Chapter 5

Leveraging Augmented Intelligence of Things to Enhance Lifetime of UAV-Enabled Aerial Networks

In the previous chapters, we discussed about the effective data transmission in high ceiling buildings using LoRa networks. This chapter proposed an approach to enhance the lifetime of UAV-enabled aerial networks. The main objective of the EU-AIoT approach is to improve the lifetime of aerial networks by replacing DNN on battery-operated UAVs with lighter versions, incurring minimal accuracy compromise.

5.1 Introduction

Artificial intelligence simulates the human cognitive ability to provide autonomous control and intelligent decision making using machines. Specifically, it creates a self-aware machine that can understand the human mind theories [103, 104]. Though artificial intelligence provides a promising future to flourish big data analytics in industries; however, exact simulation of human cognitive ability is tedious to achieve. Augmented Intelligence (AI) extends artificial intelligence and liberates the models for making autonomous decisions. AI acts in a supportive role and involves human experts or operators to take intelligent and appropriate decisions; thus, it avoids the bewilderment of simulating exact human behaviour. AI extract different patterns from the collected dataset and reports such patterns to the users. The users analyze the reported patterns to take an appropriate decision. Virtual assistants like Alexa, Siri, and Google Assistant are examples of AI, where assistants only provide demanded data instead of making decisions.

Internet of Things (IoT) enabled monitoring and control systems are proliferating due to easier availability of low cost and small sensors [105]. Unmanned Aerial Vehicle (UAV) based monitoring is one of the potential applications of IoT [106]. The coordination of AI and IoT remarkably improves the computation speed and range of IoT devices used in industries, which facilitate human decision making at network edge [103]. AI and IoT jointly form a new term, namely Artificial Intelligence of Things (AIoT). AIoT simultaneously involves human and machine intelligence working together on the network edge on IoT devices. AIoT encourages faster predictive analytics in different aspects like scientific calculations, medical urgency, activities and environment monitoring [104, 107, 108].

UAV-enabled aerial networks act as an intermediatory component between the data acquisition units and Cloud [106]. The sensors at the acquisition units produce a vast amount of data. UAVs periodically collect these data, process them, and generate results for transferring to Cloud for further actions. The data-processing and result generation on UAV is preferred over off-the-site (*e.g.,* Cloud) to avoid colossal communication delay. The battery-operated UAVs provide service for pre-specified (or limited) time-interval due to faster battery depletion. However, adverse situations (like search and rescue operations during natural calamities, floods, *etc.*) demand an extended lifetime of a UAV-enabled aerial network.

It is tedious to deploy DNN on UAV due to excessive resource demand. Researchers made significant efforts to reduce such resource demand of DNN [109–112]. However, this reduction results in accuracy compromise. Knowledge Distillation (KD) is a concept that improves the performance of the lightweight DNN using the generalization ability of the large-size DNN [3]. Furthermore, AIoT facilitates different power-saving modes in smartphones, where the users can select a suitable mode to enhance operability on low power [113]. Similarly, the lifetime of an aerial network can improve by running the suitable DNN based on the residual power. However, UAVs have limited storage; therefore, it is impractical to pre-stored different versions of DNN. An AIoT based solution is to monitor the residual energy of the UAV; thereafter, the user demands lightweight DNN from Cloud.

This work aims to design an AIoT based approach that transforms a large-size DNN into a lightweight to ensure quick training. Cloud perform this transformation and training of lightweight DNN. Let α, β, and γ be the residual battery power, available memory on UAV, and the frequency of the given task, respectively. The user (or operator) decides γ as per the requirement and available energy. We investigate the following problem in this work: *how quickly we can train a lightweight deep neural*

network that can achieve the required accuracy while satisfying α and β constraints for a given γ? To solve the problem, this work proposes an approach for Enhancing the lifetime of UAV-enabled aerial network using AIoT (EU-AIoT). This approach is applicable for the scenario of UAV-enabled aerial networks, where the residual battery power and available memory of the UAV change dynamically.

5.1.1 Motivation and contribution

This work is motivated by the following limitations in the existing literature. The prior studies [106,114–116] on UAV-based processing (or data collection) do not consider the provisioning of the dynamic environment, where processing varies overtime to enhance the lifetime of UAV-enabled networks. Next, the existing work on the dropout [117] used a fixed/random value of dropout and did not guarantee to prune recurrent units that require substantial resources. The prior work in [117–119] have not considered device constraints while compressing large-size DNN. Finally, the previous work on KD [3, 5] improved the performance of lightweight DNN. However, none of them emphasized on accelerating the training process. The main contributions are as follows:

- **(α, β, γ)-DNN transformation:** We first transform a large DNN into a lightweight, which achieves high accuracy using available residual energy α and memory β on task frequency γ. The operator decides γ using AIoT. We refer to this transformation as (α, β, γ)-DNN transformation problem.
- **Clever training of lightweight DNN:** We next utilize the knowledge distillation technique to simultaneously train the student (lightweight DNN) and teacher (large-size DNN). We introduce the concept of *clever training*, which incorporates layer sharing among teacher and student with selective back-propagation. The shared layers of the student undergo training via selective back-propagation.
- **EU-AIoT algorithm:** We further present an algorithm that uses the proposed solution of (α, β, γ)-DNN transformation problem and quick training of obtained lightweight DNN. The algorithm plays a vital role in the scenarios where the resources of UAVs change dynamically and demand different versions of DNN.
- **Evaluation:** We finally conducted various experiments to verify the effectiveness of the EU-AIoT approach. In doing so, we utilize the existing large-size DNN discussed in [60] and the publicly available River Water Pollution Monitoring (RWPM) dataset [120].

The rest of the chapter is formatted as follows. The next section discusses the existing literature resembling this work, followed by preliminaries. We propose the overall methodology of the EU-AIoT approach in Section 5.3. Further, the prototype setup

and performance evaluation are covered in Section 5.4. Finally, Section 5.5 concludes the chapter.

5.2 Background

This section describes the prior studies that serve as a background for the proposed approach, followed by preliminaries. Table 5.1 illustrates the list of notations used in this work.

