Chapter 5

Light-weight knowledge distillation deep learning approach to identify river water pollution in Internet of Things

In this chapter, we detail the development of a complex deep neural network model designed for analyzing a sensory dataset containing around 100,000 annotated data samples to determine river water pollution level. Our primary objective is to use this model to classify water data and discern various levels of river pollution. This fully trained deep learning model, referred to as the Teacher Model, is, however, unsuitable for deployment on resource-constrained IoT devices. To address this limitation, we introduce a technique known as filter-level pruning, which helps us obtain a more compact model that retains essential information. Additionally, we describe a knowledge distillation approach, wherein we train a smaller model using guidance from the larger Teacher Model. This allows us to create a compact model while preserving crucial insights from the original model. Furthermore, we successfully deploy this compact

model on IoT devices with limited processing capabilities. Despite these constraints, we achieve a high level of accuracy in detecting river water pollution levels, making it a valuable solution for practical applications in environmental monitoring.

5.1 Introduction

Rivers function as a vital resource for human activities and habitat for a host of non-human animals and plants. Polluted water spreads various bacterial, viral, and parasitic diseases. Regularly analyzing the status of river water pollution helps to improve water quality. However, regular visit for water pollution monitoring is a tedious task, especially in the areas with large river basins. In addition, it requires reliable and expensive water quality analysis equipment and trained human resources [73]. Rivers are significant sources of drinking water. Continuous monitoring is required to keep the river water clean; otherwise, drinking water will be scarce.

The rapid development of the Internet of Things (IoT) has the potential to connect a large number of low-cost, small size, and low-power devices [74, 75]. IoT devices equipped with sensors monitor the environment and transfer the data to the remote host. An IoT device first collects and processes a large amount of water quality sensory data at a given collection point. Next, the processed sensory data is quickly and conveniently transmitted to the remote host. The fundamental problem of an IoT is its limited energy, storage, and processing power.

Water Quality Index (WQI) is a means of transforming large quantities of water quality data into a single unified number that quantifies the river water quality [58]. Deep Neural Network (DNN) finds the IoT applications to estimate the WQI and assigns the class labels to the given water sample. However, DNN model needs millions of parameters for determining a class label. Therefore, it is impractical to deploy the DNN model on IoT devices having limited energy supply and processing capacity. The existing work solves this problem by adopting DNN compression techniques [76]. Though

Figure 5.1: Illustration of a river water pollution monitoring using LoRa network in IoT.

the existing DNN compression techniques significantly reduce the power consumption and processing requirements but suffer from accuracy compromise. Knowledge distillation is a concept that improves the performance of the compressed DNN by using the generalization ability of the regular DNN [10, 77]. Fig. 5.1 illustrates a LoRa-based IoT system for river water pollution monitoring, where a LoRa node (LN) acquires the water pollution monitoring sensory data (*i.e.*, including pH, electrical conductivity, dissolved oxygen, turbidity), process the sensory data at the LN by the compressed DNN, forward the data to the LoRa Gateway (LG), and finally to the Network Server (NS).

5.1.1 Motivation

Previous studies have following major limitations which motivated this work:

- Authors in [44] proposed a cost-effective remote sensing mechanism for monitoring and analyzing the quality of river water. They have used a neural network for analyzing different parameters of river water. The authors claim to solve the hurdle of monitoring the quality of river water in the logistically difficult area.
- In [45] authors used the neural network for predicting the quality of river water. They also developed a mechanism of artificially creating the data values for those

instances that are not recorded during data collection.

- Zhang *et. al.* [46] utilized nonlinear autoregressive neural network to predict the concentration of bacteria named Escherichia coli near the beach site. Through experimental analysis, they claim that the use of nonlinear autoregressive neural network outperforms the existing approach for predicting bacterial concentration.
- Authors in [47] proposed a neural network-based model for capturing the dynamical variation in the chlorophyll value of the water. They defined an optimization problem for cost-effective on sight processing and enhancing the accuracy of prediction.
- Taipalmaa *et. al.* [48] presented a mechanism for analyzing river water quality using segmentation of water surface images captured using a drone camera. They utilized temporal and spatial features of the deep learning model to classify the segmented image and determine the level of pollution in the river water. None of the existing work have simultaneously considered accuracy and smart sensing-based approach as its primary objective for detecting river water pollution level.

In this chapter, we address the problem: *How to estimate river water pollution level (estimation of WQI) using limited processing of IoT devices with acceptable accuracy?* To solve this problem, this research introduces a River Water Pollution Monitoring system based on deep neural networks. Initially, the system constructs an untrained, intricate deep neural network model. Subsequently, we train this elaborate model using a dataset comprising 100,000 annotated sensory data instances, resulting in a fully trained complex DNN model. Further, we employ filter-level pruning to compress the trained complex model. Next, we utilize a knowledge distillation deep learning technique to train the compressed model, transferring valuable insights from the larger model. This refined compact DNN model is then deployed on IoT devices and used to identify river water pollution levels effectively.

5.1.2 Major Contributions:

To the best of our knowledge, this is the first work to address the use of distilled knowledge from the cumbersome deep neural network to train the compressed deep neural network for river water pollution monitoring problem using IoT devices. We use a LN as an IoT device which consists of sensing, processing, and communication units. The sensing unit consists various sensors for monitoring the water quality parameters. The LG relays the data from LN to the remote host. Our major contributions are as follows:

- We design a cumbersome DNN (Teacher Model) and we train the cumbersome model to detect the river water pollution level.
- Estimation of WQI: We use filter-level pruning to obtain the untrained compressed DNN model (Student Model) for estimating the WQI of river water.
- We use knowledge distillation technique [10] to train the compressed DNN. The compressed DNN is successfully deployed on the LN (IoT devices), which requires limited resources (CPU, energy, and storage) and estimates the WQI with an acceptable accuracy.
- **Experimental Results:** Finally, we conduct various experiments on an open-source dataset to verify the effectiveness of built system.

The rest of chapter is organized as follows. Section 5.2 describes the preliminaries. Next, Section 5.3 presents knowledge distillation deep learning model. Further, the experimental evaluations are presented in Section 5.4. Finally, the chapter is concluded in Section 5.5.

5.2 Preliminaries and problem statement

This section describes the different terminologies used in this work. Later, this section covers a brief description of the problem associated with water pollution level monitoring

in river and the overview of solution.

