thereby uncovering insights that might be missed in isolation or from a fixed teacher. The ultimate goal is to guide the models towards solutions that generalize better to unseen data, facilitated by the collaborative exploration of a richer set of hypotheses about the data.

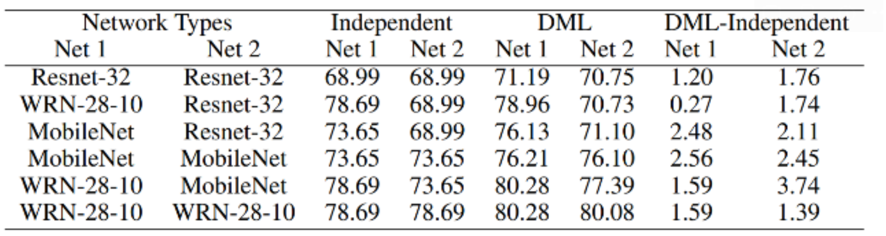
From the experimental results in the Table 3.1, it can be observed that different models, regardless of how they are paired, significantly improve their performance after undergoing DML (Deep Mutual Learning) distillation. Moreover, their performance is relatively better compared to the traditional 1 distills 2 method. So, Mutual Deep Learning (MDL) is capable of facilitating the concurrent learning of two models of the same architecture. However, the most significant improvements are witnessed when a larger model and a smaller model learn together. Engaging a smaller model in MDL with a similarly structured larger model can dramatically enhance its performance, even surpassing the outcomes achieved by distilling knowledge from a fully trained larger model into the smaller model.

Table 3.1: Results of MDL experiments [15]

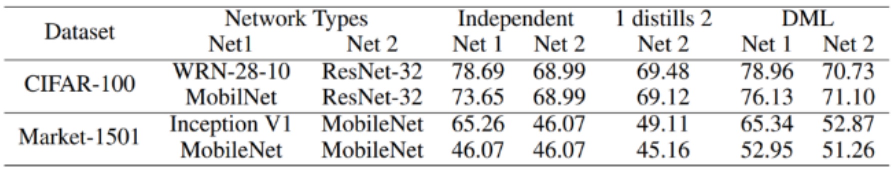


	ResNet-32	MobileNet	InceptionV1	WRN-28-10
# parameters	0.5M	3.3M	7.8M	36.5M

Number of parameters on the CIFAR-100 dataset



Network Types		Independent		DML		DML-Independent	
Net 1	Net 2	Net 1	Net 2	Net 1	Net 2	Net 1	Net 2
Resnet-32	Resnet-32	68.99	68.99	71.19	70.75	1.20	1.76
WRN-28-10	Resnet-32	78.69	68.99	78.96	70.73	0.27	1.74
MobileNet	Resnet-32	73.65	68.99	76.13	71.10	2.48	2.11
MobileNet	MobileNet	73.65	73.65	76.21	76.10	2.56	2.45
WRN-28-10	MobileNet	78.69	73.65	80.28	77.39	1.59	3.74
WRN-28-10	WRN-28-10	78.69	78.69	80.28	80.08	1.59	1.39

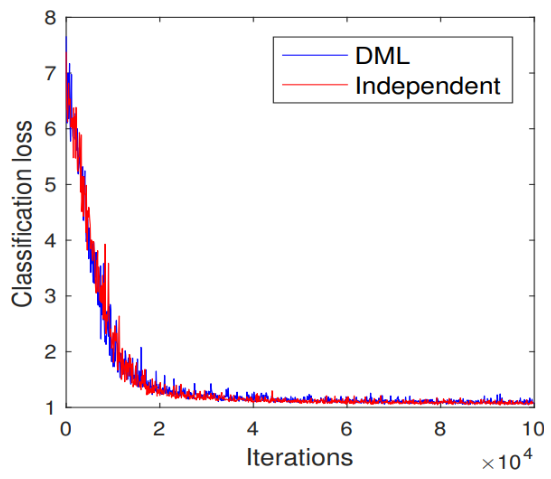


Dataset	Network Types		Independent		1 distills 2	DML	
	Net1	Net 2	Net 1	Net 2	Net 2	Net 1	Net 2
CIFAR-100	WRN-28-10	ResNet-32	78.69	68.99	69.48	78.96	70.73
	MobilNet	ResNet-32	73.65	68.99	69.12	76.13	71.10
Market-1501	Inception V1	MobileNet	65.26	46.07	49.11	65.34	52.87
	MobileNet	MobileNet	46.07	46.07	45.16	52.95	51.26

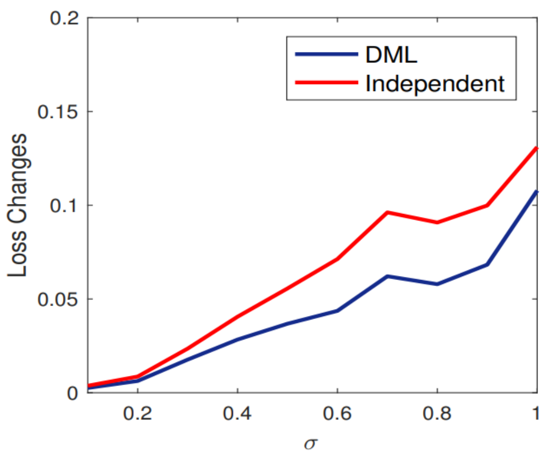
Compared to conventional optimization techniques, Deep Mutual Learning (DML) does not necessarily lead us to a superior or deeper minimum within the training loss landscape. Rather, its strength lies in identifying a broader or more dependable minimum, which offers enhanced generalization to test data and exhibits greater robustness.

To empirically substantiate this assertion, the authors conducted a focused experiment employing the Market-1501 dataset alongside the MobileNet backbone network. This investigation aimed to demonstrate DML's efficacy in pinpointing a minimum that is notably more robust, further validating the approach's utility in practical applications.

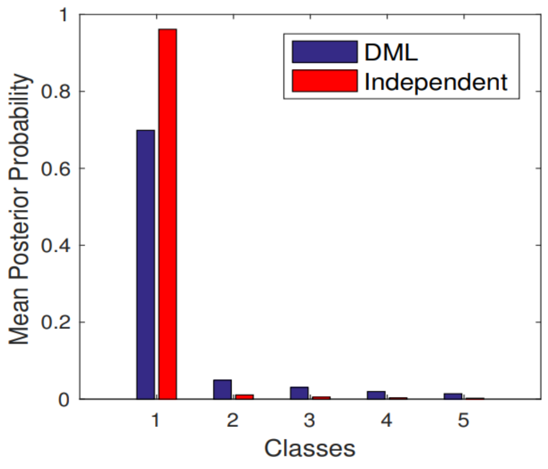
The authors compared the loss variations of the DML model and the standalone model before and after adding Gaussian noise. From Figure 3.8(a) it's evident that the depth of the minima for both models is the same. However, after adding Gaussian noise, we can see from Figure 3.8(b) and 3.8(c) that the training loss for the standalone model increased significantly, while the DML model experienced a smaller increase in training loss. This suggests that the DML model has found a broader and more robust minimum which could better resist the noise, thereby



(a) Training loss



(b) Loss change with noise



(c) Posterior certainty comparison

Figure 3.8: MDL effectiveness resisting white noise [15]

offering better generalization performance.

DML necessitates that each network aligns its probability estimates with those of its counterpart. Should one network forecast a zero while its counterpart forecasts a non-zero, the former incurs a significant penalty. However, when both models concurrently err in their predictions, the penalty for this misstep diminishes. Consequently, the value attributed to "similar" erroneous predictions—where both networks falter—does not dwindle to zero. This mechanism potentially enhances the models' generalization capabilities and mitigates the risk of overfitting. Therefore, DML aims for a more expansive minimum by synchronizing the mutual probabilities of "reasonable" sub-optimal predictions, as opposed to indiscriminately imposing severe penalties for incorrect classifications.

Inspired by the insights presented in [15], I ultimately transitioned from offline distillation to an online distillation approach based on mutual distillation. Online distillation entails that the student and teacher models engage in mutual distillation, learning from each other by incorporating distilled knowledge as a component of the loss function, thereby iteratively refining their own parameters. This paradigm shift emphasizes a dynamic learning process where both models simultaneously evolve through reciprocal feedback, enhancing their learning efficacy beyond unidirectional knowledge transfer methods

$$L_{reg_S} = \frac{1}{n}\sum_{i=1}^{n} \alpha\|p_S - p_{gt}\|^2 + (1-\alpha)\Phi_t\|p_S - p_T\|^2 \tag{3.7}$$

$$L_{reg_T} = \frac{1}{n}\sum_{i=1}^{n} \alpha\|p_T - p_{gt}\|^2 + (1-\alpha)\Phi_s\|p_S - p_T\|^2. \tag{3.8}$$

Chapter 4

Experimental Environment and Configuration

4.1 Implementation of Long-Distance Capacitive Sensors for Indoor Human Monitoring

The cutting-edge implementation of long-distance capacitive sensors for indoor personnel positioning hinges on a meticulously engineered design that pushes the envelope of sensitivity and operational range, while deftly suppressing extraneous noise.

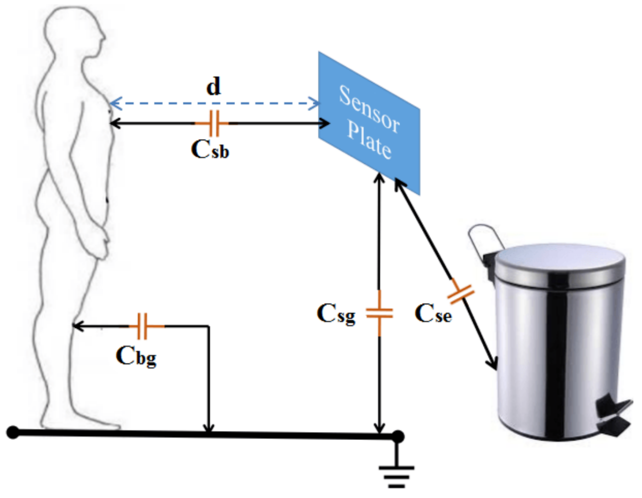


Figure 4.1: Capacitance of a load mode capacitive sensor [3]

As illustrated in Figure 4.1, the system's efficacy is anchored in its nuanced manipulation of the human body's influence on electrical capacitance. The setup intricately discerns variations across multiple capacitance components such as C_{sg} (sensor to ground), C_{bg} (body to ground), C_{se} (sensor to environment), and C_{sb} (sensor to body). These components are inherently susceptible to alterations engendered by a person's proximity to the sensor plate, which directly translates to the modulation of the sensor's electric field and the resultant capacitance.

4.2 Environment Configuration

The arranged experimental setup as delineated in this study, explicitly portrayed in Figures 4.2 and 4.3, constitutes a sophisticated system designed for the unobtrusive monitoring of indoor human activities. Within a controlled lab environment, a virtual room spanning an area of 3 m × 3 m serves as the experimental arena. This room is methodically equipped with four capacitive sensors, each positioned at the midpoint of the room's walls. These strategically located sensors are imperative components that employ loading mode operations to capture and chronicle the nuanced movements of an individual navigating within the delineated space. The visualization provided in Figure 4.2 is crucial, as it elucidates the arrangement and functioning of these sensors in real-time spatial tracking. The design integrates

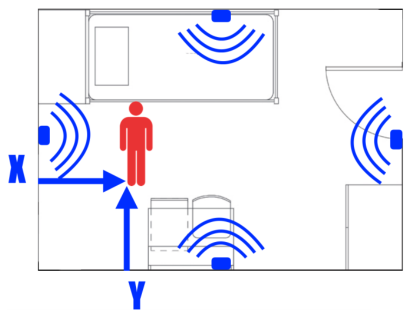


Figure 4.2: Four capacitive sensors centered on the walls of a 3 m × 3 m virtual room in the lab trace the position of a person moving in the space [13]

capacitive sensors with a specified dimension of 16 cm × 16 cm plates, which have been fine-tuned to collect data thrice per second. This rapid collection frequency ensures a dense and information-rich dataset, preparing the ground for in-depth signal analysis and interpretation. A pivotal aspect of the setup is an ultrasonic reference system consisting of four anchoring units, which complement the capacitive sensors by offering a high-fidelity localization of a mobile tag, with a commendable accuracy of ±2 cm at a sampling rate of 15 Hz. This dual system of data acquisition is essential for validating the movement paths registered by the capacitive sensors, as delineated in Figure 4.3.