5.2.1 UAVs based data collection and processing

Authors in [114] introduced the concept of UAV-based classification in remote areas. They utilized federated learning, where the global model runs on the ground station and local models on the UAVs. Extending the concept of federated learning in the UAV-enabled aerial network, the authors in [106] used the lightweight version of Dense-MobileNet on UAVs for estimating the air quality index. Authors in [115] designed a three-dimensional mapping system for evaluating air quality index using UAVs. Data collection using UAVs is one of the crucial tasks; thus, authors in [116] in exploited the UAV-based data collection from the dense wireless sensor networks. They utilized a compressive data gathering technique.

5.2.2 DNN compression and knowledge distillation

The authors in [117] utilized random values of dropout to remove insignificant connections. However, such random dropout hampers the performance due to the removal of relevant connections. In [119] the authors introduced the mechanism of layer factorization for reducing floating-point operations. Similarly, the authors in [121] emphasized generating the DNN once-for-all (OFA) networks. OFA produced various DNN with different specifications for minimizing training costs. Further, to improve the performance of the compressed DNN, the authors in [3] proposed a KD technique. It incorporated the difference between logits of large-size DNN (teacher) and compressed DNN to improve the student's generalization ability. Finally, the concept of layer sharing among teacher and student is introduced in [5]. It helped in improving the performance of the student and avoiding random initialization.

Table 5.1: List of notations used in this work.

Symbol	Description	Symbol	Description
$\mathcal{D}$	Dataset	d	Dropout
α	Residual energy	β	Available memory
γ	Task fraction	$\mathcal{N}$	Number of UAVs
$\mathcal{L}$	Loss function	n	Total instances in $\mathcal{D}$
k	Number of classes	$\mathbf{M}^t$	Teacher (large-size DNN)
$\mathcal{T}$	Flight time	$\mathbf{M}^s$	Student (lightweight DNN)

5.2.3 Preliminary

This work considers a scenario of river water pollution monitoring, where UAV collect and analyze the data from multiple data acquisition units. Let $\mathcal{N}$ denotes the set of UAVs used in the considered scenario, where $\mathcal{N} = \{1, 2, \cdots, N\}$. All battery-operated UAVs run lightweight DNN to classify the pollution level of the river water. Next, the obtained result is forwarded to Cloud for storage and further operations. Moreover, the UAVs are connected to the Cloud for receiving firmware updates. The firmware updates are the new version of lightweight DNN running on UAVs, as per the demand ($< \alpha, \beta, \gamma >$). Let $\mathcal{D}$ denotes a river water pollution data having n sensory instances and k class labels (pollution levels). An instance i of dataset $\mathcal{D}$ is denoted as $\mathbf{x}_i$, $\forall i \in \{1, \cdots, n\}$. Each instance $\mathbf{x}_i$ holds sensory data points and corresponds to a class label l, where $l \in \{1, \cdots, k\}$. Let the large-size and lightweight DNN be denoted by $\mathbf{M}^t$ and $\mathbf{M}^s$, respectively.

Definition 5.1 (Task frequency) *Let $\mathcal{T}$ and ω denote the flight time per trip and time interval between two successive classifications of water pollution by the UAV using data collected from the acquisition unit, respectively. The task frequency γ is defined as: $\gamma = \left(\frac{\mathcal{T}}{\omega}\right)$, where $\omega < \mathcal{T}$.*

Definition 5.2 (Flight time) *Flight time ($\mathcal{T}$) refers to the time for which an UAV can maintain the aerial networks at given residual energy α. Wind speed and its direction, along with the altitude of flying, affect the flight time.*

5.3 EU-AIoT: Enhancing the lifetime of UAV-enabled aerial network using AIoT

This section describes the overall methodology incorporated in the EU-AIoT approach. The principal objective of EU-AIoT is to fasten the training of lightweight DNN (student), which is created as per the demand of UAV. The demand is a tuple $< \alpha, \beta, \gamma >$, where α, β, and γ denote residual energy, available memory, and task frequency, respectively. Moreover, if a UAV receives trained lightweight DNN after long hours of training, it may shift to another level of battery depletion. The main steps of EU-AIoT approach are: **(α, β, γ)-DNN transformation** of large-size (teacher) to lightweight DNN (student) followed by **clever training of student**.

5.3.1 (α, β, γ)-DNN transformation

In the EU-AIoT approach, the UAV initially sends the request in the form of $< \alpha, \beta, \gamma >$ to the Cloud. This request demands an appropriate version of lightweight DNN from the Cloud, which enhances the lifetime of UAV-enabled aerial networks using AIoT. Such enhancement facilitates effective and long-term monitoring, especially in adverse situations like search and rescue operations during natural calamities. This section incorporates *dynamic dropout, weight factorization,* and *reduction of gated units* to transform large-size DNN into lightweight for given α, β, and γ. Procedure 1 summarizes the steps of (α, β, γ)-DNN transformation to obtain lightweight DNN.

5.3.1.1 Dropout on large-size DNN

We first apply dropout on large-size DNN for removing unimportant connections to obtain *dropout DNN*. The obtained DNN is a lighter version of a large-size DNN where weight is scaled with the given dropout, denoted as d. We dynamically estimate optimal d despite using a fixed value, suitable for given $< \alpha, \beta, \gamma >$. We start with a high dropout $d = 0.5$ and $d = 0.8$ for hidden and input units, respectively [117]. Let $\mathcal{C}_o$ and $\mathcal{C}_d$ represent the number of connections in prior and post-applications of dropout, respectively. Let $iter_{max}$ and ϕ denote maximum iterations runs for d and hyper-parameter, respectively. The updated dropout (d') is estimated as follows: $d' \leftarrow d \times \max\left\{\sqrt{\frac{\mathcal{C}_o}{\mathcal{C}_d}}, \left(1 - \frac{iter}{\phi \times iter_{max}}\right)\right\}$, where $iter$ represent the number of iterations.