5.2.1 Preliminary

Here, in this section, we focus on the river water pollution monitoring using IoT, where an IoT device acquires sensory data (i.e., including pH, electrical conductivity, dissolved oxygen, turbidity), processes sensory data to estimate WQI to detect river water pollution level. The objective of this research work is to estimate the WQI with high accuracy.

Definition 5.1 (Knowledge distillation) *In deep neural network, the process of transferring the generalization ability of a cumbersome model to a compact model for improving its recognition performance is called knowledge distillation. Here, the compact model mimics the cumbersome model and achieves acceptable accuracy by utilizing limited resources [10].*

Definition 5.2 (Logits) *The logits in DNN are the feature vectors one layer prior to the softmax or last layer of the network. In other words, logits are also referred to as the unnormalized predictions of a DNN.*

5.2.2 Problem statement and overview of solution

Monitoring river water pollution through a range of sensors proves valuable in gauging the extent of contamination. The data on water pollution aids in implementing proactive measures to curb the river water's contamination, as outlined in the introduction. Given the impracticality of employing energy-efficient and low-processing devices, the recognition approach must accurately identify river water pollution levels or estimate Water Quality Index (WQI) even when devices have limited energy, storage, and processing capabilities. This research focuses on addressing the challenge of assessing river water pollution levels using IoT devices that have limited resources, including constraints in energy, storage, and processing, all while ensuring a high level of accuracy.

Overview of the solution: This work proposes a deep neural network model to monitor river water pollution level. First, the system develops the cumbersome DNN. We train the cumbersome DNN model and finds the river water pollution level with high accuracy. We can not deploy the big model on IoT devices. Therefore, we obtain the compressed model using filter-level pruning so that we can easily deploy the compressed model on IoT devices. A highly compressed DNN model during estimation of WQI undergoes higher accuracy compromise. Therefore, we use knowledge distillation technique to train the compressed model under the guidance of big model and achieves the acceptable accuracy.

5.3 Knowledge distillation deep learning approach to estimate river water quality index

Knowledge distillation is commonly used in scenarios where you have a large, accurate model (teacher) and you want to create a smaller, faster model (student) that performs almost as well as the teacher model. The propose DNN system envisions each LN functioning as an IoT device with sensing, processing, and communication units. The sensing unit integrates various sensors to monitor water quality parameters. The system constructs a cumbersome DNN, denoted by $\mathbf{M}^t$, that can achieve significantly higher recognition performance using sensory dataset $\mathcal{D}$. Next, we use filter-level pruning DNN compression technique and obtain an untrained compressed DNN, denoted by $\mathbf{M}^s$, which can be easily deployed on LNs. Further, data reduction is performed to obtain $\mathcal{D}'$ from $\mathcal{D}$ upon which DNN $\mathbf{M}^s$ is trained under the supervision of model $\mathbf{M}^t$ using knowledge distillation technique. Finally, a classifier ($\mathbf{\Pi}$) is built that can obtain WQI of river water. The processing unit collects the sensory data and estimates a WQI using the classifier $\mathbf{\Pi}$. Fig. 5.2 illustrates The block diagram for the estimation of WQI.

Let $\mathbf{M}^t$ and $\mathbf{M}^s$ denote the cumbersome DNN and compressed DNN, respectively. The $\mathbf{M}^t$ and $\mathbf{M}^t$ are called as teacher and student models in knowledge distillation,

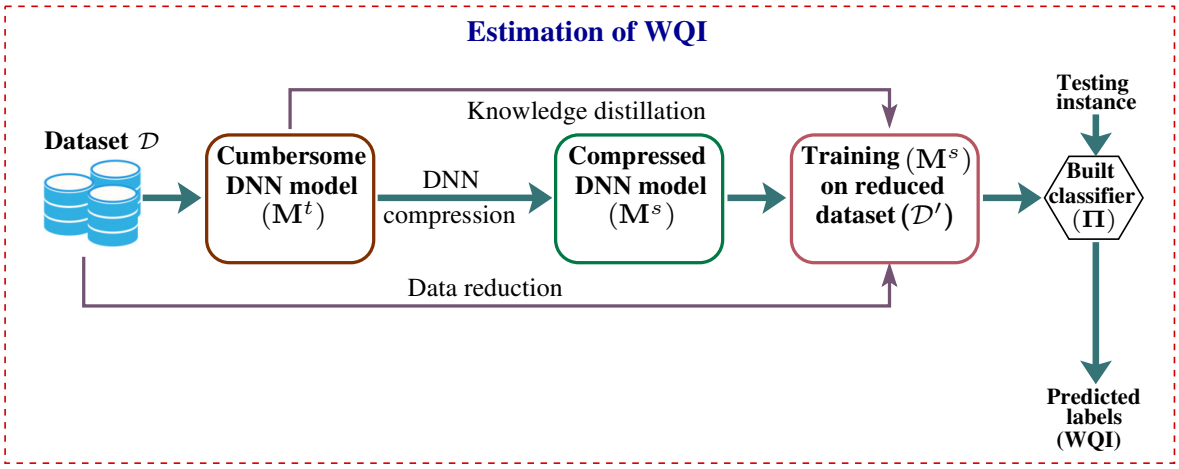


Figure 5.2: Block diagram for the estimation of WQI.

respectively. Let $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ represents the dataset having n number of instances, where, $\mathbf{x}_i$ is the i^{th} vector instance having class label y_i. If there are p sensors, each having q axes that are used for recording $\mathcal{D}$ at a sampling rate λ, then lenght of i^{th} data instance $|\mathbf{x}_i|$ is calculated as, $|\mathbf{x}_i| = \lambda \times p \times q$. For example, if we record a dataset at the sampling rate of 100 Hz using tri-axial sensor, then the length of data instance is $100 \times 3 \times 3 = 900$ per second. The $\mathbf{M}^t$ is trained on dataset $\mathcal{D}$, whereas, for $\mathbf{M}^s$, we utilize the dataset $\mathcal{D}' = \{\mathbf{x}_i', y_i\}_{i=1}^n$ with reduced length of instances. Further, to obtain $\mathbf{x}_i'$ from $\mathbf{x}_i$, such that $|\mathbf{x}_i'| < |\mathbf{x}_i|$, we incorporate following two techniques: i) Despite using the value of all q axes, use average value, then $|\mathbf{x}_i'| = \lambda \times p \times 1$; and ii) Identify the most distinguishable axis from q axes and only use value of that axis. Here, also we obtain $|\mathbf{x}_i'| = \lambda \times p \times 1$.