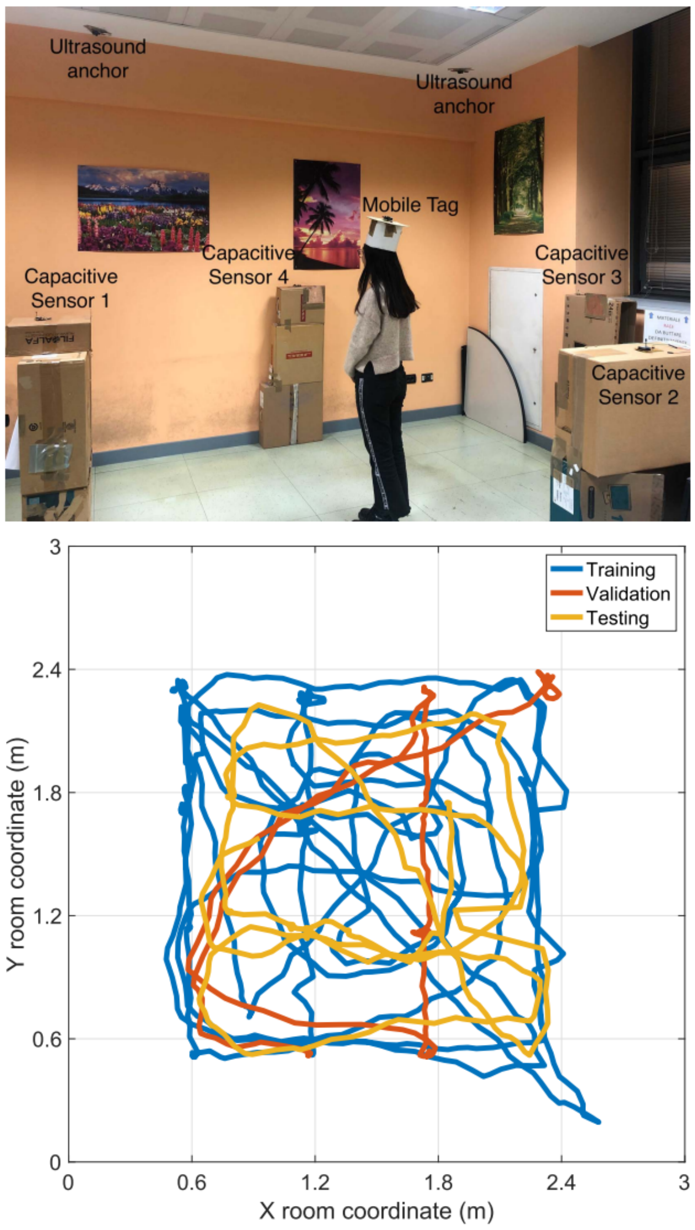


Figure 4.3: Virtual room used for movement tracking experiments and person trajectory (split into segments for NN training, validation and testing) [13]

Chapter 5

Dataset Generation and Feature Selection

5.1 Dataset Generation

In our experimental protocol, the computer captures readings from four distinct capacitive sensors at a frequency of 3 Hz, after undergoing preliminary processing. Simultaneously, it also acquires highly precise readings from an ultrasonic sensor at the same frequency to serve as the ground truth. Our objective is to employ a technique that utilizes the readings from the four capacitive plates to determine a person's coordinates within a 3 m $\times$ 3 m area with high precision. The system, which inputs the capacitive values from the four plates and outputs x and y coordinates, is highly nonlinear. Consequently, conventional system identification methods fail to yield satisfactory results. Therefore, we have opted to employ a neural network to model this system.

To train this network, we moved around within the 3 m $\times$ 3 m space carrying an ultrasonic sensor beacon [24], collecting approximately 2000 data sets. Each data set comprises the capacitive readings from the four sensors at different elevations, along with the ultrasonic sensor readings as ground truth.

In the depiction provided by Figure 4.3, the experimental narrative unfolds further with the introduction of a human participant, who is equipped with a head-mounted mobile tag. This addition introduces an element of dynamism to the experiment, as the participant's locomotion within the virtual space is meticulously documented through the capacitive sensor array and the ultrasound anchors, offering a multi-faceted perspective on motion tracking.

The graphic representation in the figure's lower section is especially telling, as it intricately plots the participant's trajectory through the room. This plot is segmented into distinct phases—training, validation, and testing—each playing a pivotal role in the machine learning process applied to the neural networks.

Table 5.1: Example of capacitive sensors readings labelled with ground truth

Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**	**X coordinate** (m)	**Y coordinate** (m)
0.774 61	0.930 93	0.932 54	0.492 37	0.928 76	1.578 8
0.764 7	0.929 71	0.932 71	0.461 78	0.877 43	1.551 9
0.754 99	0.926 04	0.933 03	0.441 25	0.806 8	1.525 9
...	...	...	...	...	...

5.2 Feature Selection

In the experiments, it was found that obtaining comparatively accurate values from single measurements taken solely from four capacitor plates proved challenging. Given the continuous nature of human movement indoors, it is logical to consider a continuous time series as input features. Therefore, we attempted to use the measurements from the capacitor plate sensors over five time steps as input features, with the ground truth of the central time step serving as the reference ground truth. We refer to this continuous time series as “windows.” Experimental validation has shown that optimal results are achievable using neural networks of any architecture when the window is set to five time steps. Hence, in this paper, we utilize the measurements from five time steps as input features.

Feature selection is a pivotal step in constructing neural network models for indoor activity reconstruction. In the case study examined, as shown in Figure 5.1, sensor data is initially discretized at a frequency of 3 Hz into quartets of samples S_1, S_2, S_3, S_4. This step is crucial to ensure the integrity of the time series in the data, reflecting the nuanced changes in human dynamics. These quartets are then chronologically concatenated, forming the feature set input into the neural network. The way these features are structured not only takes into account the information of individual samples but also considers the temporal relationships between samples through a windowing method, which is particularly significant for capturing patterns of human motion.

The feature selection process, as implemented, enables the neural network to learn and extract valuable information from noisy sensor data. Especially in dynamic and complex indoor environments, these carefully selected features assist the network in better understanding and predicting human movement patterns.

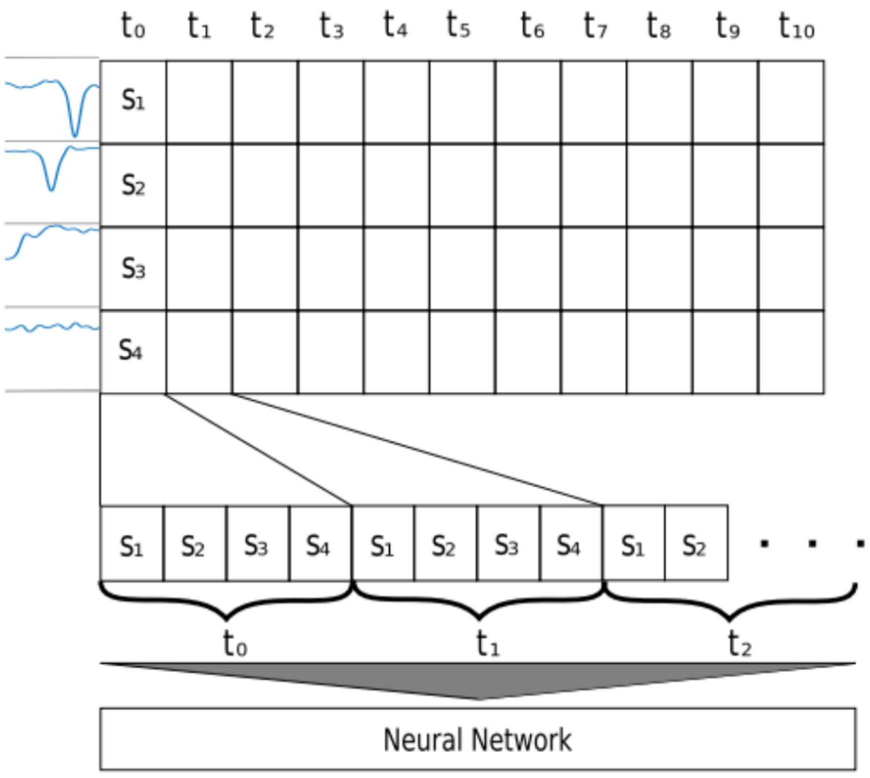

t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
S_1
S_2
S_3
S_4
S_1 S_2 S_3 S_4 S_1 S_2 S_3 S_4 S_1 S_2 . . .
t_0
t_1
t_2
Neural Network

Figure 5.1: Sensor data (top-left) is discretized at 3 Hz in four-sample tuples S1, . . . , S4, which are then concatenated in chronological order and input to the neural network with appropriate windowing [13]

Chapter 6

Selected Neural Networks

6.1 General Considerations based on Feature Selection and Specific Application Context

In the selection of an appropriate neural network model for regression tasks involving continuous time series data as features, critical considerations include the model's capability to capture long-term dependencies and its computational complexity. Time series data are characterized by their sequential nature, where the value at a current point may be influenced by the values at multiple preceding points. Therefore, the model must be able to understand and utilize these long-term dependencies to make accurate predictions. Recurrent Neural Networks (RNN) and its variants, such as Long Short-Term Memory networks (LSTM) and Gated Recurrent Units (GRU), are naturally favored due to their intrinsic design to handle such data. These models, with their recurrent connections, can process sequences of arbitrary length, effectively capturing long-term dependencies within the time series.

However, the complexity of the neural network model is also a crucial consideration. For embedded devices or environments with limited resources, the size of the model and its computational demands can become limiting factors. While larger networks may offer better performance, they also require more computational resources and storage space. Thus, in these scenarios, finding or designing models with higher computational efficiency becomes particularly important. Lightweight network designs, such as simplified variants of LSTM, can reduce the number of parameters and operations while maintaining model performance.

6.2 LSTM Neural Network

LSTM [25] networks, as a variant of RNNs, are designed to address the issue of long-term dependencies, enabling the capture of information across extended temporal intervals. This characteristic renders LSTMs highly effective in applications involving long sequence data, such as natural language processing, speech recognition, and complex time series analysis. The capability of LSTMs to manage this arises from their intricate gating mechanisms that selectively remember or forget information, thereby maintaining a long-term state or context at each point in the sequence.

The Architecture of LSTM as shown in the Figure6.1 and 6.2

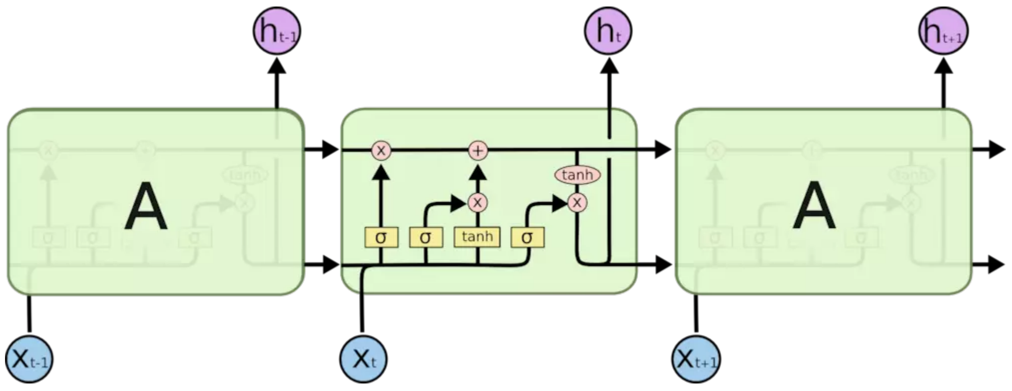


Figure 6.1: Architecture of LSTM

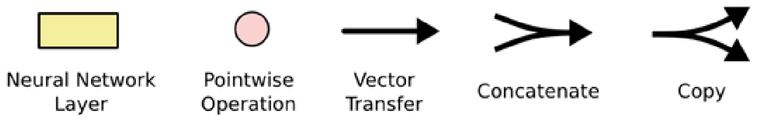


Figure 6.2: Elements of LSTM

- **Neural Network Layer**: A layer in a neural network used for learning.
- **Pointwise Operation**: Operations performed element-wise, such as element-wise multiplication, addition, vector sum, etc.
- **Vector Transfer**: The movement of a vector in the direction indicated by an arrow.
- **Concatenate**: Joining two vectors together end-to-end.
- **Copy**: Duplicating a vector into two copies.

The following is an explanation of an example of a weather forecasting task from [25]:

1. Figure 6.3 shows the process of transforming a 2D image into a 3D tensor. This is a typical preprocessing step in machine learning and deep learning for handling image data. It often involves taking a 2D image with pixel values and converting it into a 3D tensor, where the third dimension represents different channels (like RGB channels in a color image) or sometimes different features extracted from each pixel or patch. The 'P' here indicates a single pixel or a patch of pixels being transformed into a 3D structure with depth, which could represent multiple features.

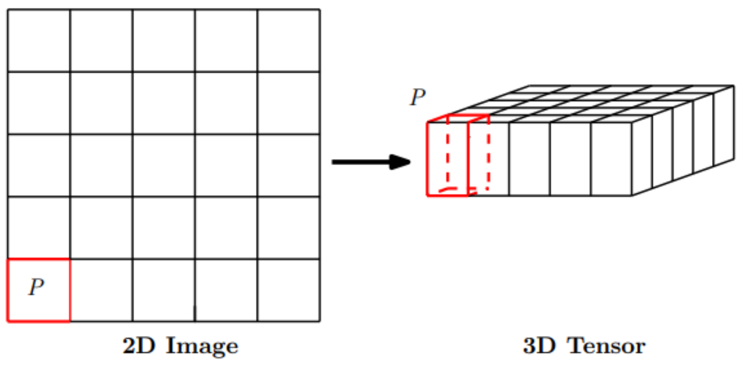


Figure 6.3: Transforming 2D image into 3D tensor

2. Figure 6.4 illustrates the inner structure of a ConvLSTM unit. ConvLSTM is a type of recurrent neural network (RNN) that is well-suited for spatiotemporal data because it can maintain spatial information through the use of convolutional structures in both the input-to-state and state-to-state transitions. In this figure, ‘Ht‘ and ‘Ct‘ represent the hidden state and cell state at time ‘t‘, respectively. The ‘Xt‘ and ‘Xt+1‘ at the bottom represent the input at time ‘t‘ and time ‘t+1‘. The dashed arrows suggest the flow and transformation of data through the ConvLSTM cell over time.