5.3.1.2 Weight factorization and reduction of gated units

We introduce the mechanism to shrink the resource requirements of DNN layers, including convolutional, fully connected, and recurrent (Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU)). We adopt the weight factorization technique to reduce the resource requirements of convolutional and fully connected layers. We further utilize the concept of reducing gated units to curtail the resources of LSTM and GRU.

- **Weight factorization:** This work uses the weight factorization technique [119] to curtail the parameters and FLOPs of convolutional and fully connected layers. The technique uses an intermediate multiplexing layer between i and $i+1$ layers. The factorization minimizes the computation requirement if the following condition holds: $R_i \leq \frac{I_i \times O_i}{I_i + O_i}$ [119]. Here R_i, I_i, and O_i are intermediate, input, and output dimensions of layer i, respectively.
- **Reducing gated unit:** The parameters and FLOPs involved in LSTM and GRU are directly proportional to the gated operations. We incorporate the technique to reduce the gated units. The elimination of gated units minimizes the execution complexity with no or minimal accuracy compromise. We replace LSTM with coupled LSTM and GRU with Minimal Gated Unit (MGU) to reduce the gated operations.

5.3.1.3 Optimizing execution energy and memory consumption

We further deduce an expression for execution energy and memory consumption of lightweight DNN. We later present an optimization problem to minimize execution energy and memory consumption that satisfy constraints α (residual energy), β (available memory), and γ (task frequency). The users (or operators) decide γ in AIoT. Let δ ($\delta < \mathcal{T}$) denotes the lifetime demanded by the UAV within α and β. For time interval ω between two sucessive classification from UAV, we have: $\gamma = \left(\frac{\delta}{\omega}\right)$. Let $\mathcal{E}$ denotes the energy consumption in performing classification once on UAV. Thus, energy consumption ($\mathbb{E}$) in δ time with γ frequency is: $\mathbb{E} = \left(\frac{\delta}{\omega}\right)\mathcal{E}$. Let e_1 is the energy consumption per FLOP on layer j of lightweight DNN running on UAV then $\mathcal{E} = \sum_{j \leftarrow 1}^{L} e_1 \times \text{FLOPs}_j$, where L is the number of layers in lightweight DNN and FLOPs_j is the FLOPs required per layer j. Thus, $\mathbb{E}$ can also be defined as:

$$\mathbb{E} = \left(\frac{\delta}{\omega}\right)\mathcal{E} = \left(\frac{\delta}{\omega}\right)\sum_{j \leftarrow 1}^{L} e_1 \times \text{FLOPs}_j, \quad \omega < \mathcal{T}, \delta < \mathcal{T}. \tag{5.1}$$

Let m_1 denote the memory required to execute a single FLOP on the UAV. Thus, to-

tal memory consumption ($\mathbb{M}$) on the UAV is estimated as follows: $\mathbb{M} = m_1 \sum_{j \leftarrow 1}^{L} \text{FLOPs}_i$. The objective function finally optimizes the execution energy and memory with the residual energy of battery (α), available memory (β) and task frequency (γ), is as follows:

$$\begin{aligned} \min \ & \Psi\mathbb{E} + (1-\Psi)\mathbb{M} \\ s.t., \ & \mathbb{E} \leq \alpha, \mathbb{M} \leq \beta, \\ & \omega < \mathcal{T}, \delta < \mathcal{T} \end{aligned} \tag{5.2}$$

where Ψ ($0 \leq \Psi \leq 1$) is a neutralizing variable to remove the mismatch between units of energy and memory consumption.

The optimization problem in Equation 5.2 can be solved (near-optimal solution) by iteratively performing dropout, followed by weight factorization and reducing gated units. Additionally, the selection of obtained near-optimal solution is subject to the highest achieved accuracy. The solution provides suitable lightweight DNN for UAVs. However, it needs faster training before its deployment. Procedure 1 summarizes the steps involved in the (α, β, γ)-DNN transformation to obtain lightweight DNN from large-size DNN.

5.3.2 Clever training of lightweight DNN

This section introduces a **clever training approach** for lightweight DNN obtained in the previous step. The mechanism substantially improves the performance of lightweight DNN (student) and reduces the training time. We incorporate the knowledge distillation technique, where student (un-trained lightweight DNN) training is propagated under the teacher's guidance. The teacher ($\mathbf{M}^t$) is an un-trained large-size DNN, which is simultaneous trained with the student ($\mathbf{M}^s$) on dataset $\mathcal{D}$. Despite simple training and logits comparison in knowledge distillation, we adopt *layer sharing* among $\mathbf{M}^t$ and $\mathbf{M}^s$. We further utilize *selective backpropagation* of the shared layers to reduce the training time. We finally discuss an iterative algorithm (Algorithm 1) for the EU-AIoT approach.

5.3.2.1 Layer sharing among teacher ($\mathbf{M}^t$) and student ($\mathbf{M}^s$)

$\mathbf{M}^s$ utilizes the knowledge (or generalization ability) of $\mathbf{M}^t$ to improve its performance in KD. $\mathbf{M}^s$ compares its logits with the logits of $\mathbf{M}^t$ and reduces its loss after training. In baseline KD [3], $\mathbf{M}^t$ is pre-trained; thus, its fine-tuned logits become a hard target

Procedure 5.1: 1: (α, β, γ)-DNN transformation.