5.3.1 Construction of cumbersome DNN model (Teacher model)

Teacher model is a well-trained, often complex and accurate model, which serves as the source of knowledge. This teacher model has learned a lot about the task at hand and can make predictions with high accuracy. The teacher model $\mathbf{M}^t$ is a high configuration model that incorporates a large number of parameters. $\mathbf{M}^t$ achieves significantly higher accuracy but requires substantial computational energy and storage.

Therefore, deployment of model $\mathbf{M}^t$ on low-processing IoT devices becomes prohibitive. The model $\mathbf{M}^t$ extracts features from dataset $\mathcal{D}$ using the combination of convolutional, LSTM and Fully Connected (FC) layers. The model first computes spatial and temporal features and then combines them with hand-crafted features to obtain final features (denoted as $\mathbf{F}$). Next, the model $\mathbf{M}^t$ learns a mapping between instances of feature $\mathbf{F}$ and its corresponding labels. Finally, the model builds a classifier $\mathbf{\Pi}^t$ that predicts testing label y_{te} against testing instance $\mathbf{x}_{te}$.

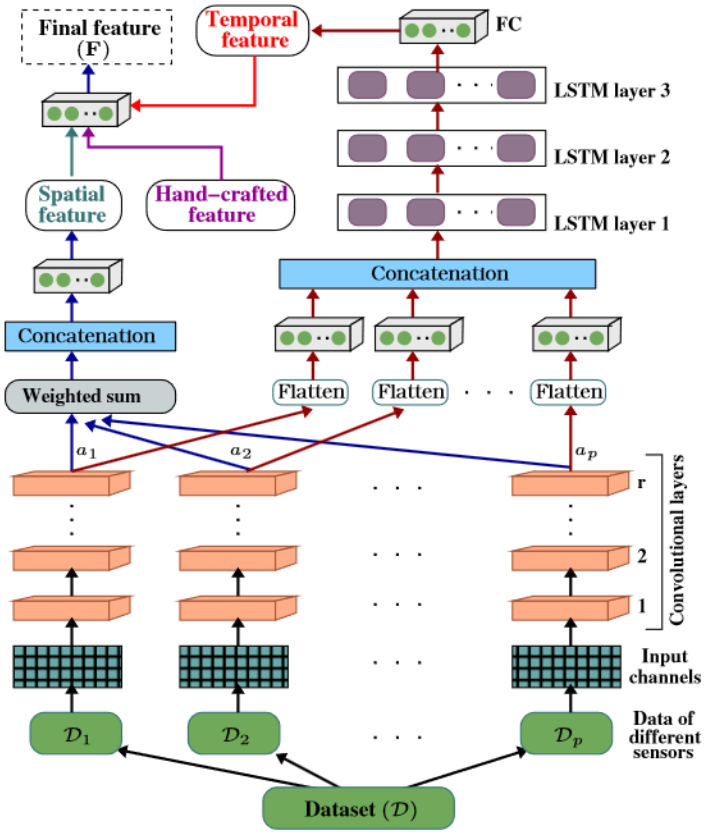


Figure 5.3: Teacher model ($\mathbf{M}^t$) for estimating the WQI using combination of spatial, temporal, and hand-crafted features.

The dataset $\mathcal{D}$ consist of data from p sensors. The teacher model first divide $\mathcal{D}$ into p sub-datasets, denoted as $\mathcal{D}_j, 1 \leq j \leq p$. Since each sub dataset $\mathcal{D}_j$ consists of the value of multiaxial sensors, these values are channelized prior to convolutional operations. The channelized sub-dataset associated with p sensors is passed as input to r convolutional layers. The convolution output is supplied as input to two different

operations. First operation is inspired by the concept of deep learning model proposed in [78], we use the weighted sum, $a_1, a_2, \cdots, a_q$ of convolutional operations of p sensors, $1, 2, \cdots p$, respectively. The weighted output is concatenated and passed through a FC layer to obtain spatial features. In the second operation, the convolutional outputs are separately supplied to flatten layers followed by FC layer to generate aggregated output using concatenation layer. The aggregated out is further passes through three LSTM layers followed by a FC layer to generate temporal features. Finally, the obtained spatial and temporal features are combined with hand-crafted features (*i.e.,* minimum, maximum, autocorrelation, below mean, and above mean) and are passed through a FC layer to obtain final feature $\mathbf{F}$. Fig. 5.3 illustrates the structural configuration of the teacher model that uses convolutional, FC, and LSTM layers and generates a feature that is a combination of spatial, temporal, and hand-crafted features.

5.3.2 Obtain the compressed DNN model (Student model)

The student model, which is typically smaller and less complex than the teacher model. The student model is the one we have to improve or fine-tune. Next, to reduce the energy, processing, and storage requirements of the teacher model $\mathbf{M}^t$, we adopt DNN compression techniques discussed in [76]. The compression of model $\mathbf{M}^t$ to obtain student model $\mathbf{M}^s$ is performed as per the availability of energy, processing capacity, and storage at the IoT device. We can train the model $\mathbf{M}^s$ on dataset $\mathcal{D}$ to build a classifier. However, to further preserve the energy consumption, we utilize the dataset $\mathcal{D}'$ with reduced length of instances for training $\mathbf{M}^s$. The model $\mathbf{M}^s$ is suitable for its deployment on the IoT device, but, due to limiting structural configuration of $\mathbf{M}^s$, it suffers from inadequate recognition performance.

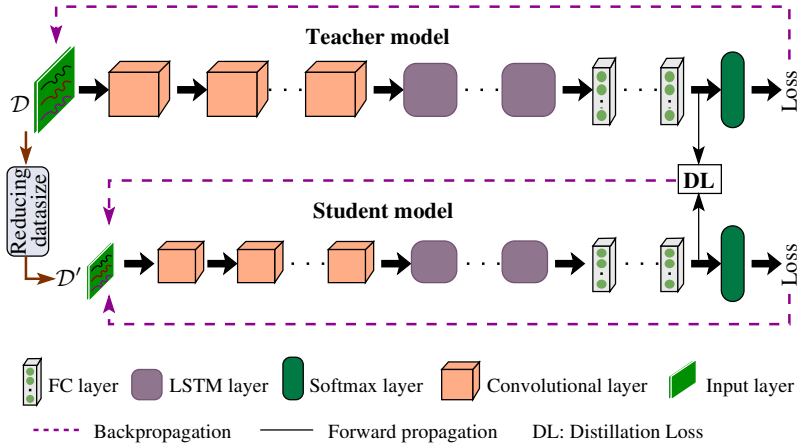


Figure 5.4: Framework for training student on reduced dataset $\mathcal{D}'$ under the guidance of teacher trained on dataset $\mathcal{D}$.