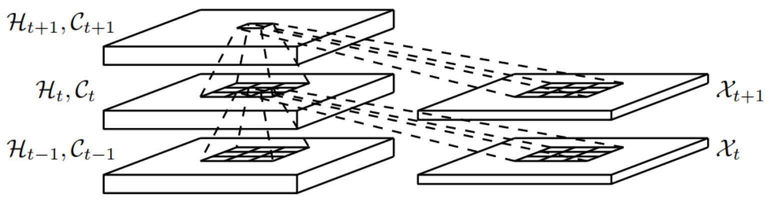

$\mathcal{H}_{t+1}, \mathcal{C}_{t+1}$
$\mathcal{H}_{t}, \mathcal{C}_{t}$
$\mathcal{H}_{t-1}, \mathcal{C}_{t-1}$
$\mathcal{X}_{t+1}$
$\mathcal{X}_{t}$

Figure 6.4: Inner structure of ConvLSTM

3. Figure 6.5 presents an encoding-forecasting ConvLSTM network for precipitation nowcasting. It consists of two parts: the encoding network, which processes the input sequence and encodes it into a higher-level representation; and the forecasting network, which uses the encoded representation to make predictions about the future. The 'Copy' labels indicate that the final state of the encoding network is used as the initial state for the forecasting network. This network architecture allows for sequential processing of spatial data and is capable of making predictions about future frames, which is essential in tasks such as weather forecasting.

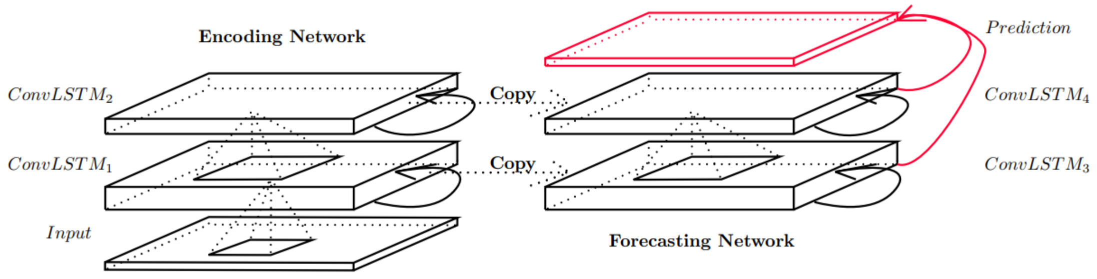


Figure 6.5: Encoding-forecasting ConvLSTM network for precipitation on nowcasting

The overall theme of these figures is to represent the ConvLSTM network's ability to handle spatiotemporal sequences for tasks such as predicting future rainfall intensity from radar image sequences. The figures are taken from a paper likely discussing how to apply deep learning techniques to weather prediction, specifically for predicting precipitation by learning from sequences of weather radar images. However, when the features of interest encompass only short time sequences, around five time steps of sensor readings—the advantages of LSTMs become less pronounced. In such scenarios, the dependencies within the sequence are relatively short-term, negating the need for complex gating mechanisms to preserve long-term state information. Indeed, employing LSTMs in these contexts may introduce unnecessary computational overhead and model complexity, which is particularly disadvantageous in resource-constrained environments. For instance, in embedded systems or real-time processing applications, computational efficiency and response speed are crucial, and overly complex models can adversely impact these performance metrics.

Given this context, despite the clear advantages of LSTM networks in handling long sequence data, their complex structure and computational costs do not constitute a justifiable choice for scenarios involving only short-term time sequences, such as a few timesteps of sensor readings. Selecting a more appropriate model architecture based on the specific requirements of the application scenario not only enhances the model's efficiency and utility but also ensures the effective use of resources, thereby meeting performance requirements without incurring unnecessary complexity and computational expense.

6.3 Capsule Neural Network

Capsule Neural Networks (CapsNets) are considered one of the potentially optimal architectures within the realm of neural network technologies, particularly due to their distinctive approach to handling input data when compared to traditional Convolutional Neural Networks (CNNs). Unlike CNNs, which employ pooling layers to reduce the spatial dimensions of the input data through downsampling, CapsNets are designed to retain the hierarchical spatial relationships between the parts of an object. This retention of spatial hierarchies is crucial because downsampling inherently leads to a loss of information, which can be detrimental when the task requires preserving intricate spatial relationships.

The Architecture of Capsule Neural Network as shown in the Figure 6.6 and 6.7.

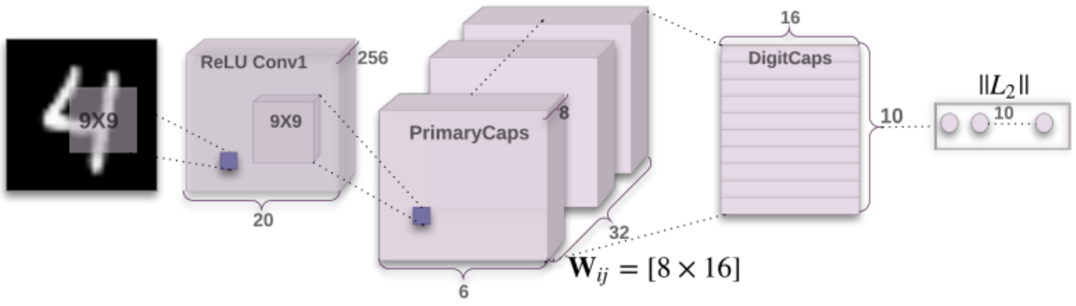


Figure 6.6: A simple CapsNet with 3 layers [26]

6.3.1 CapsNet Architecture

The architecture CapsNet neural network is shown in Figure 6.6 and contains:

- **Input Image**: An image from the MNIST dataset representing a handwritten digit.
- **Convolutional Layer (ReLU Conv1)**: The initial layer applies 256 9×9 convolutional kernels with ReLU activations to detect local features.
- **PrimaryCapsules**: A convolutional capsule layer with 32 channels of 8D capsules. Capsules represent combinations of features detected by the Conv1 layer. The vector's length indicates the probability of feature presence.

- **DigitCaps**: Capsules in this layer represent the 10 possible digit classes. Each capsule receives inputs from the PrimaryCapsules layer through a dynamic routing mechanism, which depends on the agreement between the predicted outputs and the actual inputs.
- **Output**: The activity vector's length in the DigitCaps layer signifies the presence probability of a digit, and its orientation represents the instantiation parameters.
- **Weight Matrix (W_{ij})**: A weight matrix transforms outputs from the PrimaryCapsules to predictions for the DigitCaps.

Algorithm 2 Routing algorithm [26]

```
1: procedure ROUTING($\hat{u}_{j|i}, r, l$)
2:     for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$ : $b_{ij} \leftarrow 0$.
3:     for $r$ iterations do
4:         for all capsule $i$ in layer $l$ : $c_i \leftarrow \text{softmax}(b_i)$ ▷ softmax computes Eq. 3
5:         for all capsule $j$ in layer $(l+1)$ : $s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$ ▷ squash computes Eq. 1
6:         for all capsule $j$ in layer $(l+1)$ : $v_j \leftarrow \text{squash}(s_j)$
7:         for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$ : $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
8:     return $v_j$
9: end procedure
```

6.3.2 Dynamic Routing Algorithm

The dynamic routing algorithm, Algorithm 2, is pivotal for directing information flow in Capsule Networks, facilitating effective representation of hierarchical data relationships. Herein, we outline the algorithm's procedure in a consolidated format:

1. **Initialization:** Logit values b_{ij} between capsules across layers are initialized to zero, setting a neutral starting point for routing.
2. **Iterative Routing:** Through r iterations, the algorithm refines the coupling coefficients, balancing between precision and computational demands.
3. **Softmax Calculation:** Normalizes coupling coefficients c_i for each capsule using softmax, ensuring that probabilities across potential parent capsules sum to one.
4. **Weighted Sum:** Each capsule in layer $(l+1)$ computes its input as the weighted sum of prediction vectors from the lower layer, with weights given by the current routing preferences.
5. **Squashing:** Applies a non-linear squashing function to transform the total input into output vectors, effectively encoding entity presence and properties.

6. **Logit Update:** Updates the logits b_{ij} based on the scalar product between prediction vectors and actual outputs, reinforcing the routing based on prediction agreement.

This routing mechanism allows for a dynamic, hierarchical representation of data within Capsule Networks, enhancing their ability to parse and understand complex inputs.

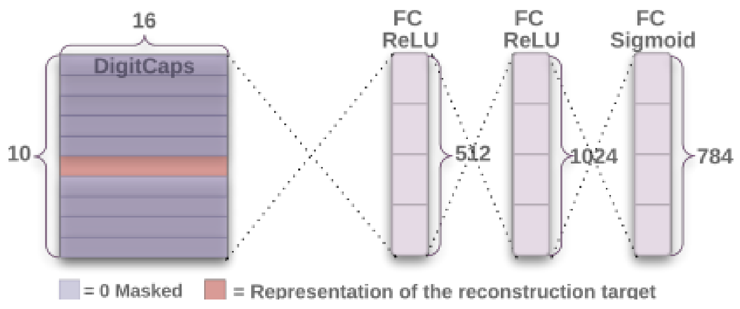


Figure 6.7: Decoder structure to reconstruct a digit from the DigitCaps layer representation

6.3.3 Decoder for Digit Reconstruction

The Decoder for digit reconstruction shown in Figure 6.7 is made of:

- **DigitCaps Layer**: Serves as the starting point for the reconstruction of the digit image.
- **Masking**: During training, vectors from all but the correct digit class are masked.
- **Decoder Network**: Comprises three fully connected layers with ReLU activations (except the final Sigmoid layer) that reconstruct the digit image.
- **Output**: The reconstruction loss, or the squared difference between the reconstructed image and the input, is minimized during training with the true label serving as the reconstruction target.

The CapsNet architecture employs vector outputs from capsules to encode instantiation parameters. This design enables the network to generalize effectively to new viewpoints and achieve superior performance in tasks such as segmentation and classification of overlapping digits.

Despite the larger scale of CapsNets compared to traditional CNNs, the complexity of the chosen features does not necessitate a proportionally large increase in the number of parameters. This is partly because the dynamic routing mechanism efficiently utilizes the network's capacity to focus on relevant spatial hierarchies without needing to significantly increase the parameter count. As a result, the parameter overhead remains within an acceptable range, making CapsNets a viable and potentially superior alternative to CNNs for applications where the preservation of spatial hierarchies and the avoidance of information loss through downsampling are critical.

6.4 Temporal Convolutional Neural Network

In the exploration of optimal neural network architectures for applications requiring the analysis of sequential data, the Temporal Convolutional Network (TCN) model emerges as a superior contender. Distinctively, the TCN architecture offers an expanded receptive field compared to traditional Convolutional Neural Networks (CNNs) while operating under an identical parameter budget. This extended receptive field is instrumental in capturing the temporal dynamics and continuity of human movement within indoor environments, a domain where the intricacies of sequential data are paramount. Specifically, within the context of five temporal steps, the TCN's ability to integrate all pertinent information and features becomes critically important, as it ensures a comprehensive understanding of the sequence's evolution over time.

The architecture of capsule neural network is shown in Figures 6.8, 6.9, and 6.10:

1. **Overall Structure** (Figure 6.8): The diagram shows a TCN where each layer's output has the same length as the input, ensuring there's no leakage of information from future to past. The layers are stacked vertically, and dilation increases exponentially as you move up through the layers (d = 1, 2, 4, . . .), allowing the network to have a very large receptive field without increasing the depth significantly.
2. **Residual Blocks** (Figure 6.9): TCNs use residual blocks, which can contain a branch leading out to a series of transformations F whose outputs are added back to the block's input. This setup helps in training deeper networks by allowing layers to learn modifications to the identity mapping. Within a block, there are dilated causal convolutions, non-linearity (ReLU), normalization (weight normalization), and spatial dropout for regularization. An optional 1x1 convolution is included when the residual input and output dimensions differ.
3. **Example of Residual Connection** (Figure 6.10): The figure illustrates the use of a residual connection in a TCN, where blue lines represent convolutional filters in the residual function and the green line shows the identity mapping. This design is essential to allow each layer of the network to look back at all prior states, giving TCNs the ability to consider long-range dependencies.