Input: Large-size DNN ($\mathbf{M}^t$) with C_o connections, hyper parameter ϕ, UAV demand $< \alpha, \beta, \gamma >$, $\mathcal{T}, \delta$;
Output: Lightweight model ($\mathbf{M}^s$) as per $< \alpha, \beta, \gamma >$;
Initialize: $d \leftarrow 0.5$, $iter \leftarrow 1$, $iter_{max} \leftarrow 50$, $mo \leftarrow []$;
do
  Apply dropout d on $\mathbf{M}^t$ with C_o connections;
  $C_d \leftarrow$ connections after dropout;
  $d' \leftarrow d \times \max\left\{\sqrt{\frac{C_d}{C_o}}, \left(1 - \frac{iter}{\phi \times iter_{max}}\right)\right\}$;
  $d \leftarrow d'$; /*Updated dropout value*/
  $C_o \leftarrow C_d$; /*Updated connections count*/
  Perform weight factorization;
  Apply recurrent unit reduction;
  Estimate $\mathbb{E} \leftarrow \left(\frac{\delta}{\omega}\right) \sum_{j \leftarrow 1}^{L} e_1 \times \text{FLOPs}_j$;
  Compute $\mathbb{M} = m_1 \sum_{j \leftarrow 1}^{L} \text{FLOPs}_i$;
  if *Constraints of Equation 5.2 are satisfied* **then**
    Store lightweight DNN j;
    $mo \leftarrow append$(loss of lightweight DNN);
    $j \leftarrow j + 1$;
  $iter \leftarrow iter + 1$;
while $(iter \leq iter_{max})$;
if $size\_of(mo) == 0$ **then**
  Increase $iter_{max}$ **goto** Step 1;
$a \leftarrow \arg\max\{mo\}$;
a^{th} lightweight DNN is desired model;
return lightweight DNN satisfying $< \alpha, \beta, \gamma >$;

for comparison with logits of the un-trained student. Therefore, the performance improvement is not up to the mark. Consequently, it is beneficial to train $\mathbf{M}^t$ and $\mathbf{M}^s$ simultaneously on dataset $\mathcal{D}$. The logits of $\mathbf{M}^t$ are soft to compare with $\mathbf{M}^s$; hence, we observe substantial performance improvement. Moreover, it consumes colossal resources due to simultaneous training of $\mathbf{M}^t$ and $\mathbf{M}^s$. It also takes a large number of epochs to converge the training of $\mathbf{M}^s$.

This chapter uses the concept of layer sharing among $\mathbf{M}^t$ and $\mathbf{M}^s$, as discussed in [5]. We share the first f fractions of student layers with the teacher. In other words, the first f fractions of $\mathbf{M}^s$ layers are common among $\mathbf{M}^s$ and $\mathbf{M}^t$, as shown in Figure 5.1(a). Thus, we have only $(1 - f)$ fractions of $\mathbf{M}^s$ layers that should undergo DNN compression (discussed in Section 5.3.1). The layer sharing helps in improving the performance of $\mathbf{M}^s$, reduces the training time, and establishes the probabilistic output in fewer epochs (faster convergence). Furthermore, we use data from multiple

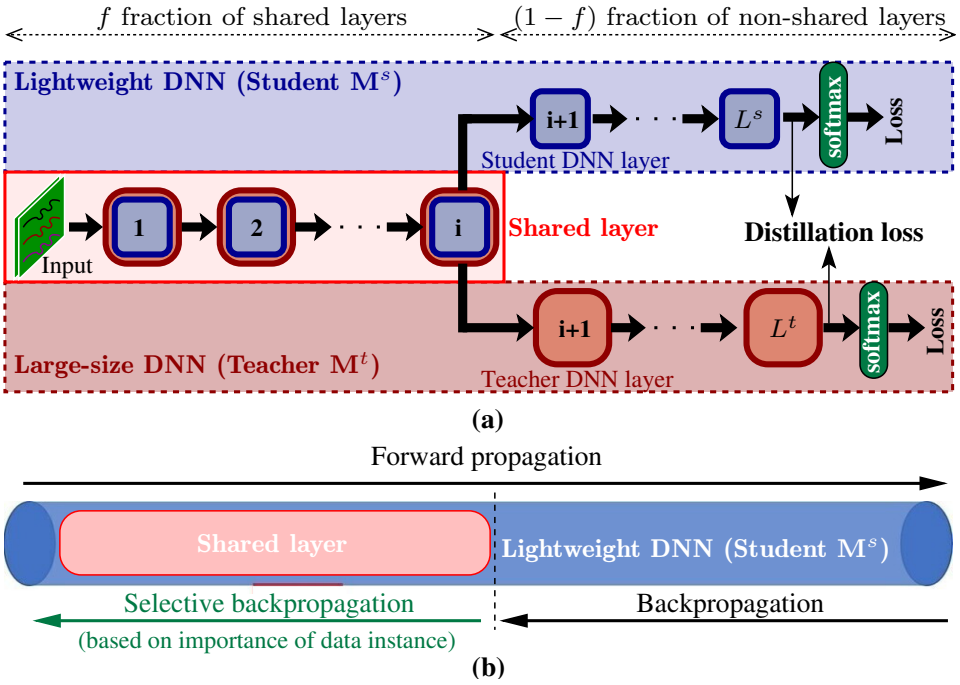


Figure 5.1: Illustration of clever training approach involving knowledge distillation from the teacher ($\mathbf{M}^t$) to the student ($\mathbf{M}^s$).

sensors to monitor river water. Thus, the sensory dataset with low complexity of $\mathbf{M}^t$ facilitate high-order layers sharing among $\mathbf{M}^s$ and $\mathbf{M}^t$. $\mathbf{M}^t$ facilitates 30% layers sharing (determined experimentally) with $\mathbf{M}^s$ for Raspberry Pi attached with UAV. In other words, $f = 0.3$ fractions of layers are common in $\mathbf{M}^s$ and $\mathbf{M}^t$.

5.3.2.2 Selective backpropagation

Layer sharing among $\mathbf{M}^s$ and $\mathbf{M}^t$, discussed in the previous section, helps to improve the performance and reduce the training time of $\mathbf{M}^s$. This section performs another operation of selective backpropagation on shared layers to further reduce the training time of $\mathbf{M}^s$. We choose only shared layers of $\mathbf{M}^s$ for selective backpropagation because the weight parameters in shared layers are more refined and stable due to the $\mathbf{M}^t$.

The training time of the DNN depends on the dataset and epochs for training. Time consumed in each epoch is proportional to the time required in forward and backward propagation. Moreover, the time of backpropagation is nearly twice that of forward [122] due to the gradient operations. Thus, we incorporate the mechanism of selective backpropagation, where selected weight parameters undergo backpropagation. However, selecting the weight parameters during backpropagation is tedious. Therefore, we estimate the importance of data instances to select optimal weights for

backpropagation [123].