5.3.3 Knowledge distillation (transfer of knowledge) to train student model

During the training process, you use the teacher model to generate soft targets (probabilistic predictions or logits) for the same data that the student model is being trained on. These soft targets contain more information than traditional labels because they capture the teacher model's uncertainty and confidence in its predictions. The inherent limitations of accuracy compromise due to DNN compression in student model $\mathbf{M}^s$ can be overcome by adopting knowledge distillation technique [79]. Here, model $\mathbf{M}^t$ is simultaneously trained with $\mathbf{M}^s$ on datasets $\mathcal{D}$ and $\mathcal{D}'$, respectively. Fig. 5.4 illustrates the simultaneous training of models $\mathbf{M}^t$ and $\mathbf{M}^s$. Here, the logits of $\mathbf{M}^t$ and $\mathbf{M}^s$ are compared at the end of each epoch to estimate loss of $\mathbf{M}^s$.

5.3.4 Loss of student model

The training objective for the student model is to minimize the difference between its predictions and the soft targets provided by the teacher model. This is typically done by using a specialized loss function designed for knowledge distillation. Now, the loss function of student model can be estimated using the two terms, distillation loss function and combined cross entropy loss function. The student model loss function

incorporates distillation loss, where, logits of $\mathbf{M}^t$ and $\mathbf{M}^s$ are compared. The distillation loss function is defined as

1. Distillation loss function:

$$\mathcal{L}_{DL}(\mathbf{v}, \mathbf{z}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \left\| v_{ij} - z_{ij} \right\|_2^2, \tag{5.1}$$

   where, $\mathbf{v}$ and $\mathbf{z}$ are the logits of $\mathbf{M}^t$ and $\mathbf{M}^s$, respectively. $v_{ij} \in \mathbf{v}$ and $z_{ij} \in \mathbf{z}$ are the elements of the corresponding logits vectors. n denotes the number of elements in dataset having k class labels. Let θ_t and θ_s denote the parameters of teacher and student models, respectively. $|| \cdot ||_2^2$ is Fibonacci norm that captures the distribution of noisy labels.

2. Combined cross entropy loss function:

   Let $\mathcal{L}_{CE}^t(\cdot)$ and $\mathcal{L}_{CE}^s(\cdot)$ denote the generalized cross-entropy loss function of $\mathbf{M}^t$ and $\mathbf{M}^s$, respectively. Then the combined loss function that estimates the loss occurs during simultaneous training of $\mathbf{M}^t$ and $\mathbf{M}^s$, is given as follows

$$\mathcal{L}_{comb}(\theta_t, \theta_s; \mathbf{X}, \mathbf{X}', Y) = \underbrace{\mathcal{L}_{CE}^t(\theta_t; \mathbf{X}, Y)}_{\textbf{Teacher loss}} + \underbrace{\mu_1 \mathcal{L}_{CE}^s(\theta_s; \mathbf{X}', Y) + \mu_2 \mathcal{L}_{DL}(\mathbf{v}, \mathbf{z})}_{\textbf{Student loss}}, \tag{5.2}$$

   where, $\mathbf{X} \in \mathcal{D}$, $\mathbf{X}' \in \mathcal{D}'$, and Y denote the set of data instances and set of class labels. μ_1 and μ_2 are the hyper-parameters that control the fractional contribution of loss functions $\mathcal{L}_{CE}^s(\cdot)$ and $\mathcal{L}_{DL}(\cdot)$, respectively.

By aggregating Eq. 5.1 and Eq. 5.2, we define an optimization problem to minimize student loss. Here, the principal objective is to minimize the $\mathcal{L}_{comb}(\cdot)$ during simultaneous training of models $\mathbf{M}^t$ and $\mathbf{M}^s$. Since the contribution of $\mathcal{L}_{CE}^t(\cdot)$ is uniform throughout the training process; therefore we minimize student loss as follows:

$$\min \mu_1 \mathcal{L}_{CE}^{s}(\theta_s; \mathbf{X}', Y) + \mu_2 \mathcal{L}_{DL}(\mathbf{v}, \mathbf{z})$$

$$, \text{s.t.} \quad \mu_1 + \mu_2 = 1, \tag{5.3}$$

$$, \quad 1 \leq \mu_1, \mu_2 \leq 1, \tag{5.4}$$

5.3.5 Training of teacher and student model on dataset

The student model is trained using both the input data and the soft targets from the teacher model. By minimizing the loss between the student's predictions and the teacher's soft targets, the student model learns to mimic the teacher's knowledge. By determining the optimal value of hyper-parameters μ_1 and μ_2, we can estimate the minimum value of student loss during training. Algorithm 5.1 illustrates the steps involve in the training of teacher and student model on dataset $\mathcal{D}$ and $\mathcal{D}'$, respectively.

5.3.6 Compression

Once the student model has been trained with knowledge distillation, it can be deployed in place of the teacher model. Because it is smaller and simpler, it is often more computationally efficient and can run on resource-constrained IoT devices. Here, in our work, we used LNs as IoT devices. The cumbersome model requires substantial computational resources for WQI estimation, leading to increased operational expenses if deployed. To offer a cost-effective solution, we incorporated resource-constrained IoT devices with limited memory, processing power, and storage. Since the cumbersome model cannot be deployed directly on these devices, we applied filter pruning to compress the model. Subsequently, we conduct training for this compressed DNN. Following training, we deployed the trained compressed DNN on our IoT devices for WQI estimation. This deployment on IoT devices allows us to attain the desired level of accuracy in detecting water pollution level, significantly reducing operational costs.

Algorithm 5.1: Training of compressed DNN.