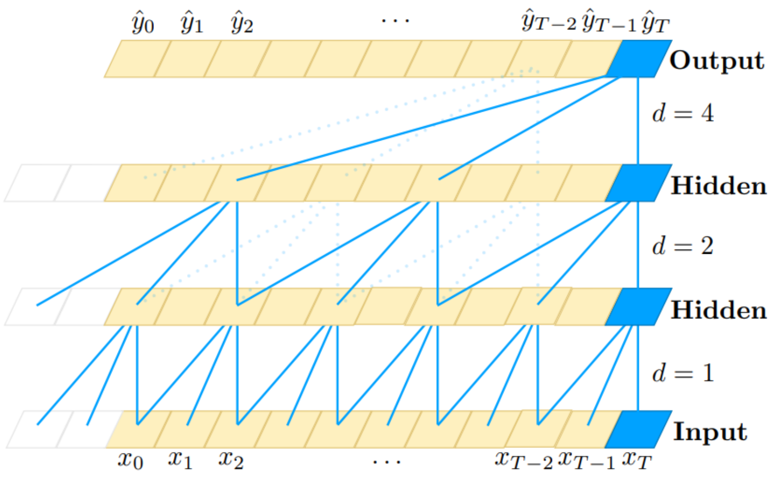

$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\cdots$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 $\cdots$ x_{T-2} x_{T-1} x_T

Figure 6.8: Overall Structure [27]

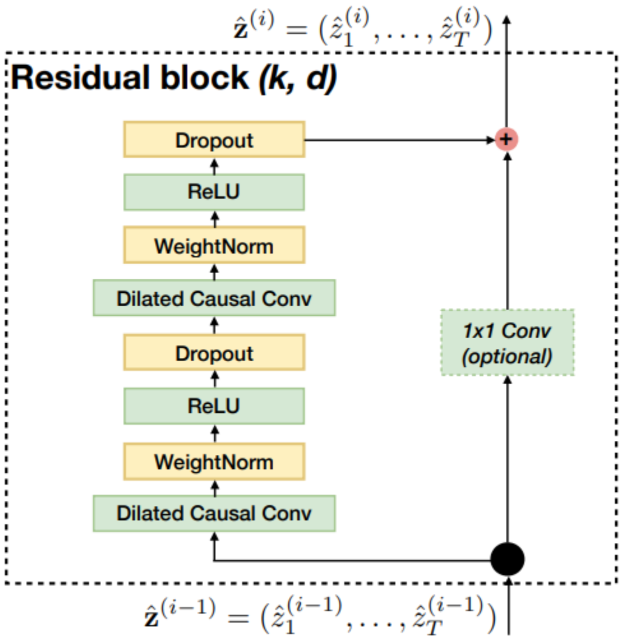

$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$

Figure 6.9: Residual Blocks [27]

The TCN model's efficiency and effectiveness are not solely confined to its enhanced receptive field. Another salient feature of the TCN is its suitability for deployment on lightweight devices, a characteristic that is increasingly relevant in the

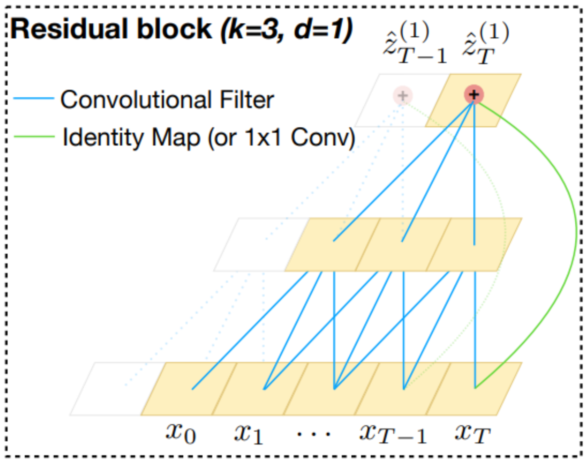


Figure 6.10: Example of Residual Connection [27]

era of mobile computing and Internet of Things (IoT) applications. This compatibility stems from the TCN's architectural efficiency, which allows it to deliver high-performance temporal data processing capabilities without necessitating the computational resources typically associated with more complex models. Consequently, the TCN model stands out not only for its theoretical contributions to the field of neural network architectures but also for its practical applicability in scenarios where computational efficiency and the ability to capture temporal dependencies are crucial.

6.5 Neural Network Selection

Given the considerations outlined in the preceding sections, the selection of Capsule Neural Networks (CapsNets) and Temporal Convolutional Networks (TCN) as the preferred neural network architectures for our application context is founded on their unique strengths and alignment with our specific requirements. The decision to incorporate CapsNets is motivated by their exceptional ability to preserve the hierarchical spatial relationships inherent in the input data. Unlike traditional Convolutional Neural Networks (CNNs) that rely on pooling layers and consequently risk information loss through downsampling, CapsNets employ a dynamic routing mechanism that efficiently captures and maintains spatial hierarchies without a significant increase in parameter count. This capability is particularly advantageous for applications demanding the preservation of intricate spatial details, making CapsNets a superior choice for scenarios where such preservation is critical to achieving high accuracy and performance.

On the other hand, the Temporal Convolutional Network (TCN) model is selected due to its expanded receptive field, which facilitates a deeper understanding of temporal dynamics and continuity, especially in applications involving sequential data. The TCN architecture's ability to integrate information across multiple time steps ensures a comprehensive analysis of sequences, a feature crucial for tasks requiring the capture of long-term dependencies within time series data. Furthermore, the TCN's architectural efficiency renders it suitable for deployment on lightweight devices, addressing the need for high-performance temporal data processing in resource-constrained environments. This compatibility with lightweight devices, combined with the TCN's enhanced receptive field, positions it as an ideal choice for applications that demand efficient and effective analysis of sequential data.

In summary, the selection of Capsule Neural Networks and Temporal Convolutional Networks is predicated on their respective abilities to address the specific challenges posed by our application scenario. CapsNets' preservation of spatial hierarchies without significant parameter overhead and TCNs' enhanced receptive field and computational efficiency align perfectly with our requirements for handling intricate spatial relationships and efficiently processing sequential data. Together, these architectures offer a synergistic solution that leverages the strengths of each model to achieve superior performance and efficiency, justifying their selection as the neural networks of choice for our application.

Chapter 7

Improvement of Online Distillation Effectiveness

7.1 Adaptive Weights

In the exploration of methodologies to augment the efficacy of online distillation techniques, inference from the mechanisms of DML and knowledge distillation, using adaptive weights could be a feasible approach. This approach postulates that the relative significance attributed to the Kullback-Leibler (KL) divergence loss in networks of divergent architectures ought to dynamically vary in accordance with the training phase.

Empirically, it is observed that networks of larger complexity, embodying a more extensive parameter space, exhibit a swifter learning curve. This characteristic necessitates an initial emphasis on the weight (w_2) associated with the teacher network's contribution, predicated on its advanced learning capabilities. Consequently, at the inception of the training regimen, a pronounced disparity between the weights (w_1 and w_2) is advocated, with w_2 significantly outweighing w_1.

As the training progression approaches maturation, the student network assimilates substantial insights from both the domain knowledge imparted by the teacher and the foundational truth. This phase of the training suggests a recalibration of the weights (w_1 and w_2) towards parity, reflecting the enhanced learning equilibrium between the student and teacher networks.

The iterative refinement of the student network, underpinned by a balanced integration of domain knowledge and empirical data, potentially augments the generalization capabilities of the teacher network. This reciprocal enhancement serves to mitigate the propensity for overfitting, fostering a symbiotic advancement in the distillation process. The deployment of an improved teacher network further enriches the quality of domain knowledge, thereby catalyzing the student network's learning trajectory.

The equations governing the adaptive weights mechanism are formalized as follows:

$$w_1(t) = f\left(L_{C_2}(t-1)\right) \tag{7.1}$$
$$L_{\Theta_1} = L_{C_1} + w_1(t) \cdot D_{KL}(p_2 \parallel p_1) \tag{7.2}$$
$$w_2(t) = f\left(L_{C_1}(t-1)\right) \tag{7.3}$$
$$L_{\Theta_2} = L_{C_2} + w_2(t) \cdot D_{KL}(p_1 \parallel p_2) \tag{7.4}$$

where L_{Θ_1} and L_{Θ_2} represent the loss functions for the student and teacher networks, respectively. The terms L_{C_1} and L_{C_2} denote the conventional loss components, while $w_1(t)$ and $w_2(t)$ are the time-varying weights assigned to the KL divergence terms, reflective of the evolving learning dynamics.

This adaptive weighting framework (see Figure 7.1) posits a promising avenue for empirical validation and exploration, particularly in the context of regression tasks. The iterative adjustment of weights, in alignment with the training phase, harbors the potential to significantly enhance performance metrics and learning efficiency in online distillation methodologies. These are achieved by the controller shown in the Figure 7.1, which continuously adjusts the weight of student and teacher during the training process based on the training status of the student model. The controller has a rolling cache to store the constantly updated training errors of the student model, and calculates the mean and standard deviation of the errors to assess the training situation of the student, as shown in Algorithm 3.

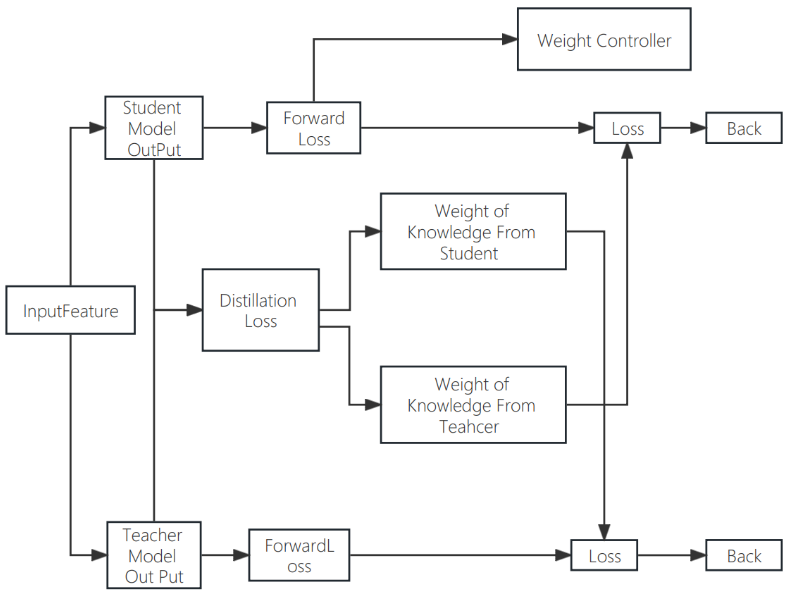


Figure 7.1: Schematic Diagram of Adaptive Weighting Framework

Algorithm 3 The mechanism of the controller

```
1: Initialize $\text{Cache}^s$ with elements $e_t^s$ for $t = 1, 2, \ldots, T$
2: Compute the average $\bar{e} = \frac{1}{N} \sum_{j=i-N}^{i} e_j$ over the last N elements
3: Compute the standard deviation $\sigma_{\text{Cache}^s} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (e_t^s - \mu_{\text{Cache}^s})^2}$
4: Compute $w_s, w_t = f(\bar{e}, \sigma_{\text{Cache}^s})$ based on the average and standard deviation
5: return $w_s$,$w_t$
```

Then, based on the logic of this method, I conducted experiments on both CPS and TCN networks under the TensorFlow environment to verify the feasibility and effectiveness of this idea.

Experimental specifications for each setting:

- Train 100 times.
- Terminate a single training if the validation error does not decrease after 200 epochs.
- Maximum of 1500 epochs for each training.
- Output the best result, referring to the lowest validation error.

are shown in Table 7.1. The configurations for the Student and Teacher structures in the Temporal Convolutional Network (TCN) experiments are shown in Table 7.1. The size of of the teacher model are far greater than those of the student model, but the student model has more parameters in its final fully connected layer compared to the teacher model. This is to enhance the expressive capability of the student model, compensating for its smaller size to a certain extent.

Table 7.1: TCN Student vs. Teacher Structures

Parameter	**Student Structure**	**Teacher Structure**
Hidden Layers	2	3
Number of Stacks	1	1
Number of Filters	8	16
Kernel Size	3	5
Dense Unit	16	8

The experimental results are shown in Figure 7.2 and reported in Table 7.2."Offline" means training is conducted in the form of offline distillation, "online" means training is conducted using the form of online distillation as defined in the previous Chapter 3, and "online with controller" means training is conducted using the structure shown in Figure 7.1, employing a "controller" to dynamically control the weights between the student model and the teacher model based on the training process. While the minimal error was recorded in scenarios devoid of controller utilization, specifically through the employment of static distillation and ground truth error weights, it is conspicuously observed that the adoption of Adaptive Weights facilitated the attainment of numerous lower validation errors. Given the stochastic characteristics inherent in neural network training processes, it is reasonable to infer that the online distillation methodology incorporating Adaptive Weights consistently demonstrates superior efficacy in minimizing validation errors.

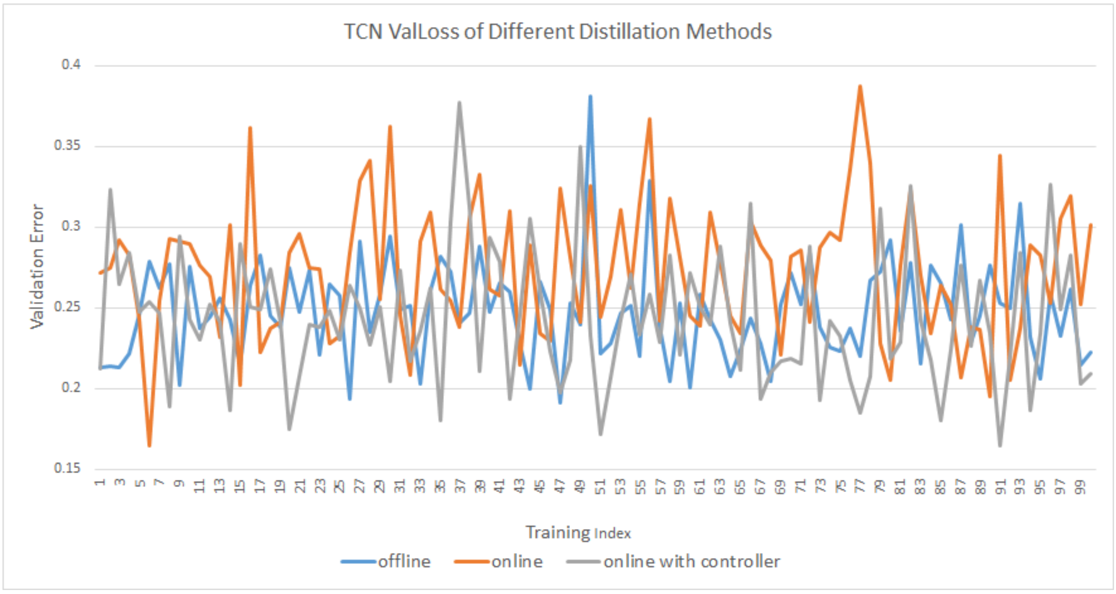


Figure 7.2: TCN ValLoss of Different Distillation Methods

Table 7.2: CPS Validation Errors of Different Settings

Setting	**Validation Error** (m^2)
TCN offline	0.191
TCN online	0.164
TCN online with Controller	0.164

The configurations for the Student and Teacher structures in the Capsule Neural Network (CPS) experiments are shown in Table 7.3.