Importance of data instance: Gradient determines the importance of the data instance, *i.e.,* a large value of gradient implies higher relevancy and vice-versa. Thus, we estimate the importance after the forward propagation in each epoch of shared layers (f fractions) among $\mathbf{M}^s$ and $\mathbf{M}^t$. We next perform backpropagation on the data instances having importance greater than threshold. This selective backpropagation reduces the epoch time. Let $\mathbf{d}_i \in \mathcal{D}$ denotes the i^{th} instance of $\mathcal{D}$ having instance-label pair $(\mathbf{x}_i, y_i)$ and weight vector $\mathbf{w}_i$. The importance (ξ_i) of instance $\mathbf{d}_i$ with DNN specific constant ζ ($0 < \zeta \leq 1$) and loss $\mathcal{L}$, is given as:

$$\xi_i = \zeta \left\| \frac{\partial \mathcal{L}(\mathbf{w}_i)}{\partial \mathbf{w}_i} \right\|_2^2, \text{where } \forall i \in \{1, 2, \cdots, n\}. \tag{5.3}$$

5.3.2.3 Estimating reduction in training time

Let Ω_o denotes the time consumed in one epoch of $\mathbf{M}^s$ training. Let t^f and t^b represent the forward and backward propagation time of an epoch on dataset $\mathcal{D}$, respectively. Ω_o is estimated as: $\Omega_o = t^f + t^b$. t^f and t^b are directly proportional to non-zero rows in weight matrix $\mathbf{W}$, denoted as $\mathcal{Z}(\mathbf{W})$. $\mathbf{W} = \{\mathbf{w}_i\}_{i \leftarrow 1}^n$, n is the number of instances in dataset $\mathcal{D}$. Now, $t^f = a_1 \mathcal{Z}(\mathbf{W})$ and $t^b = b_1 \mathcal{Z}(\mathbf{W})$, where a_1 and b_1 are the proportionality constants. Now, we can estimate the total time ($\mathbb{T}$) consume in E epochs for training $\mathbf{M}^s$ as:

$$\mathbb{T} = E\Omega_o = E\Big(t^f + t^b\Big) = E\Big(a_1 \mathcal{Z}(\mathbf{W}) + b_1 \mathcal{Z}(\mathbf{W})\Big). \tag{5.4}$$

Using the assumption that time consumed in backpropagation is nearly twice in contrast with the forward [122], we have:

$$\mathbb{T} = E\Big(a_1 \mathcal{Z}(\mathbf{W}) + 2a_1 \mathcal{Z}(\mathbf{W})\Big) = 3a_1 E \mathcal{Z}(\mathbf{W}). \tag{5.5}$$

Let $\mathbb{1}(\xi_i)$ denotes the inductive function of gradient importance for data instance i ($1 \leq i \leq n$), which is given as:

$$\mathbb{1}(\xi_i) = \begin{cases} 1, & \textbf{if } \xi_i \geq \mathcal{G}_{th}, \\ 0, & \text{otherwise}, \end{cases} \tag{5.6}$$

where $\mathcal{G}_{th}$ is the gradient threshold to retain a gradient during backpropagation. Weight

matrix $\mathbf{W}$ changes to $\mathbf{W}'$ for selective backpropagation, *i.e,* $\mathbf{W}' = \{\mathbb{1}(\xi_i)\mathbf{w}_i\}_{i\leftarrow 1}^{n}$. Thus, total time ($\mathbb{T}'$) consume in E epochs for training $\mathbf{M}^s$ with selective backpropagation, is given as:

$$\mathbb{T}' = E\Omega_o{}' = E\Big(t^f + t^{b\prime}\Big) = E\Big(a_1\mathcal{Z}(\mathbf{W}) + b_1\mathcal{Z}(\mathbf{W}')\Big). \tag{5.7}$$

Using Equation 5.5 and Equation 5.7, we can estimate the reduction in training time ($\mathcal{R}$) due to selective backpropagation, as follows:

$$\mathcal{R} = \mathbb{T} - \mathbb{T}' = E\Big(2a_1\mathcal{Z}(\mathbf{W}) + b_1\mathcal{Z}(\mathbf{W}')\Big). \tag{5.8}$$

Since, we apply selective backpropagation on f fraction of layers, thus, total reduction in training time is given as:

$$\mathcal{R}^f = (\mathbb{T} - \mathbb{T}')f = E\Big(2a_1\mathcal{Z}(\mathbf{W}) + b_1\mathcal{Z}(\mathbf{W}')\Big)f. \tag{5.9}$$

5.3.2.4 Estimating loss of $\mathbf{M}^s$

During the training of $\mathbf{M}^s$, two types of loss functions persist side-by-side, *i.e.,* standard loss (*e.g.,* cross entropy) and distillation loss [3]. The standard cross-entropy loss, denoted as $\mathcal{L}_{CE}(\cdot)$, is the local loss of $\mathbf{M}^s$ that captures the discrepancy between actual and predicted output. The distillation loss, denoted as $\mathcal{L}_{DL}(\cdot)$, captures the difference between logits of $\mathbf{M}^s$ and $\mathbf{M}^t$, during their simultaneous training. Let u_{ij} and v_{ij} denote elements of logit vectors of $\mathbf{M}^s$ and $\mathbf{M}^t$, respectively, $\mathcal{L}_{DL}(\cdot)$ is given by:

$$\mathcal{L}_{DL}(\mathbf{u}, \mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k}\Big\|u_{ij} - v_{ij}\Big\|_2^2. \tag{5.10}$$

The combined loss, denoted as $\mathcal{L}_{comb}(\cdot)$), incurred on $\mathbf{M}^s$ is estimated as: $\mathcal{L}_{comb}(\cdot) = \mathcal{L}_{CE}(\cdot) + \mathcal{L}_{DL}(\cdot)$. We can also add a variable ($\lambda, \lambda \in [0, 1]$) to vary the contribution of $\mathcal{L}_{CE}(\cdot)$ and $\mathcal{L}_{DL}(\cdot)$, *i.e.,* $\mathcal{L}_{comb}(\cdot) = \lambda\mathcal{L}_{CE}(\cdot) + (1-\lambda)\mathcal{L}_{DL}(\cdot)$.