Input: Dataset $\mathcal{D}$ having class label y_i against $\mathbf{x}_i \in \mathcal{D}$, learning rate τ_t and τ_s for teacher and student, respectively, testing data instance x_{te};

Output: Trained parameters θ_t and θ_s of teacher and student, loss $\mathcal{L}_{comb}$, and testing label y_{te} against $\mathbf{x}_{te}$;

Initialize: $\theta_s, \theta_t, \mathcal{L}_{comb}$, $\mu_1 \leftarrow 0.5$ and $\mu_2 \leftarrow 0.5$;

Reducing length of data instance in $\mathcal{D}$ to obtain $\mathcal{D}'$ for training student model $\mathbf{M}^s$;

Perform either of these two steps;

a.) Take average of q axes of p sensors;

b.) Select one most distinguishable axis from q axes;

Obtain dataset $\mathcal{D}'$ with reduced length of instances;

/*Train student model under the guidance of teacher */

while *not converge* **do**

  $a \leftarrow a + 1$;

  $b \leftarrow b + 1$;

  /*Training teacher model $\mathbf{M}^t$ on $\mathcal{D}$*/

  Forward propagation & compute $\mathcal{L}^t_{CE}(\cdot)$;

  Backward propagation $g^a \leftarrow \bigtriangledown^a \mathcal{L}_{CE}(\cdot)$;

  Update weight θ_a by $\theta_{a+1} \leftarrow \theta_a - \tau g^a$;

  /*Training student model $\mathbf{M}^s$ on $\mathcal{D}'$*/

  Forward propagation & compute $\mu_1 \mathcal{L}^s_{CE}(\cdot) + \mu_2 \mathcal{L}_{DL}(\cdot)$;

  Backward propagation $g^b \leftarrow \bigtriangledown^b \Big(\mu_1 \mathcal{L}^s_{CE}(\cdot) + \mu_2 \mathcal{L}_{DL}(\cdot)\Big)$;

  Update weight θ_b by $\theta_{b+1} \leftarrow \theta_b - \tau g^b$;

  Estimate optimal hyper-parameters μ_1 and μ_2 that satisfy Eq. 5.3 and Eq. 5.4;

  Compute combined loss function $\mathcal{L}_{comb}(\cdot)$;

$\mathcal{L}_{comb} \leftarrow \mathcal{L}_{comb}(\cdot)$; /*final combined loss*/

$\theta_t \leftarrow \theta_a$; /*trained teacher parameter*/

$\theta_s \leftarrow \theta_b$; /*trained student parameter*/

Built a classifier $\mathbf{\Pi}^s$ for model $\mathbf{M}^s$;

$\mathbf{\Pi}^s$ predicts testing label y_{te} against $\mathbf{x}_{te}$ ($\mathbf{\Pi}^s : x_{te} \rightarrow y_{te}$);

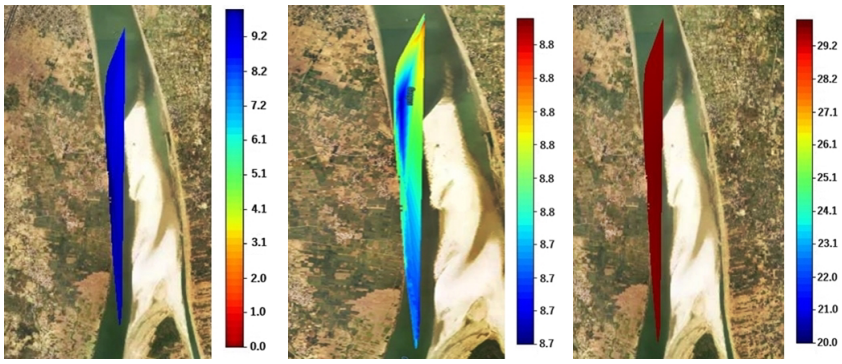
return $\mathcal{L}_{comb}, \theta_t, \theta_s$ and y_{te};

5.4 Experimental evaluation and results

This section presents the experimental analysis for verifying the effectiveness of DNN system. Firstly, we describe a publicly available dataset called River Water Monitoring (RWM) [1]. Next, we present the implementation details of DNN system. Finally, we describe the experimental results.

5.4.1 RWM dataset

The RWM dataset [1] consists of the water parameters of the major Indian rivers. The dataset is collected to measure and analyze the river water quality. In the dataset, the water parameters that are considered to measure the water quality include, temperature, the potential of Hydrogen (pH), Electrical Conductivity (EC), Dissolved Oxygen (DO), total dissolved solids, turbidity, ammonia, *etc.* Further, the dataset is annotated with six labels, *i.e.,* "very bad" , "bad", "medium", "good", "very good", and "excellent", depending upon the pollution labels of the river. Such labels are denoted by $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$, $\mathbf{a_4}$, $\mathbf{a_5}$, and $\mathbf{a_6}$ classes, respectively. Fig. 5.5 illustrates the heat map of river water parameters. The RWM dataset contains 100000 instances with their annotated class labels.



(a) DO. (b) pH. (c) Temperature.

Figure 5.5: Heat maps of different parameters in the sensory dataset [1], on a specific date (for river Ganges in Varanasi).

5.4.2 Implementation details

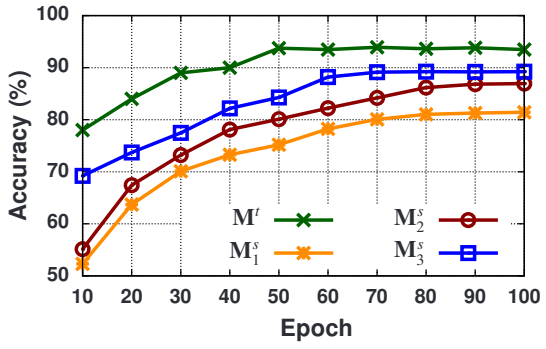
We incorporate Python programming language, where, we use deep learning library Tensorflow for implementing both teacher and student models. Next, we consider four devices, including NodeMCU ($\mathbf{d_1}$), Raspberry Pi 2 ($\mathbf{d_2}$), Raspberry Pi 3 ($\mathbf{d_3}$), and Dell laptop ($\mathbf{d_4}$) with different configuration. The availability of resources (processing/energy/storage) at $\mathbf{d_1}$ and $\mathbf{d_2}$ is 100%, whereas $\mathbf{d_3}$ and $\mathbf{d_4}$ have 50% and 25%

available resources, respectively. Under this configuration, the teacher model can not be deployed on $\mathbf{d_1}$, $\mathbf{d_2}$, and $\mathbf{d_3}$. Thus, we need to compress teacher model ($\mathbf{M}^t$) to obtain three student models $\mathbf{M}_1^s$, $\mathbf{M}_2^s$, and $\mathbf{M}_3^s$ for devices $\mathbf{d_1}$, $\mathbf{d_2}$, and $\mathbf{d_3}$, respectively. The $\mathbf{M}_1^s$ and $\mathbf{M}_3^s$ are highest and least compressed models, respectively.