The size of the teacher model are far greater than those of the student model, but the student model has more parameters in its final fully connected layer and bigger Kernel Size compared to the teacher model. This is to enhance the expressive capability of the student model, compensating for its smaller size to a certain extent, almost the same reason as TCN setting. In CPS networks, we stack not the

Table 7.3: CPS Network Configuration

Parameter	**Student NN**	**Teacher NN**
Routings	3	3
Dimension of capsule cps1	3	3
Dimension of capsule cps2	5	5
Number of capsules in cps1	7	12
Number of classes	3	3
Number of filters	16	16
Kernel size	3	2
Dense units	8	32

regular convolutional layers, but a type of Capsule (see Section 6.3 and Alorithm 2). Capsules are a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity, such as an object or a part of an object. The length of the activity vector is used to represent the probability that the entity exists, and its orientation represents the instantiation parameters. Active capsules at one level make predictions via transformation matrices for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher-level capsule becomes activated. Multiple capsules are stacked through specific structures and hierarchies to form a multi-layer network,In the configuration of Table 7.3,2 capsules(cps1 and cps2) are stacked in both student and teacher models, teacher model has a bigger size cps1.

The experimental results are shown in Figure 7.3 and Table 7.4 and are unexpectedly impressive."Offline" means training is conducted in the form of offline distillation, "online" means training is conducted using the form of online distillation as defined in the previous Chapter 3, and "online with controller" means training is conducted using the structure shown in Figure 7.1, employing a "controller" to dynamically control the weights between the student model and the teacher model based on the training process.At first, when I applied online distillation with fixed weights, the outcomes were so disappointing that I doubted the usefulness of online distillation for CPS networks. But, after introducing a controller, the CPS network achieved the best results. Yet, given the CPS student model has 3,814 parameters, and the TCN student has only 986 parameters, it's difficult to definitively say which model is better without first tuning the parameters.

During the training process, the most crucial parameter is the η value of the Attentive Imitation Loss mentioned in Section 3.2, as this value serves as a benchmark for assessing the quality of distilled knowledge. Throughout training, we use it to calculate a normalized weight. If the discrepancy between the current output and the ground truth is significant, it indicates that the output has low referential value as distilled knowledge for another model, and therefore, the

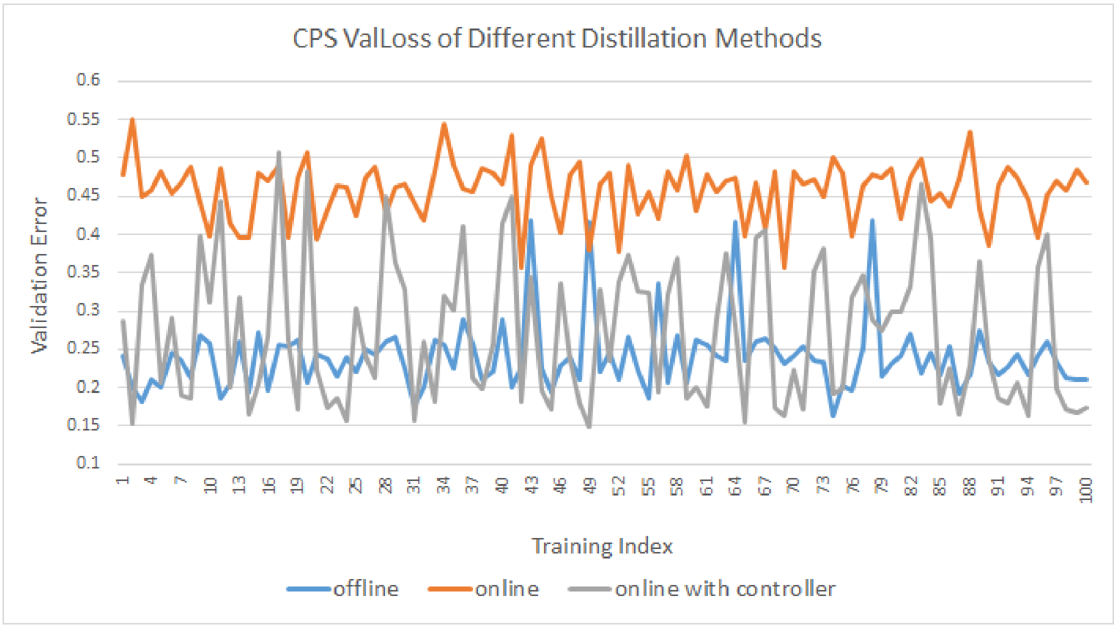


Figure 7.3: CPS Validation Loss of different Distillation Method

Table 7.4: Validation Errors of Different Settings

Setting	**Validation Error** (m^2)
CPS offline	0.162
CPS online	0.357
CPS online with Controller	0.149

distillation error should be down-weighted during parameter updates. In the offline distillation process, since the teacher model is already trained, this value can be derived from the maximum error output by the teacher model. However, in online distillation, as both teacher and student models are untrained, the relative output errors compared to the ground truth will be exceedingly large during training, making a large denominator impractical for assessing distilled knowledge. This situation renders the Attentive Imitation Loss mechanism ineffective. Consequently, I conducted extensive experiments to find an optimal η value for TCN with the results shown in Figure 7.4 and reported in Table 7.5 and for CPS with the results shown in Figure 7.5 and reported in Table 7.6.

After some straightforward tuning in η it was found that the TCN model achieved the optimal validation error of $0.124\,\mathrm{m}^2$ when η was set to 0.9. Meanwhile, the CPS model obtained its best validation error of $0.13\,\mathrm{m}^2$ at η equal to 1.2. Considering that the CPS student model has 3,814 parameters compared to the TCN student

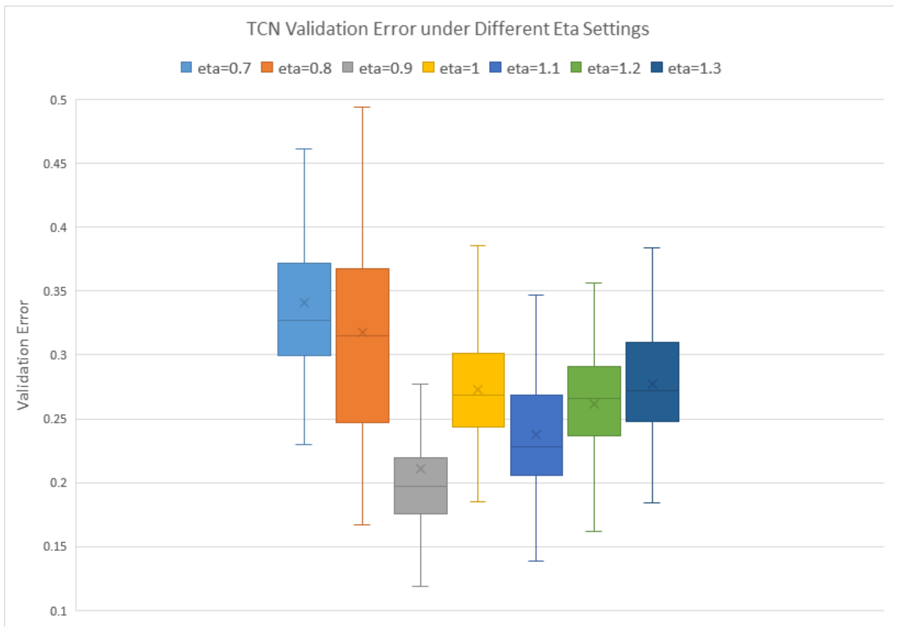


Figure 7.4: TCN Validation Error under Different η

Table 7.5: TCN Minimum Validation Errors for Various η Values

η	**Min Validation Error** (m^2)
0.7	0.229
0.8	0.167
0.9	0.124
1.0	0.185
1.1	0.139
1.2	0.153
1.3	0.184

model's 986 parameters, it can be concluded that the TCN model is a more efficient network for our scenario.

7.2 Pre-Distillation Training

In offline distillation, the teacher model is typically pre-trained, thereby ensuring the quality of distilled knowledge and numerical stability during training. In contrast, online distillation faces the challenge that both teacher and student models are untrained at the outset. Consequently, the value of distilled knowledge as a reference is relatively limited for both the student and teacher models in the initial phase of

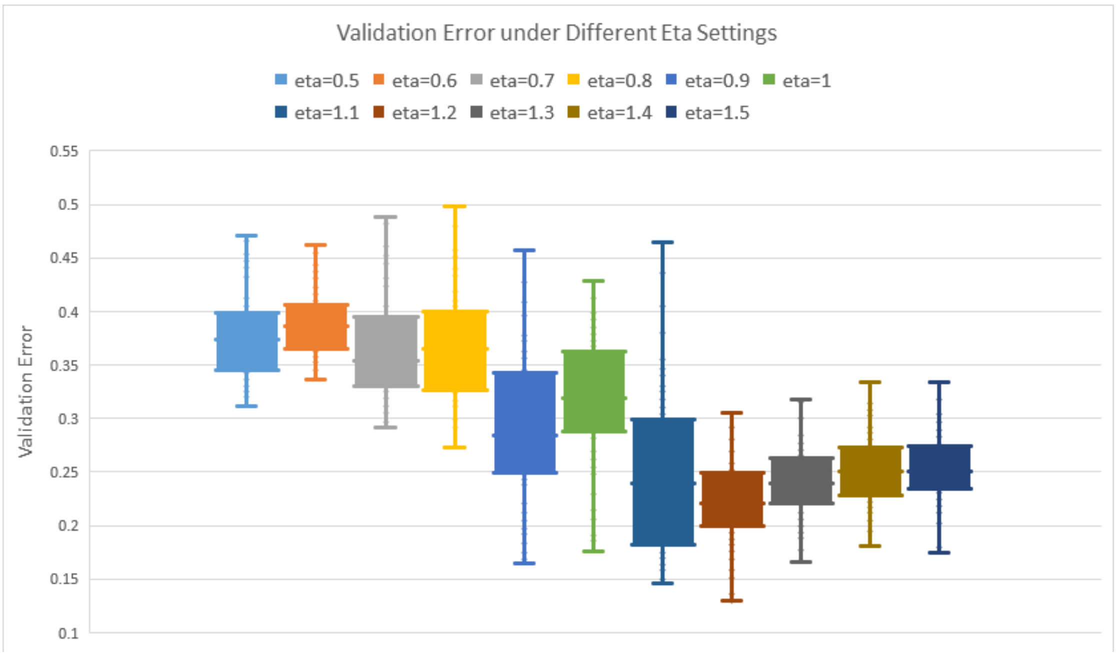


Figure 7.5: CPS Validation Error under Different η

Table 7.6: CPS Minimum Validation Errors for Various η Values

η	**Min Validation Error** (m^2)
0.7	0.292
0.8	0.272
0.9	0.165
1.0	0.158
1.1	0.146
1.2	0.130
1.3	0.165

training, and may even lead to potential misguidance.

Based on this hypothesis, I have decided to conduct two experiments. The first experiment involves training the teacher model independently for several epochs before incorporating the distilled knowledge into the loss function, to prevent the teacher model from being “misled” by the student model, while the student model continues to train as per its original scheme. The second experiment entails independent training of both the student and teacher models for several epochs before integrating distilled knowledge into their respective loss functions.

7.2.1 Teacher pre-training for 50, 100, 150, and 200 epochs

Experimental specifications for each setting:

- Train 100 times.
- Terminate a single training if the validation error does not decrease after 200 epochs.
- Maximum of 1500 epochs for each training.
- Output the best result, referring to the lowest validation error.

As shown in Figure 7.6 and Table 7.7, apart from the special case of pre-training by 100 epochs, pre-training more epochs typically leads to higher minimum validation errors. However, if we look at the test errors for models that got the lowest validation error, it seems like pre-training for 100 or 200 epochs might give models better ability to generalize. But, it's hard to make a clear conclusion from this.

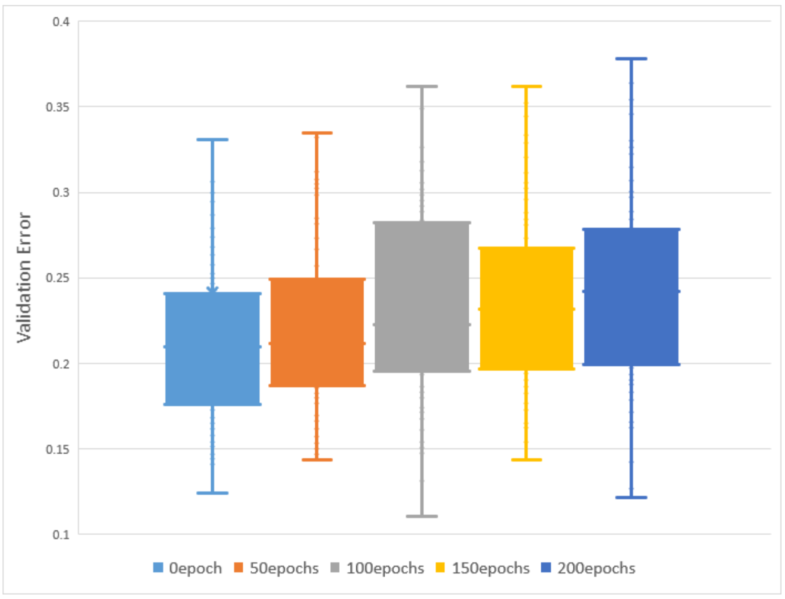


Figure 7.6: Pre-training Teacher Network for 50, 100, 150, and 200 epochs

7.2.2 Teacher and student models pre-train for 50, 100, 150, and 200 epochs

As shown in the Figure 7.7 and Table 7.7, although the results are similar to the first experiment, when both the student model and the teacher model are trained independently for several epochs before distillation knowledge is incorporated into their respective loss functions, the more epochs are trained, the worse the

Table 7.7: Minimum validation loss and the corresponding test loss from pre-training teacher model

Epochs	**Min Validation Loss** (m^2)	**Corresponding Test Loss** (m^2)
0	0.124	0.131
50	0.144	0.177
100	0.131	0.119
150	0.143	0.133
200	0.142	0.112

performance of the models. However, under the condition of training by 50 epochs, a lower validation error was achieved compared to the condition of training by 0 epochs (no independent training).