5.3.2.5 Algorithm for EU-AIoT approach

We finally present an iterative algorithm (Algorithm 1) for the EU-AIoT approach. It transforms a large-size DNN into lightweight as per the demand of UAV using (α, β, γ)-

Algorithm 5.1: EU-AIoT algorithm.

Input: Dataset $\mathcal{D}$, large-size DNN ($\mathbf{M}^t$), energy α_i, memory β_i, and frequency γ_i of UAV i ($\forall i \in \mathcal{N}$);
Output: Trained lightweight DNN for UAVs;
1 Initialize: $f \leftarrow 0.3$; /*Assuming 30% layer sharing*/
2 **while** *True* **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 Receive request from i^{th} UAV $< \alpha_i, \beta_i, \gamma_i >$;
5 Ensure f fraction of layer sharing $\mathbf{M}^t$ and $\mathbf{M}^s$;
6 Call **Procedure 1** to determine un-trained $\mathbf{M}^s$;
  • Solve optimization problem in Equation 5.2;
7 Obtained un-trained lightweight DNN $\mathbf{M}^s$;
8 **for** *epoch* $\leftarrow 1$ *to* E_i **do**
9 Forward propagation for all L_i layers of $\mathbf{M}^s$;
10 Estimate$\mathcal{L}_{comb}(\cdot) = \lambda\mathcal{L}_{CE}(\cdot) + (1-\lambda)\mathcal{L}_{DL}(\cdot)$;
11 Backpropagation for $(1-f)$ fraction of L_i;
12 Estimate importance ξ using Equation 5.3;
13 Selective backpropagation on f of L_i;
14 **return** trained lightweight DNN ($\mathbf{M}^s$) for UAV i;

transformation (Procedure 1).

Time complexity of Algorithm 5.1 depends upon the *while* loop at Step 2, *for* loop at Steps 3 and 8, and Procedure 1. We first estimate the time complexity of Procedure 1, which depends upon the estimation of dropout d', energy $\mathbb{E}$, and memory $\mathbb{M}$. Additionally, $iter_{max}$ also determines the complexity of Procedure 1. Since the estimation of d', $\mathbb{E}$, and $\mathbb{M}$ requires complexity of $O(1)$ and $iter_{max}$ is limited to 50; therefore, the time complexity of Procedure 1 is $O(1)$. We next obtain the time complexity of the *for* loop at Step 8 using Equation 5.5, which is given as: $O(3a_1EW) \approx O(EW)$. Using the estimated complexities of Procedure 1 and *for* loop at Step 8, we obtain time complexity of *for* loop at Step 3 as $O((N)EW)$. *while* loop at Step 2 reflects the number of times the monitoring is performed using EU-AIoT approach. Thus, the complexity of Algorithm 5.1 is $O((N)EW)$ for single round monitoring. Moreover, this work considers $(N) << E$; therefore, the time complexity of Algorithm 5.1 is $\approx O(EW)$.

5.4 Prototype setup and performance evaluation

This section evaluates EU-AIoT approach on existing dataset [120] using large-size DNN [60].

Table 5.2: Baseline schemes for ablation studies. $\mathbf{M}^s$ and $\mathbf{M}^t$ are student and teacher, respectively. KD: Knowledge Distillation.

Scheme	Symbol	Description
No KD	$\mathbf{S}_1$	$\mathbf{M}^s$ is trained independently
Pre-trained KD	$\mathbf{S}_2$	$\mathbf{M}^s$ is trained with pre-trained $\mathbf{M}^t$
Un-trained KD	$\mathbf{S}_3$	$\mathbf{M}^s$ is trained with un-trained $\mathbf{M}^t$
Layer sharing KD	$\mathbf{S}_4$	$\mathbf{M}^s$ is trained with un-trained $\mathbf{M}^t$ using shared layers
Proposed	$\mathbf{S}_5$	$\mathbf{M}^s$ is trained with un-trained $\mathbf{M}^t$ using shared layers & selective backpropagation

5.4.1 Experimental setup

5.4.1.1 Dataset

This work uses a publically available dataset, namely River Water Pollution Monitoring (RWPM) [120]. RWPM was collected to determine the pollution level of Indian rivers, including Godavari, Yamuna, Ganga, Hoogly, and Hindon. The pollution was determined by analyzing parameters, including pH, electrical conductivity, dissolved oxygen, turbidity, *etc.* RWPM dataset has 100000 instances with annotated class labels (pollution levels), *i.e., very bad, bad, medium, good, very-good*, and *excellent*. We use symbols $\mathbf{c_1}$, $\mathbf{c_2}$, $\mathbf{c_3}$, $\mathbf{c_4}$, $\mathbf{c_5}$, and $\mathbf{c_6}$ to represent these six classes. We randomly select the dataset partitions, *i.e.,* 70% and 30% for training and testing, respectively.

5.4.1.2 Schemes for ablation studies

We use different schemes for ablation studies, as shown in Table 5.2.

5.4.1.3 Prototype specification and implementation details

We used various resources in the prototype setup, including a sensor probe (Libelium), UAV, Raspberry Pi, router, and laptop. The sensor probe (Libelium) is deployed near the bank of the river, which wirelessly transfers data to the Raspberry Pi (1.2 GHz CPU and 1 GB of RAM) attached to the UAV. The lightweight DNN running on the Raspberry Pi classifies the pollution level of the river. Further, using a router (Cisco router), the UAV transfers the result to the Laptop (Intel Core $i7$ processor and 8 GB RAM) acting as Cloud. Additionally, the UAV sends demand in the form of $< \alpha, \beta, \gamma >$ to the laptop to extend the lifetime. The flight time ($\mathcal{T}$) of UAV is 60 minutes in a single run under regular operation. Figure 5.2 illustrates the different components of the

Figure 5.2: Prototype components: (1) Sensor probes (Libelium), (2) prototype deployment area, (3) router deployment, (4) UAV, (5) laptop (as Cloud), and (6) water parameters on smartphone.

prototype setup for the EU-AIoT approach. Most of the parameters for implementation are similar as given in [60].