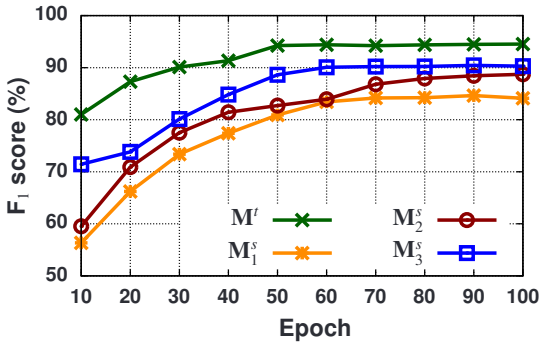
Further, for randomly partitioning the dataset for training and testing sub-datasets, we incorporate *train_test_split* function in sklearn library of Python language. Here, the number of convolutional layers considered during the experiment is 5 having 128 filters each. This layer configuration is selected as we obtained the best performance out of several configurations during the experiments. We use Relu function for normalization on each layer other than softmax function at the output layer. The optimizer during the experiment was "adam", and the learning rate is 0.005.

5.4.3 Experimental results

This section presents various experimental results on RWM [1] dataset to study the training performance, class-wise accuracy, impact of the number of LNs, and the devices used for deploying the DNN models.



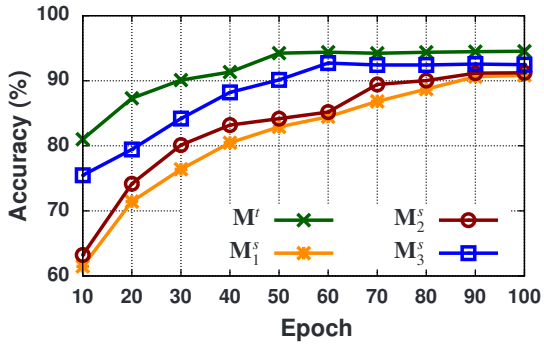
(a1) Accuracy on RWM_1.



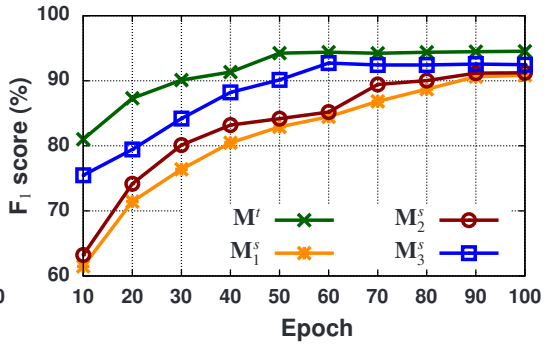
(a2) F_1 score on RWM_1.

5.4.3.1 Training performance

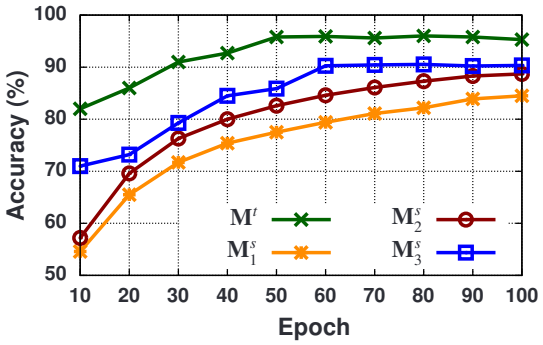
In the experiment, we study the performance of the teacher model ($\mathbf{M}^t$), and its three derived students ($\mathbf{M}_1^s$, $\mathbf{M}_2^s$, $\mathbf{M}_3^s$) during training. The experiments are conducted on



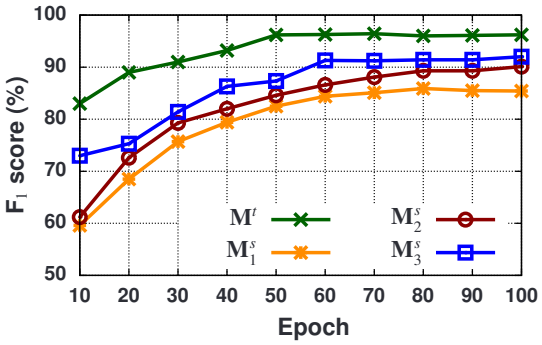
(a3) Accuracy on RWM_1 using KD.



(a4) F_1 score on RWM_1 using KD.



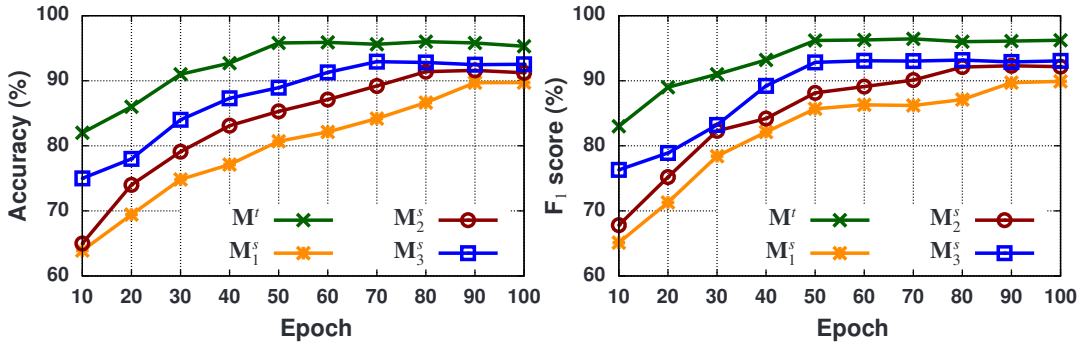
(b1) Accuracy on RWM_2.