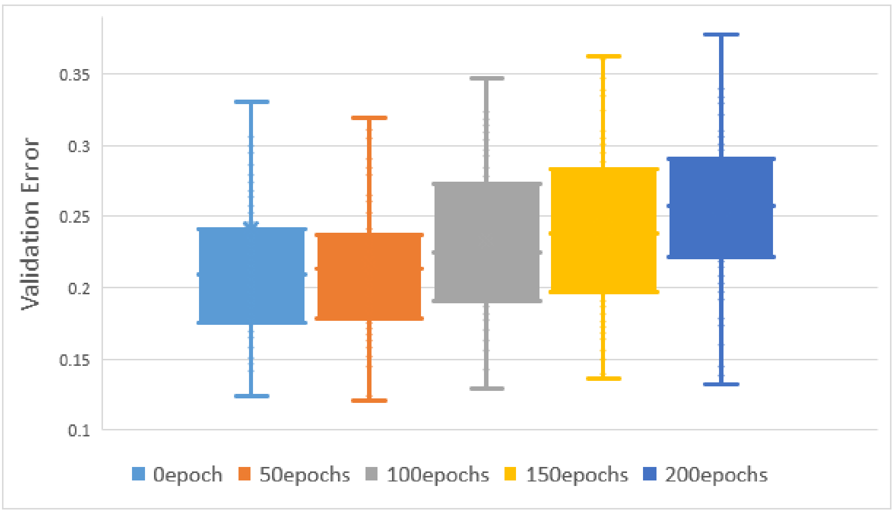


Figure 7.7: Pre-training both teacher and student network for 50, 100, 150, and 200 epochs

7.2.3 Hypothesis and Verification

Hypothesis

When both the teacher model and the student model are trained by 50 epochs, the lowest error actually decreases. Observing the errors in the training process, it is noteworthy that when the student and teacher models are first trained independently for 50 epochs, the previously mentioned problem of numerical instability during the training process is improved while maintaining the “online” nature of distillation.

Table 7.8: Minimum validation loss and the corresponding test loss from pre-training both teacher and student models

Epochs	**Minimum Validation Loss** (m^2)	**Corresponding Test Loss** (m^2)
0	0.124	0.131
50	0.121	0.127
100	0.129	0.109
150	0.137	0.137
200	0.132	0.150

Experiments for Verification

Observations from Figure 7.8 indicate that after undergoing training for about 1 epoch (64 batches), both the student and teacher model prediction errors will converge to an acceptable level ensuring that the teacher model and the student model do not mislead each other. Considering that undergoing 50 epochs of pre-training would largely affect the online "nature" of the training process, as shown in Figure 7.8, after approximately 30 batches of training, the prediction error for the majority of the student networks is within $0.6\,\mathrm{m}^2$. The teacher network, benefiting from a larger search space, converges faster, and its error will be lower compared to the student networks. The magnitude of this error is already on the same order as the smallest prediction error obtained at the end of training. Given that both the teacher model and the student model are essentially ready for distillation after about 1 epoch of training. Therefore, I decided to experiment by training the teacher and student models by 2 epochs and 5 epochs, respectively, and then compare the experimental results. Because after training for 2 epochs (128 batches), the prediction results of both the teacher and student models will be of reference value to each other.

As shown in the Figure 7.9 and Table 7.9, when both the student model and the teacher model are pre-traineld by 2 epochs before incorporating distilled knowledge into their respective loss functions, the lowest training error and corresponding test error are simultaneously achieved.

Table 7.9: Validation and corresponding test losses when pre-training for 0, 2, 5 epochs

Epochs	**Minimum Validation Loss** (m^2)	**Corresponding Test Loss** (m^2)
0	0.124	0.131
2	0.118	0.129
5	0.127	0.136

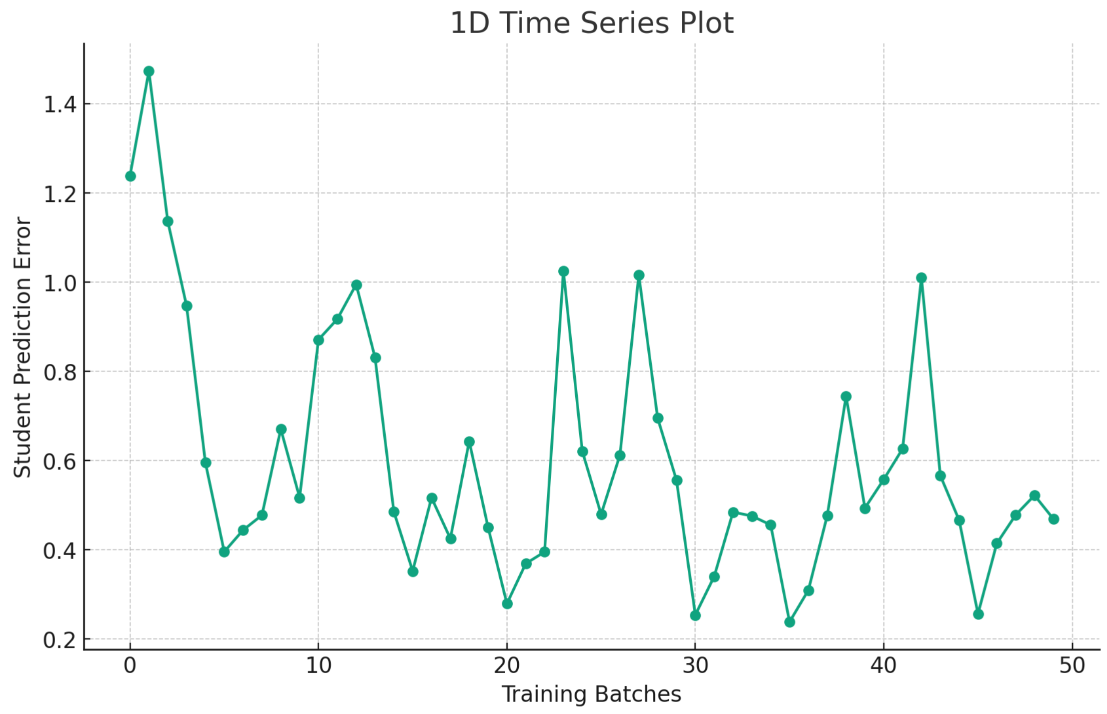


Figure 7.8: Prediction error in the first epoch

This observation leads to the conclusion that, without pre-train, the quality of distilled knowledge is very poor in the initial epochs of training, as both teacher and student models provide random inferences. This lowers much the quality of distilled knowledge and has a negative impact on the overall training, resulting in a model performance that is inferior to that achieved with a 2-epoch pre-train. On the other hand, the reason why pre-training both models by 5 epochs results in worse outcomes compared to other scenarios is that a longer pre-train may shift the training process out of reach of the best achievable using online distillation. By the time distilled knowledge is introduced into the training process, as shown in Figure 7.8, after both the teacher network and the student network undergo two epochs of training independently, their prediction errors will converge to an acceptable level, which means that the two models can start effective distillation without misleading each other. Thus the training is maintained almost entirely "online".

7.2.4 Overall Conclusion

Results From Optimized Online Training and Offline Training has shown that online distillation methods, enhanced with two novel mechanisms, Adaptive Weights and Pre-Distillation Training, indeed achieve superior validation errors in regression

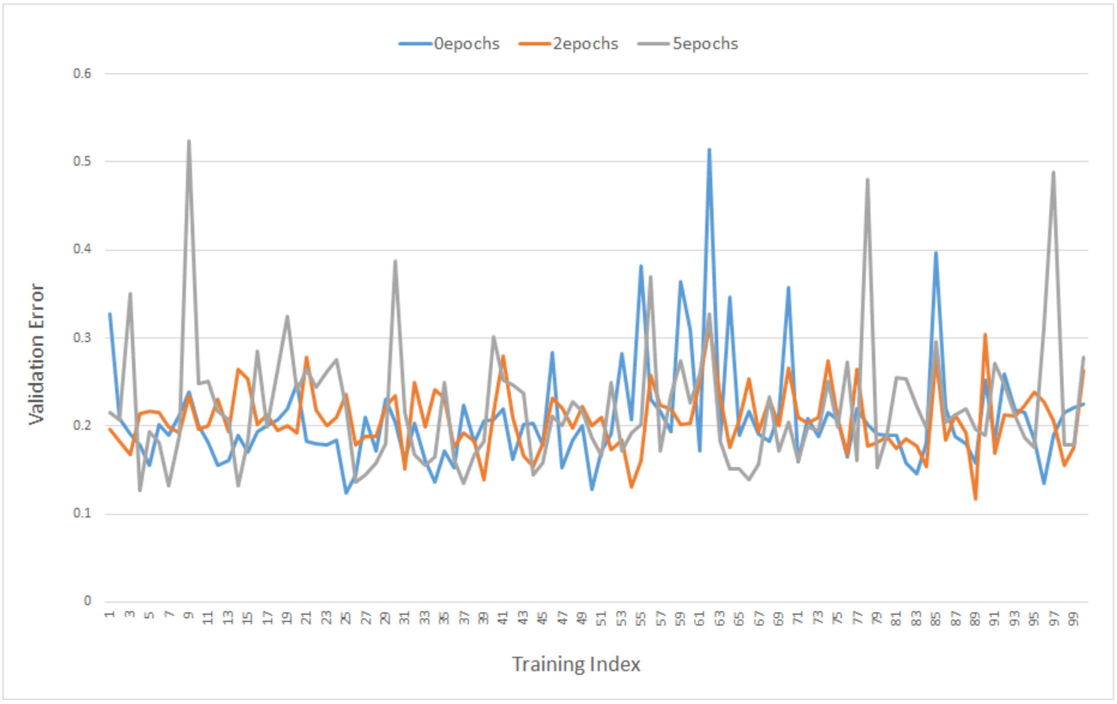


Figure 7.9: Experiments Results for Verification, pre-trained for 0, 2, 5 epochs

tasks compared to offline distillation methods, as shown in the 7.10. However, models achieving optimal validation errors through these optimized online distillation methods do not exhibit smaller test errors. Based on this, it is explored the integration of some form of domain knowledge into the distillation error that may enhance the model's generalization ability.

7.3 Domain Knowledge From Kalman Filter

The noise and the relatively low sampling frequency of 3 Hz of long-distance capacitive plate sensors and ultrasonic sensors used as ground truth, inevitably lead to the loss of some information, such as the smoothness of the dynamic of human indoor trajectories. This level of detail can hardly be captured through position information from a low sampling rate. Considering that the essence of knowledge distillation involves adding some meaningful, information-rich noise to the loss function to allow models to learn "domain knowledge," incorporating the "prior knowledge" of the smoothness of human indoor trajectories into the distilled knowledge becomes crucial.

A Kalman filter is a good option for this purpose. As shown in 7.11, the Kalman filter is used to process the output of the teacher model to some extent restore the

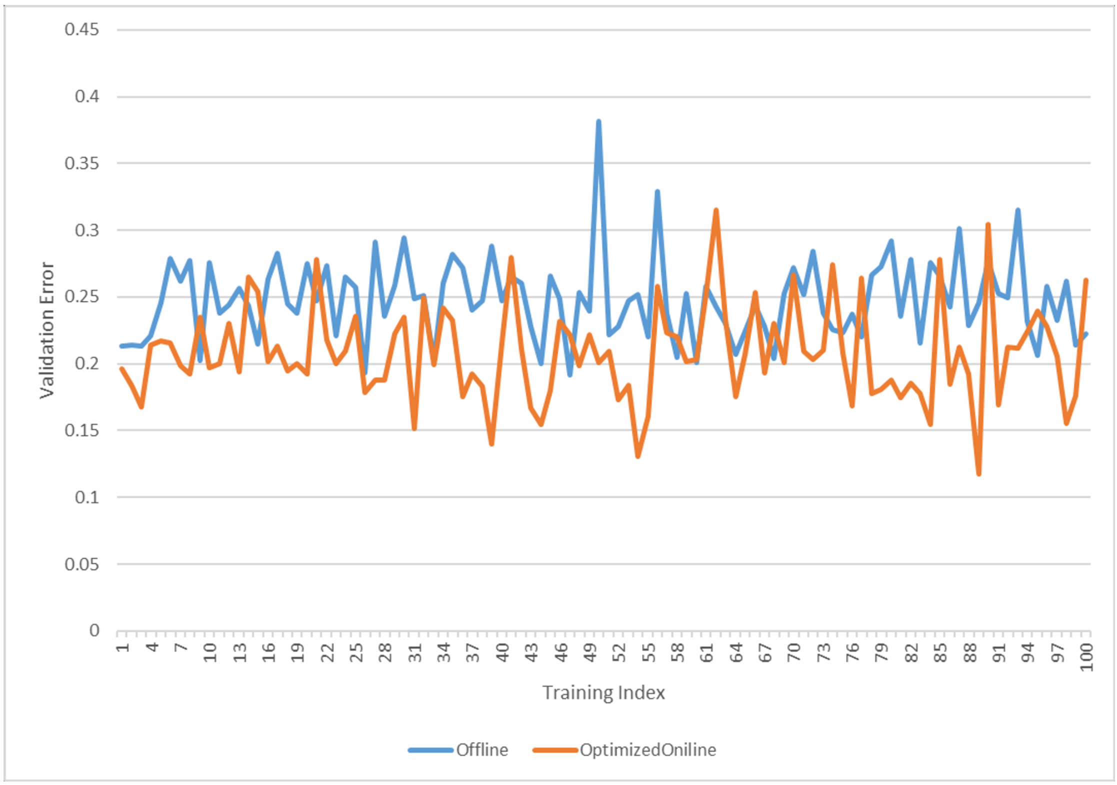


Figure 7.10: Results From Optimized OnlineTraining and Offline Training

dynamics of human indoor positioning data. This is achievable by modeling human walking dynamics as the dynamic model of the Kalman filter and treating the output of the teacher network as observation noise. Consequently, this approach “fuses” the characteristics of human movement indoors with the output of the teacher neural network, presenting it as more “complete” and “reasonable” domain knowledge to the student model. This method enhances training quality and improves the model’s generalization capabilities.