5.4.1.4 Performance metrics

We considered different performance metrics, including F1-score (F1), precision (P), recall (R), and Training Time Reduction Ratio (TTR^2). Let $T_1, \cdots, T_5$ denote time taken by Schemes $\mathbf{S}_1, \cdots, \mathbf{S}_5$ to train student, respectively. TTR^2 is defined as: $\frac{T_i}{\max\{T_1,\cdots,T_5\}}$, $i \in \{1, \cdots 5\}$. TTR^2 lies in the range $(0, 1]$, where $\text{TTR}^2 \rightarrow 1$ indicates higher training time and vice-versa.

5.4.2 Experimental results

5.4.2.1 Impact of different schemes

We first performed the experiments to study the impact of different schemes ($\mathbf{S}_1 - \mathbf{S}_5$) on F1-score, precision, recall and TTR^2. We set a fixed value of $\alpha = 0.5$, $\beta = 0.8$, $\gamma = 12$, and $\delta = 40$ minutes and use large-size DNN [60]. Table 5.3 illustrates that $\mathbf{S}_1$ requires the minimal training FLOPs and training time (TTR^2=0.65). It is because $\mathbf{S}_1$ does not incorporate KD guided training. Contrarily, scheme $\mathbf{S}_1$ achieved the least performance in the absence of the KD technique. Further, $\mathbf{S}_3$ required maximum time (TTR^2=1.00) and epochs due to simultaneous training of $\mathbf{M}^s$ and $\mathbf{M}^t$ without layer sharing. Table 5.3 also illustrates that the proposed EU-AIoT approach ($\mathbf{S}_5$) achieved high order performance and required comparable training time. It is due to the inclusion of layer sharing and selective backpropagation on optimal lightweight

Table 5.3: Illustration of F1-score, precision, recall, and TTR^2 on different schemes using RWPM dataset and large-size DNN in [60]. ($\alpha = 0.5$, $\beta = 0.8$, $\gamma = 12$, and $\delta = 40$ minutes).

Scheme	FLOPs	F1-score	Precision	Recall	TTR^2
$\mathbf{S}_1$	6.25×10^9	78.22%	75.49%	73.41%	0.65
$\mathbf{S}_2$	7.12×10^9	87.59%	86.22%	84.47%	0.78
$\mathbf{S}_3$	1.57×10^{10}	90.29%	88.76%	87.83%	1.00
$\mathbf{S}_4$	1.13×10^{10}	92.07%	91.27%	89.71%	0.92
$\mathbf{S}_5$	9.14×10^9	91.43%	90.47%	89.42%	0.83

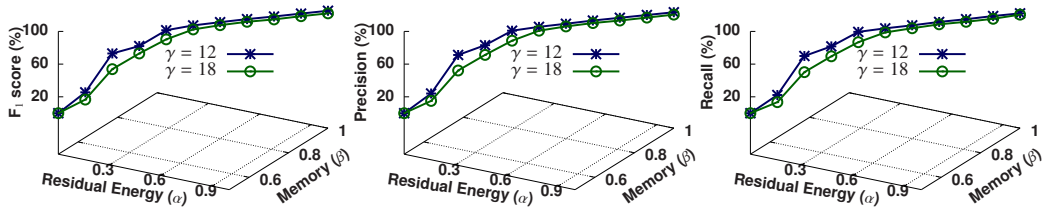
Table 5.4: Impact of layer sharing (in %) among $\mathbf{M}^s$ and $\mathbf{M}^t$ with fixed $\beta = 0.8$, $\gamma = 12$, and $\delta = 40$ minutes.

Layer sharing	10%	20%	**30%**	40%	50%	60%
Residual energy (α)	0.44	0.47	**0.51**	0.59	0.68	0.79
F1 score $\mathbf{M}^s$ (in %)	89.44	90.26	**91.43**	92.51	93.07	93.73
Precision $\mathbf{M}^s$ (in %)	87.81	89.44	**90.47**	91.77	92.27	92.84
Recall $\mathbf{M}^s$ (in %)	86.54	87.72	**89.42**	90.04	90.89	91.06
FLOPs ($\times 10^9$)	13.1	11.7	**9.14**	8.25	6.49	5.37

DNN ($\mathbf{M}^s$), obtained using (α, β, γ)-DNN transformation. Therefore, $\mathbf{S}_5$ proves to be effective for UAVs both in terms of performance and training time.

5.4.2.2 Impact of layer sharing

Table 5.4 depicts the impact of layer sharing on residual energy (α), training FLOPs, and other performance metrics. We set a fixed value of $\beta = 0.8$, $\gamma = 12$, and $\delta = 40$ minutes. The obtained results indicate that layer sharing increases the achieved performance (F1 score, precision, and recall). It is due to the fact that layer sharing made $\mathbf{M}^s \rightarrow \mathbf{M}^t$ in architecture, which improved the performance. However, the increase in the size of $\mathbf{M}^s$ demands a higher value of α and training FLOPs, as shown in Table 5.4. Moreover, beyond 30% ($f = 0.3$) layer sharing the increment in α ($0.58 - 0.51 = 0.08$) is more rapid despite performance improvement. Thus, 30% layers sharing proves optimal and achieved F1 score $> 91\%$.



(a) F1 score. (b) Precision. (c) Recall.

Figure 5.3: Impact of simultaneous change in α (residual energy) and β (available memory) on the performance of $\mathbf{M}^s$ on $\gamma \in \{12, 18\}$.