(b2) F_1 score on RWM_2.

RWM dataset in two versions, *i.e.,* RWM_1 with all six classes ($\mathbf{a_1}$-$\mathbf{a_6}$) and RWM_2 with five classes ($\mathbf{a_1}$-$\mathbf{a_5}$). The class $\mathbf{a_6}$ (excellent) is obsoleted from RWM_2 as it holds the minimum number of samples in the collected dataset. This experiment also depicts the effect of knowledge distillation during the training of student models under the guidance of teacher model. Part (a1) of Fig. 5.6 illustrates the training time accuracy of different models on the varying number of epochs on RWM_1. The results demonstrate that teacher performance stabilized at 50 epochs, whereas $\mathbf{M}_3^s$, $\mathbf{M}_2^s$, and $\mathbf{M}_1^s$ achieved consistent accuracy at 60, 80, and 80 epochs, respectively. The teacher converses faster due to complex architecture configuration that generates well-refined features in limited training epochs. A similar observation can be made for F_1 score on RWM_1, as illustrated in part (a2) of Fig. 5.6. The training accuracy and F_1 score of teacher model goes beyond 93% for RWM_1, but three student models suffer from accuracy compromise as in this part of the experiment knowledge distillation was not adopted during training.

Further, parts (a3) and (a4) of Fig. 5.6 illustrate the effect of adopting knowledge



(b3) Accuracy on RWM_2 using KD.

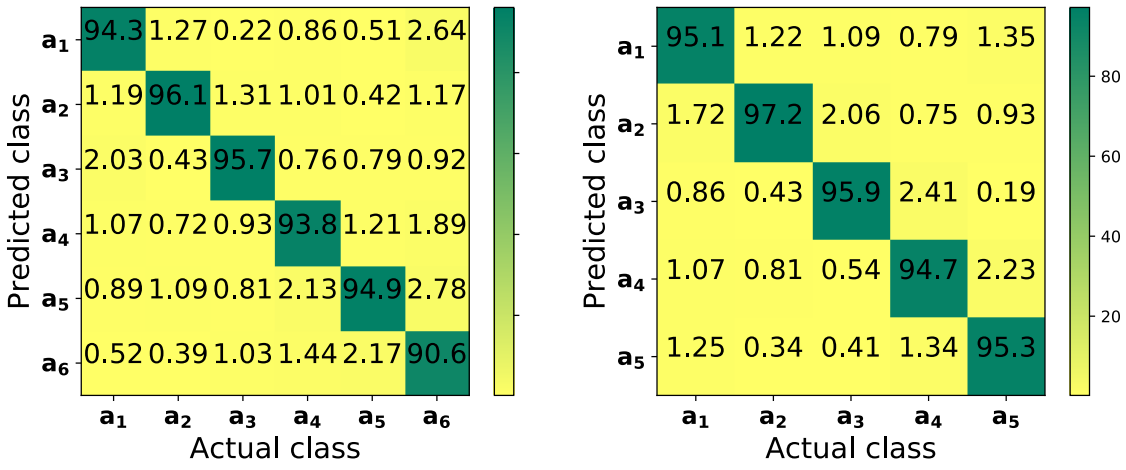
(b4) F_1 score on RWM_2 using KD.

Figure 5.6: Performance results of models $\mathbf{M}^t, \mathbf{M}_1^s, \mathbf{M}_2^s$ and $\mathbf{M}_3^s$ during training with and without Knowledge Distillation (KD). Parts (a1)-(a4) and (b1)-(b4) consider RWM dataset versions RWM_1 and RWM_2 having six ($\mathbf{a_1} - \mathbf{a_6}$) and five ($\mathbf{a_1} - \mathbf{a_5}$) classes, respectively.

distillation during the training of the student models on RWM_1. We observe from the result that a significant improvement in training accuracy ($> 6\%$) of student models is achieved by incorporating knowledge distillation. Similarly, we also observed the same accuracy and F_1 score patterns on RWM_2 with slight improvement as it incorporates only 5 classes. Parts (b1)-(b4) of Fig. 5.6 illustrate the accuracy and F_1 score with and without KD achieved during training on RWM_2.

5.4.3.2 Class-wise accuracy of teacher model

This experiment derives the class-wise accuracy of the teacher model in the proposed ERWM system using RWM_1 and RWM_2. We observe from part (a) of Fig. 5.7 that the class $\mathbf{a_2}$ (bad) achieves the highest accuracy of 96.10% and class $\mathbf{a_6}$ (excellent) achieves the lowest accuracy of 90.6%. It is due to $\mathbf{a_2}$ have a large number of training instances during feature extraction, and the number of instances during training are lowest for class $\mathbf{a_6}$. During the experiment, we observe that several instances of class $\mathbf{a_4}$ overlap with $\mathbf{a_5}$, which hampers the accuracy of both the classes. Similarly, we observe the class-wise accuracy achieved on dataset version RWM_2 having 5 classes ($\mathbf{a_1} - \mathbf{a_5}$), as shown in part (b) Fig. 5.7.



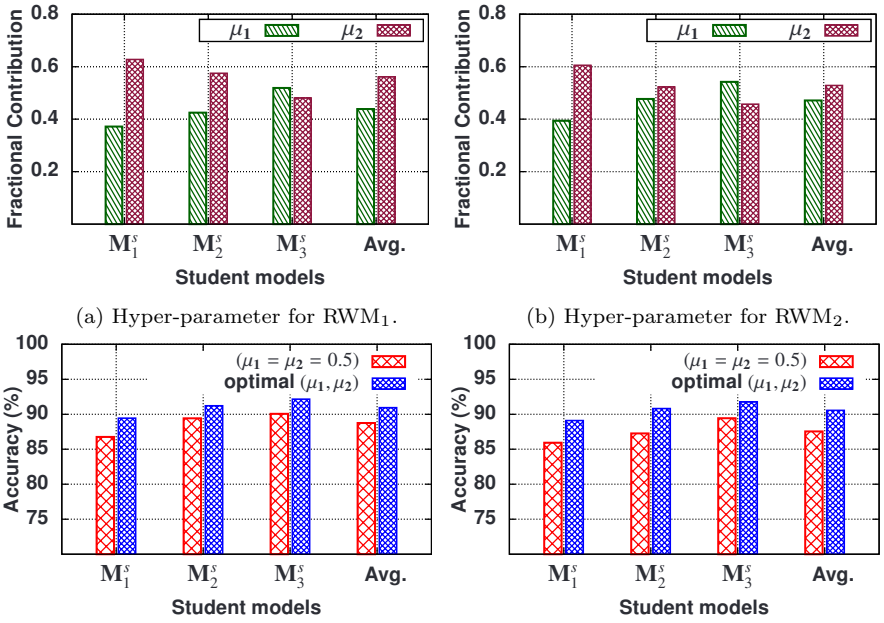
(a) RWM_1 dataset. (b) RWM_2 dataset.