To verify whether the Kalman filter could achieve the expected effect, I conducted experiments with the results shown in Figure 7.12 and reported in Table 7.13.

To assess the efficacy of Kalman filters within the training regimen, various deployment strategies of Kalman filters were explored. Given the inherent instability in the output quality of Teacher networks at the nascent stages of training and the pivotal role these initial stages play in the model’s training trajectory, a strategic approach was adopted. This involved postponing the integration of Kalman filters by 200 epochs and subsequently ceasing their application after the completion of 200 epochs, thereby providing a nuanced examination of their impact on the training process.

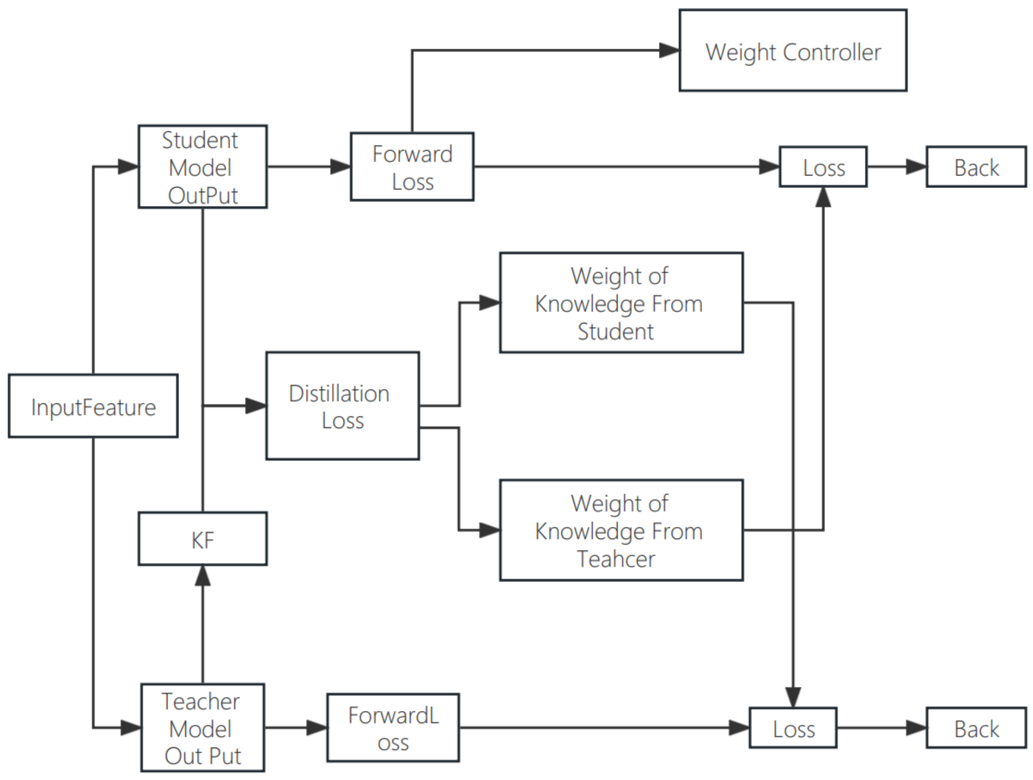


Figure 7.11: Schematic Diagram of Adaptive Weighting Framework With Kalman

Experimental specifications for each setting:

- Train 100 times.
- Terminate a single training if the validation error does not decrease after 200 epochs.
- Maximum of 1500 epochs for each training.
- Output the best result, referring to the lowest validation error.
- **WithoutKalman**: Distillation with the pure teacher prediction, without Kalman filter application.
- **KalmanLater**: Distillation with the teacher prediction, then after 200 epochs, distillation with the filtered teacher prediction using the Kalman filter.
- **KalmanFirst**: Distillation with the filtered teacher prediction using the Kalman filter, then after 200 epochs, distillation with the pure teacher prediction.
- **WithKalman**: Continuous distillation with the filtered teacher prediction using the Kalman filter.

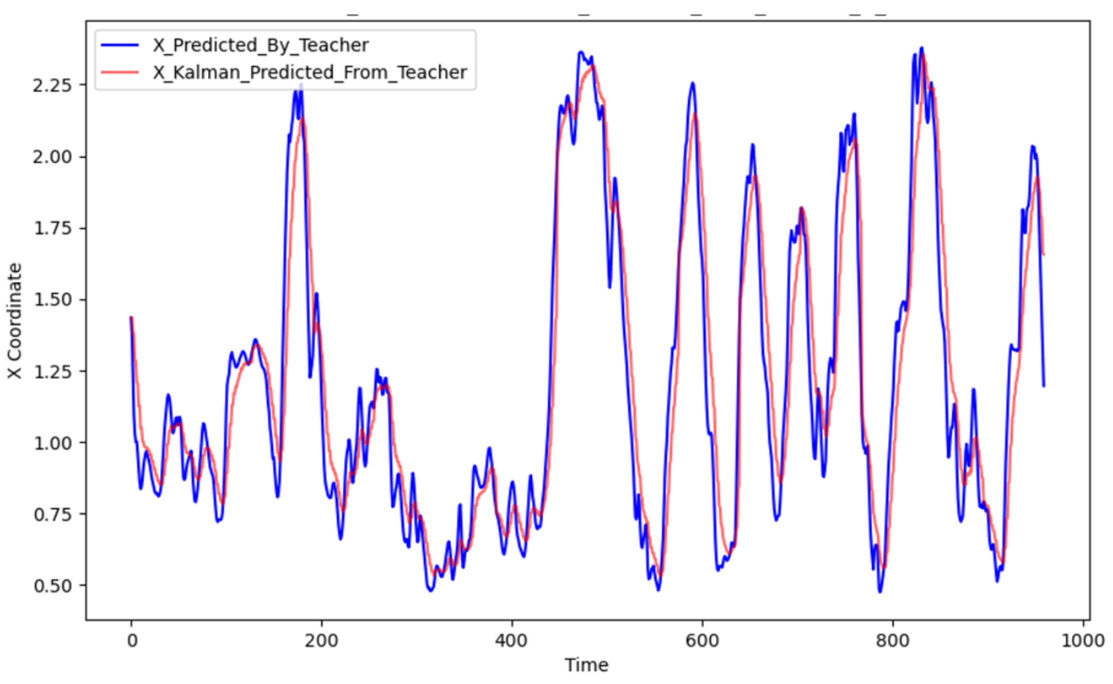


Figure 7.12: Teacher Output vs Filtered Teacher Output X Coordinates

- **Results** as show in the Table 7.10 and Figure 7.14

Table 7.10: Best Validation Error and Corresponding Test Error under Different Conditions

Condition	Best Validation Error (m^2)	Corresponding Test Error (m^2)
WithoutKalman	0.118	0.129
KalmanLater	0.149	0.140
KalmanFirst	0.148	0.109
WithKalman	0.148	0.095

The experimental results show that applying the Kalman filter to the outputs of the teacher model does not minimize the validation error, but it seems to substantially reduce the test error. This effect can be attributed to the filter's ability to prevent overfitting, thereby improving the model's ability to generalize to new data, assumption that will be tested next.

Furthermore, the Kalman filter was proven to effectively smooth the outputs of the network, fulfilling its intended role. It successfully removed high-frequency noise, which was inconsistent with the characteristics of human walking, confirming the initial expectations.

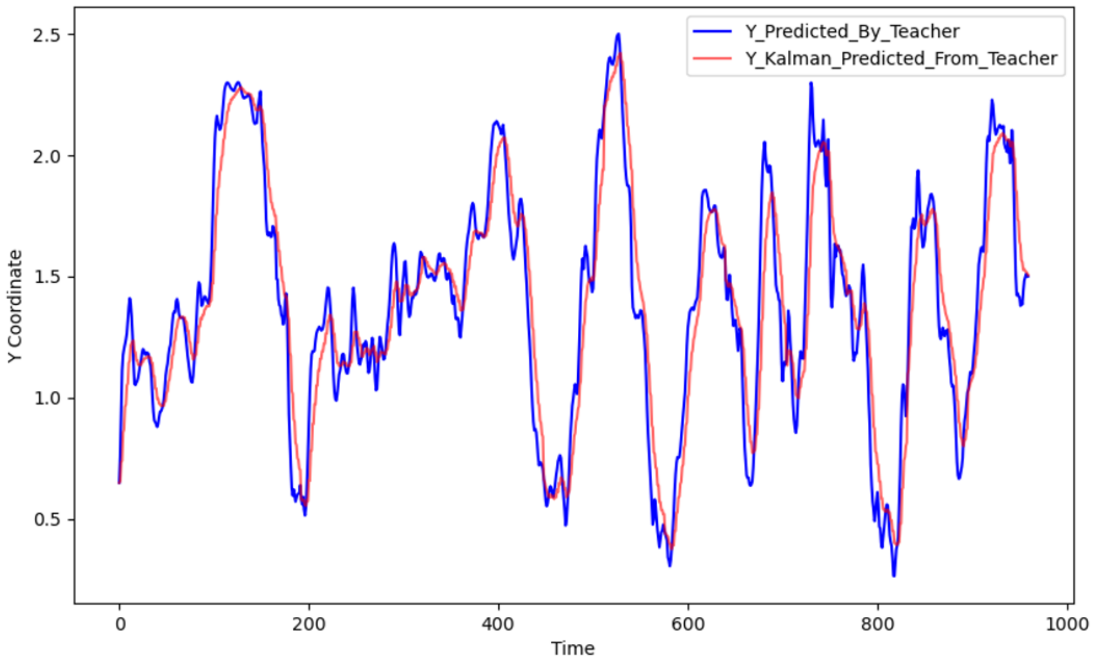


Figure 7.13: Teacher Output vs Filtered Teacher Output Y Coordinates

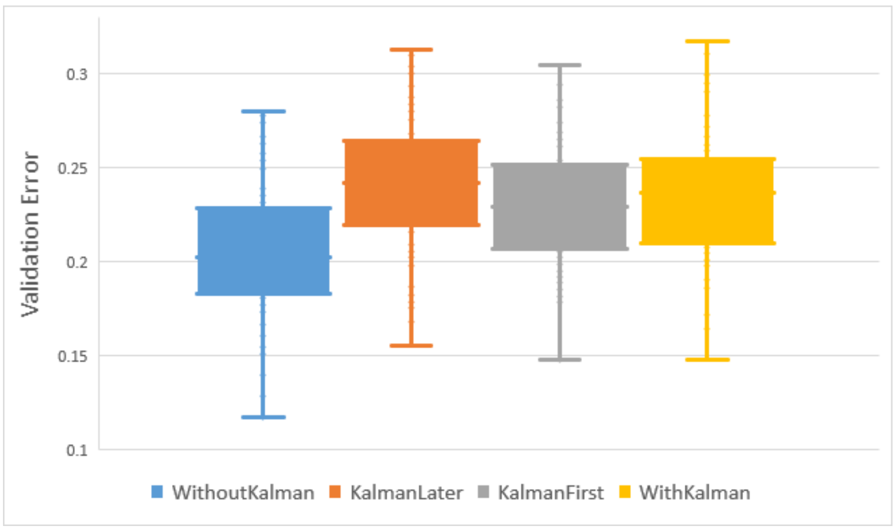


Figure 7.14: Results under different Kalman filter setting

Chapter 8

Generalization Results

Through the exploration and research conducted in Chapter 7, by applying different optimization strategies, new training logic, and structures within the online distillation process, it is observed that the performance of the student model, as measured by validation error, has indeed improved significantly. However, this improvement is not reflected in the training error.

Therefore, to validate the effectiveness of online distillation relative to offline distillation, I test the models optimized in Chapter 7 on two entirely new test sets, collected in similar experiments but in different days and environmental conditions, to evaluate their generalization performance. This approach provides a more effective and robust assessment.

Experimental specifications for each setting:

- Train 100 times.
- Terminate a single training if the validation error does not decrease after 200 epochs.
- Maximum of 1500 epochs for each training.
- Output the best result, referring to the lowest validation error.
- **OnlineWithoutKalman**: Distillation with the pure teacher prediction, without Kalman filter application.
- **OnlineKalmanLater**: Distillation with the teacher prediction, then after 200 epochs, distillation with the filtered teacher prediction using the Kalman filter.
- **OnlineKalmanFirst**: Distillation with the filtered teacher prediction using the Kalman filter, then after 200 epochs, distillation with the pure teacher prediction.
- **OnlineWithKalman**: Continuous distillation with the filtered teacher prediction using the Kalman filter.
- **Offline**: Original Offline Distillation Method.
- **CapEXP2, CapEXP3**: Two new Test Sets.