5.4.2.3 Impact of change in α and β

We next study the performance of $\mathbf{M}^s$ on the collected dataset with varying α, β, $\gamma \in \{12, 18\}$ and fixed $\delta = 40$ minutes. Figure 5.3(a) illustrates the F1 score achieved by $\mathbf{M}^s$, trained using the EU-AIoT approach. We can observe from the results that $\mathbf{M}^s$ achieves F1 score around 91% at $\alpha = 0.5$ and $\beta = 0.8$ for $\gamma = 12$. The higher performance in limited residual energy and memory is due to layer sharing and selective backpropagation in the EU-AIoT approach. Similar observations can be made for precision and recall, as illustrated in Figure 5.3(b) and Figure 5.3(c), respectively. Furthermore, a higher γ demands the lighter version of $\mathbf{M}^s$ in limited residual energy and storage. Thus, we can observe subsequent decrement in the performance (F1 score, precision, and recall) on increasing γ from 12 to 18.

5.4.2.4 Impact of compression ratio

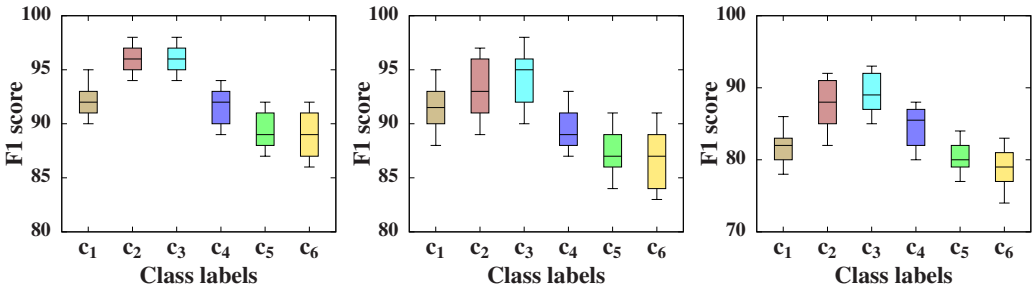
We further illustrate the impact of the compression ratio (size of M^s/size of M^t) on F1 score, precision, recall, and training FLOPs. The compression ratio depends upon the available resources (α, β) on the UAV that demands a lightweight DNN from Cloud. Additionally, γ and δ are crucial in determining the compression ratio. Table 5.5 demonstrates the various compression ratio and corresponding performance metrics. The results follow an expected pattern of decreasing performance upon increasing the compression ratio. We observe that increasing compression beyond $\times 20$ results in higher accuracy compromise due to limited layers and gated units to meet the compression demand from UAVs.

5.4.2.5 Impact of lifetime demanded (δ) on performance

We also evaluate the impact of lifetime demanded (δ) by UAVs on the class-wise F1 score of $\mathbf{M}^s$ obtained using EU-AIoT approach. We considered $\delta = 30$ minutes, $\delta = 40$

Table 5.5: Illustration of performance *vs.* compression ratio.

Compression ratio	×60	×50	×40	×30	×**20**	×10
F1 score $\mathbf{M}^s$ (in %)	64.43	74.59	79.81	84.54	**91.43**	93.22
Precision $\mathbf{M}^s$ (in %)	61.38	72.93	77.29	82.71	**90.47**	91.59
Recall $\mathbf{M}^s$ (in %)	60.29	70.14	75.63	79.29	**89.42**	93.82
FLOPs ($\times 10^9$)	1.29	4.23	6.54	8.21	**9.14**	12.21



(a) $\delta = 30$ minutes. $\delta = 40$ minutes. $\delta = 50$ minutes.

Figure 5.4: Impact of lifetime demanded (δ) by UAV on class-wise F1 score at fixed $\alpha = 0.5$, $\beta = 0.8$, and $\gamma = 12$.

minutes, and $\delta = 50$ minutes for fixed value of $\alpha = 0.5$, $\beta = 0.8$, and $\gamma = 12$. The results in Figure 5.4 depicts the higher class-wise performance with lower δ. It is because a lower value of δ required a less compressed version of $\mathbf{M}^s$ that can achieve higher performance. Moreover, an increase in δ, increased the compression requirements, which results in accuracy compromise. Additionally, the F1 score of class $\mathbf{c_2}$ ("bad") and $\mathbf{c_3}$ ("medium") is higher at all three values of δ.

5.4.2.6 Impact of the loss contribution (λ) on performance

We finally conduct experiments to analyze the impact of the loss contribution variable (λ) on the performance at different values of α. There are three basic methods to estimate λ, *i.e.,* optimal, equal (0.5), and random. The optimal value of λ always achieved higher performance than equal and random, as shown in Table 5.6. We can observe from the result that with the increase α, the role of distillation loss $(1 - \lambda)$ decreases. It is because the gap between $\mathbf{M}^s$ and $\mathbf{M}^t$ decreases, which compels performance improvement through a minimal level of knowledge distillation. Thus, selecting

Table 5.6: Impact of the loss contribution (λ) on performance with varying α at fixed $\beta = 0.8$ and $\gamma = 12$.

Residual energy	**Optimal value of λ**		$\lambda = 0.5$	**Random value of λ**
	λ	**F1 score**	**F1 score**	**F1 score**
$\alpha = 0.4$	0.39	88.73%	86.22%	85.42%
$\alpha = 0.5$	0.44	91.43%	89.32%	89.49%
$\alpha = 0.6$	0.51	92.22%	90.17%	91.29%
$\alpha = 0.7$	0.57	92.81%	90.83%	90.40%
$\alpha = 0.8$	0.67	93.29%	91.58%	91.22%

an optimal value of λ for a given α is crucial to obtain adequate performance.

5.5 Conclusion and future work

This chapter proposed an approach to enhance the lifetime of UAV-enabled aerial networks via AIoT. The main objective of the EU-AIoT approach is to improve the lifetime of aerial networks by replacing DNN on battery-operated UAVs with lighter versions, incurring minimal accuracy compromise. EU-AIoT has used optimal dropout selection followed by weight factorization and reducing gated units to obtain lightweight DNN from given large-size DNN satisfying constraints battery power, memory, and task frequency. We further utilized the concept of knowledge distillation, where we adopted layer sharing among teacher and student, followed by selective backpropagation of shared layers during student training. We finally carried out various experiments to validate the effectiveness of the EU-AIoT approach. This work provides a future direction toward developing a DNN compression technique for more versatile applications utilizing UAVs during disasters.