Figure 5.7: Class-wise accuracy of teacher on dataset versions RWM_1 and RWM_2.

5.4.3.3 Impact of hyper-parameters

Next, we study the impact of hyper-parameters on the accuracy of the student models $\mathbf{M}_1^s$, $\mathbf{M}_2^s$, and $\mathbf{M}_3^s$ on RWM_1 and RWM_2 datasets. The hyper-parameters determine the fractional contribution of cross-entropy and distillation loss functions. Parts (a) and (b) of Fig. 5.8 illustrate the value of hyper-parameters μ_1 and μ_2, respectively, for student models on RWM_1 and RWM_2. We can observe from the result that for low resource consuming smaller model ($\mathbf{M}_1^s$) the value of hyper-parameter μ_2 is high. It is because distillation loss plays a significant role in improving the accuracy of smaller models. However, for model $\mathbf{M}_3^s$, the gap between student and teacher model is small and $\mathbf{M}_3^s$ is somewhere self capable; thus, the contribution of cross-entropy loss supersedes distillation loss. Fig. 5.8 also demonstrates the average value (Avg.) for hyper-parameters (fractional contribution of loss) and accuracy over all the student models. Further, parts (c) and (d) of Fig. 5.8 illustrate the impact of taking an optimal value of μ_1 and μ_2 on the recognition performance of student models. We can observe from the result that the least improvement of 1.9% in accuracy is obtained by taking the optimal value

of μ_1 and μ_2 over the equal value of hyper-parameters ($\mu_1 = \mu_2 = 0.5$).



(a) Hyper-parameter for RWM_1.

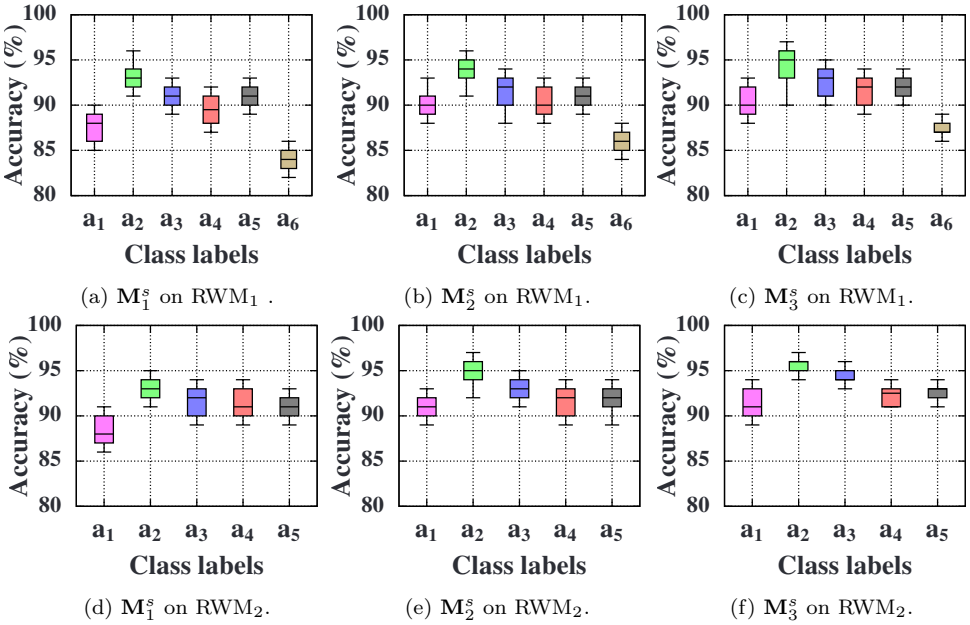
(b) Hyper-parameter for RWM_2.

(c) Accuracy on RWM_1.

(d) Accuracy on RWM_2.

Figure 5.8: Impact of hyper-parameters (μ_1 and μ_2) on student models.

5.4.3.4 Performance of student model



(a) $\mathbf{M}_1^s$ on RWM_1 .

(b) $\mathbf{M}_2^s$ on RWM_1.

(c) $\mathbf{M}_3^s$ on RWM_1.

(d) $\mathbf{M}_1^s$ on RWM_2.

(e) $\mathbf{M}_2^s$ on RWM_2.

(f) $\mathbf{M}_3^s$ on RWM_2.

Figure 5.9: Class-wise performance of student models on RWM_1 and RWM_2.

Fig. 5.9 shows the boxplots for estimated accuracy of different classes in datasets versions RWM_1 and RWM_2 on $\mathbf{M}_1^s$, $\mathbf{M}_2^s$, and $\mathbf{M}_3^s$ models. Parts (a)-(c) of Fig. 5.9 illustrate the class-wise accuracy of student models on different classes in RWM_1. It is observed from the result that for all the student models, the recognition performance of class $\mathbf{a_2}$ supersedes all other classes. This is because the RWM_1 version of RWM dataset contains a significantly higher number of instances for class $\mathbf{a_2}$. Also, the features extracted for class $\mathbf{a_2}$ holds most distinguishable features in contrast with features of other classes in the RWM_1. Similarly, parts (d)-(f) of Fig. 5.9 depict the class-wise performance of student models on RWM_2. Hereafter, results are illustrated only for RWM_1 version of dataset due to space limitations for depicting all the results.

5.5 Conclusion

In this chapter, we proposed a River Water pollution Monitoring system in LoRa-based Internet of Things. First, the system builds a cumbersome DNN model. Next, we train the big model to estimate WQI with high accuracy. Further, we proposed a compressed DNN using filter-level pruning to estimate WQI. Next, the system uses the knowledge distillation technique for training compressed DNN. During knowledge distillation, the student learns under the supervision of teacher model. After deploying the small model on IoT devices, we obtained acceptable accuracy in detecting water pollution level.