Among all evaluated models for which the results are reported in Table 8.1, and shown in Figure 8.1 and Figure 8.2, "OnlineKalmanGoLater" stands out for its notable stability across various datasets and validation metrics. While it may not always achieve the best performance in every single instance, its consistent results across different datasets indicate a superior generalization capability. This makes practical sense because the purpose of incorporating the Kalman filter into the distillation process is to improve the model's understanding of the dynamic aspects of potential physical phenomena, and introducing the Kalman filter in the later stages of distillation optimizes this improvement.

Table 8.1: Test error (MSE) for different distillation methods on new test sets

Configuration	**Data Set**			
	CapEXP2		**CapEXP3**	
	MSE (m^2)	**ΔMSE** (%)	**MSE** (m^2)	**ΔMSE** (%)
Offline (baseline)	0.520	N/A	0.461	N/A
WithoutKalman	0.433	−16.7	0.595	29.1
OnlineWithKalman	0.549	5.58	0.488	5.86
OnlinKalmanGoFirst	0.460	−11.5	0.388	−15.8
OnlinKalmanGoLater	0.416	−20.0	0.420	−8.89

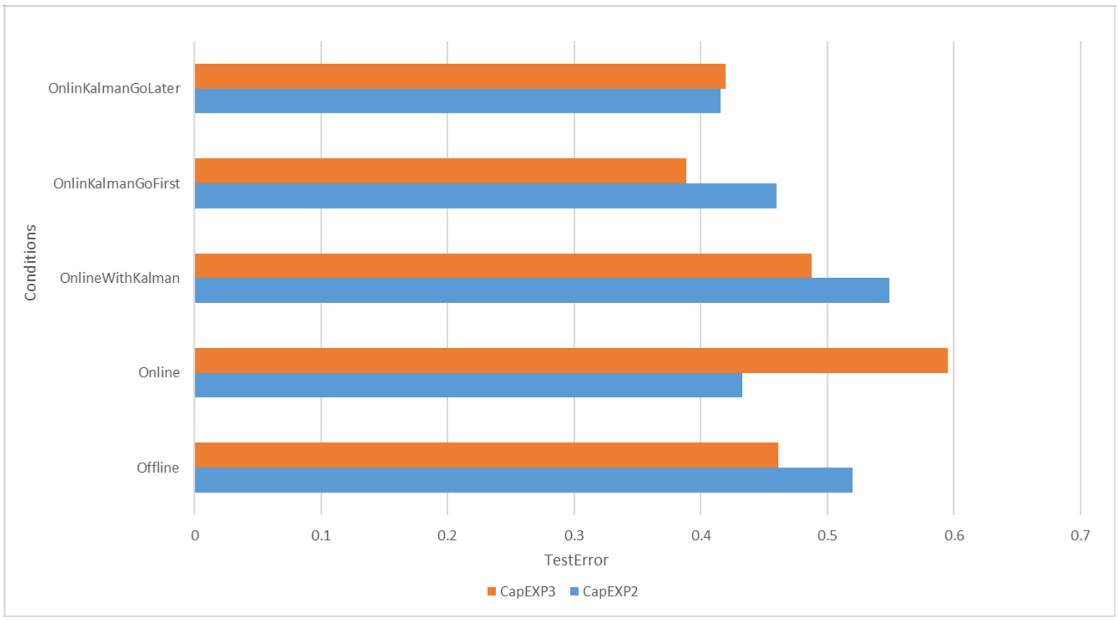


Figure 8.1: Test error (MSE) from different distillation methods on new test sets

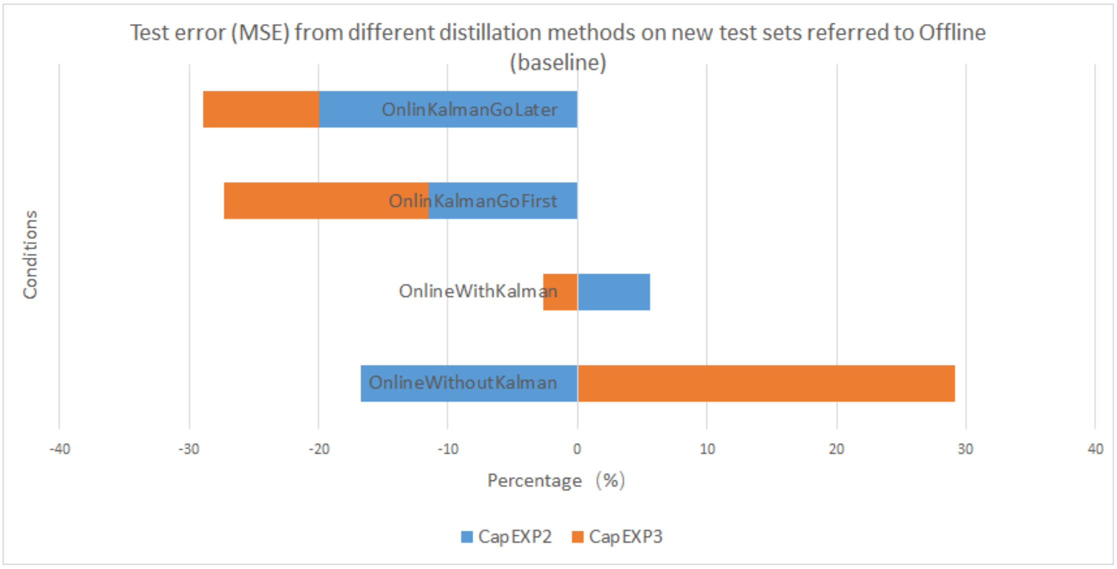


Figure 8.2: Improvement percentage in test error (MSE) referred to the offline setting

This approach is expected to improve the quality of the model's learning outcomes. Such evidence supports the argument that an optimized online distillation method can enhance model generalization performance.

Chapter 9

Conclusion

In the context of using long-range capacitive plate sensors for indoor positioning, this work discusses the application of mutual learning in online knowledge distillation. This method aims to achieve higher training quality and model performance than offline distillation by allowing real-time mutual learning between student and teacher networks through mutual distillation. The thesis explores mechanisms and new methods for improving training quality and model performance, namely adaptive weights, pre-distillation training, and the application of a Kalman filter, to optimize the online distillation process and its generalization capabilities.

The research demonstrates that through adaptive weights, the weight of distillation knowledge in the loss functions of the student and teacher networks can be dynamically adjusted during the training phase based on the progress of the student network's learning. This adaptive approach facilitated more effective knowledge transfer, significantly improving the validation error of the student network.

Furthermore, the thesis introduces pre-distillation training, where preliminary independent training of the teacher (and in some cases, also the student) models is conducted before the mutual distillation phase. This strategy is aimed at stabilizing and improving the quality of distillation knowledge early in the training process, thus avoiding the instability caused by untrained models that could lead to misleading guidance.

Additionally, the application of the Kalman filter to process the output of the teacher model was studied, aimed at restoring the "smoothness" of human indoor movement that is often lost due to low sampling rates and sensor imperfections. This approach was expected to enrich distillation knowledge with more accurate and meaningful information, thereby enhancing the learning outcomes and generalization capabilities of the student model.

These strategies and methods, when combined in the context of online distillation, demonstrated improved model performance on new test sets compared to traditional online distillation methods. The integration of adaptive weights, pre-distillation training, and the Kalman filter showcased the potential to refine and optimize the knowledge distillation process in complex applications such as indoor positioning systems.

In our current exploration and research, we have only incorporated some information about human motion patterns into knowledge distillation through a kinematic model based on Kalman filtering. This represents a very preliminary and simple exploration. However, even through such exploratory methods, we have obtained models that perform better on new datasets. Perhaps in future research, we can seek to explore new methods that include real domain knowledge to achieve better model performance.

Bibliography

[1] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. «A Survey of Indoor Localization Systems and Technologies». In: *IEEE Communications Surveys and Tutorials* 21.3 (2019), pp. 2568–2599 (cit. on p. ii).

[2] Tero Kivimäki, Timo Vuorela, Pekka Peltola, and Jukka Vanhala. «A Review on Device-Free Passive Indoor Positioning Methods». In: *International Journal of Smart Home* 8 (2014), pp. 71–94 (cit. on p. ii).

[3] Javed Iqbal, Mihai Teodor Lazarescu, Osama Bin Tariq, and Luciano Lavagno. «Long range, high sensitivity, low noise capacitive sensor for tagless indoor human localization». In: *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE. 2017, pp. 189–194 (cit. on pp. ii, 21).

[4] Andreas Braun, Reiner Wichert, Arjan Kuijper, and Dieter W. Fellner. «Capacitive proximity sensing in smart environments». In: *J. Ambient Intell. Smart Environ.* 7 (2015), pp. 483–510 (cit. on p. ii).

[5] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. «Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction». In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM. 2017, pp. 3293–3315 (cit. on p. ii).

[6] Javed Iqbal, Arslan Arif, Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno. «A contactless sensor for human body identification using RF absorption signatures». In: *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE. 2017, pp. 1–6 (cit. on p. ii).

[7] Javed Iqbal, Mihai Teodor Lazarescu, Osama Bin Tariq, Arslan Arif, and Luciano Lavagno. «Capacitive Sensor for Tagless Remote Human Identification Using Body Frequency Absorption Signatures». In: *IEEE Transactions on Instrumentation and Measurement* 67.4 (2018), pp. 789–797 (cit. on p. ii).

[8] Atika Arshad, Sheroz Khan, A. H. M. Zahirul Alam, Rumana Tasnim, Teddy S. Gunawan, Robiah Ahmad, and Chandrasekharan Nataraj. «An activity monitoring system for senior citizens living independently using capacitive sensing technique». In: *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. IEEE. 2016, pp. 1–6 (cit. on p. ii).

[9] Alireza Ramezani Akhmareh, Mihai Teodor Lazarescu, Osama Bin Tariq, and Luciano Lavagno. «A Tagless Indoor Localization System Based on Capacitive Sensing Technology». In: *Sensors* 16.9 (2016), p. 1448 (cit. on p. ii).

[10] Javed Iqbal, Mihai Teodor Lazarescu, Osama Bin Tariq, and Luciano Lavagno. «Long range, high sensitivity, low noise capacitive sensor for tagless indoor human localization». In: *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE. 2017, pp. 189–194 (cit. on p. ii).

[11] Javed Iqbal, Mihai Teodor Lazarescu, Arslan Arif, and Luciano Lavagno. «High sensitivity, low noise front-end for long range capacitive sensors for tagless indoor human localization». In: *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*. IEEE. 2017, pp. 1–6 (cit. on p. ii).

[12] Raphael Wimmer, Matthias Kranz, Sebastian Boring, and Albrecht Schmidt. «A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition». In: *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*. IEEE. 2007, pp. 171–180 (cit. on p. ii).

[13] Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno. «Neural Networks for Indoor Human Activity Reconstructions». In: *IEEE Sensors Journal* 20.22 (2020), pp. 13571–13584 (cit. on pp. ii, 22, 23, 26).

[14] Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno. «Neural network-based indoor tag-less localization using capacitive sensors». In: *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*. ACM. 2019, pp. 9–12 (cit. on p. ii).

[15] Ying Zhang, Tao Xiang, Huchuan Lu, and Timothy M. Hospedales. «Deep Mutual Learning». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR. 2018, pp. 4320–4328 (cit. on pp. 2, 15, 16, 18–20).

[16] Ethem Alpaydin. *Introduction to Machine Learning*. Cambridge: MIT press, 2020 (cit. on p. 3).

[17] Matiur Rahman Minar and Jibon Naher. «Recent Advances in Deep Learning: An Overview». In: *ArXiv* abs/1807.08169 (2018). URL: `https://api.semanticscholar.org/CorpusID:49908818` (cit. on p. 3).

[18] Thomas Epelbaum. «Deep learning: Technical introduction». In: *ArXiv* abs/1709.01412 (2017). URL: `https://api.semanticscholar.org/CorpusID:43867975` (cit. on p. 4).

[19] Vinod Nair and Geoffrey E. Hinton. «Rectified Linear Units Improve Restricted Boltzmann Machines». In: *International Conference on Machine Learning*. IMLS. 2010 (cit. on p. 5).

[20] Yuandong Tian. «Student Specialization in Deep ReLU Networks With Finite Width and Input Dimension». In: *arXiv: Learning* (2019). URL: `https://api.semanticscholar.org/CorpusID:207863698` (cit. on p. 5).

[21] David E. Caughlin. *Chapter 48 applying K-fold cross-validation to logistic regression: R for HR: An introduction to human resource analytics using R.* URL: `https://rforhr.com/kfold.html` (cit. on p. 9).

[22] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. «Knowledge distillation: A survey». In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819 (cit. on pp. 10–13).

[23] Muhamad Risqi U. Saputra, Pedro PB De Gusmao, Yasin Almalioglu, Andrew Markham, and Niki Trigoni. «Distilling knowledge from a deep pose regressor network». In: *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*. ICCV. 2019, pp. 263–272 (cit. on p. 14).

[24] Faheem Ijaz, Hee Kwon Yang, Arbab Waheed Ahmad, and Chankil Lee. «Indoor positioning: A review of indoor ultrasonic positioning systems». In: *2013 15th International Conference on Advanced Communications Technology (ICACT)*. IEEE. 2013, pp. 1146–1150 (cit. on p. 24).

[25] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. «Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting». In: *Advances in Neural Information Processing Systems (NIPS)*. NIPS. 2015, p. 28 (cit. on pp. 27, 28).

[26] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. «Dynamic Routing Between Capsules». In: *Advances in Neural Information Processing Systems*. NIPS. 2017, p. 30 (cit. on pp. 31, 32).

[27] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. «An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling». In: *ArXiv* abs/1803.01271 (2018). URL: `https://api.semanticscholar.org/CorpusID:4747877` (cit. on pp. 35, 36).