Divide2Conquer (D2C): A Comprehensive Study on Decentralized Overfitting Remediation in Deep Learning

by

Md. Saiful Bari Siddiqui
22166054

A Thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is my own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Saiful Bari Siddiqui
22166054

Approval

The thesis titled "Divide2Conquer (D2C): A Comprehensive Study on Decentralized Overfitting Remediation in Deep Learning" submitted by

1. Md. Saiful Bari Siddiqui (22166054)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on October 19, 2024.

Examining Committee:

External Examiner:
(Member)

Dr. Taskeed Jabid
Professor
Department of Computer Science and Engineering
East West University

Internal Examiner:
(Member)

Dr. Aniqua Nusrat Zereen
Assistant Professor
Department of Computer Science and Engineering
Brac University

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md Sadek Ferdous
Professor
Department of Computer Science and Engineering
Brac University

Chairperson:
(Chair)

Sadia Hamid Kazi, Ph.D
Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

This research has been conducted using publicly available datasets, including CIFAR-10, Fashion MNIST, MNIST, FER2013, UrbanSound8k, AG News, TESS, RAVDESS, and CREMA-D. These datasets are widely used in the machine learning and deep learning communities and were created for educational, research, and benchmarking purposes. Each dataset has been obtained from sources that provide them for public use, with appropriate permissions where necessary.

No personal or private information is included in any of the datasets used in this research, and all data has been anonymized. Furthermore, this research does not involve any human participants, animals, or biological materials. As such, it falls under the category of research that does not require institutional ethics approval.

The usage of these datasets adheres to the ethical guidelines established by the data providers, and the research aligns with the principles of fairness, transparency, and respect for privacy in AI research. Additionally, the findings and models developed in this research are not intended to be used in any context that could lead to harm, exploitation, or discrimination.

Abstract

Overfitting remains a persistent challenge in deep learning, primarily attributed to data outliers, noise, and limited training set sizes. This thesis presents **Divide2Conquer (D2C)**, a novel technique designed to address this issue. D2C proposes partitioning the training data into multiple subsets and training separate identical models on them. To avoid overfitting on any specific subset, the trained parameters from these models are aggregated and averaged periodically throughout the training phase, enabling the model to learn from the entire dataset while mitigating the impact of individual outliers or noise. Empirical evaluations on multiple benchmark datasets across various deep learning tasks from different domains demonstrate that D2C effectively improves generalization performance, particularly for larger datasets. This study verifies D2C's ability to achieve significant performance gains as a standalone technique and also when used in conjunction with other overfitting reduction methods through a series of experiments, including analysis of decision boundaries, loss curves, and other performance metrics. Additionally, we provide a rigorous mathematical justification for our hypothesis and analyze the applicability of the D2C method through extensive experimentation on various datasets covering multiple domains. We also delve into the trade-offs associated with D2C and explore strategies to mitigate these challenges, providing a comprehensive understanding of D2C's strengths and weaknesses.

Keywords: Aggregating; Averaging; Deep Learning; Hyperparameter; Overfitting; Training Data.

Dedicated to the Martyrs

of the

July Revolution

Acknowledgement

Firstly, I thank the almighty Allah for granting me the courage, patience, persistence and determination to accomplish this research work.

I express my deepest sense of gratitude and indebtedness to my supervisor, **Professor Dr. Md. Golam Rabiul Alam, Brac University** for his unconditional faith in me. His sincerity and dedication to research always inspired me. Throughout my thesis journey, he made me feel capable through his persistent support and knowledgeable counsel.

I am very much grateful to my best friend, **Md. Mohaiminul Islam, Lecturer, United International University**, for his enormous support in culminating this research work. He was instrumental in refining the ideas and formulating the mathematical intuition behind my hypothesis.

I am also very thankful to **Institute for Advanced Research (IAR), UIU** and especially **Professor Dr. M. Rezwan Khan, Director, IAR, UIU** for believing in us and funding this research work. Our experiments were computationally expensive. It would be impossible to complete this research work without the financial support from IAR.

I express my gratitude to all my teachers throughout my life, especially the ones who believed in me from the very beginning, **Mr. Shahed Mahmud, Mr. Anis Faruque** & **Mr. Aziz Uddin** from Chittagong Collegiate School. I also thank my classmates at BUET and Brac University for being there whenever I needed them.

Last but not the least, I would like to take this opportunity to express gratitude to my parents for their unwavering support, encouragement and prayers. I would like to thank my beloved wife for being there for me at every juncture throughout this journey. I also thank my sisters for pushing me towards my goals through support and encouragement.

Table of Contents

List of Figures

List of Tables

Nomenclature

The following list describes several symbols & abbreviation that will be later used within the body of the document:

$\emptyset$ Null/Empty Set

ϵ Epsilon

σ Sigma

$AUC - ROC$ Area under the Recall Over Precision Curve

$D2C$ Divide2Conquer

E Number of Epochs in the Edge Model training before each round of Central Averaging

MCC Matthew's Correlation Co-efficient

N Number of Subsets of the Training Set

Chapter 1

Introduction

Deep Learning models often do well on the data they train on, but the performance drops massively on unseen data from different distributions. It is a common problem in machine learning, particularly in deep learning, where a model learns the training data too well, to the point of memorizing it. This phenomenon is known as Overfitting. Due to this, the model's performance on new data becomes significantly worse than its performance on the training data. The model's ability to make accurate predictions is compromised as a result. This is why addressing overfitting is absolutely essential in deep learning.

1.1 Motivation

Many methods have been proposed to reduce overfitting over the years [1]. Early Stopping [2] is one of the most intuitive methods used for reducing overfitting, not letting the model know the training set too well. However, this method also brings a constraint in learning complex features. Also, often the best accuracy is achieved quite a bit after the point where overfitting seemingly began [3]. Network reduction views outlier data points as noise and reduces the model complexity. However, this method also brings a constraint in learning complex features [4]. This method is beneficial in decision-tree-based models [5] but is not always the best solution for neural networks. Expanding the Training Set is a method that is well-suited for almost all kinds of models [6]. Acquiring more training data is one way of doing so, but it often requires monumental efforts. Data Augmentation is another way of doing the same and is often the go-to method followed in many deep learning applications [7]. However, selecting the appropriate data augmentation technique for a particular dataset is often tricky. Regularization is one of the most influential and popular techniques to reduce overfitting. This essentially picks the more essential features that have a decisive impact on determining which class a particular sample belongs to. Regularization techniques are associated with penalties so that the model is not entirely dependent on the training data points. L1 regularization eliminates the less important features, while L2 regularization assigns smaller weights to them. Dropout is the most relevant overfitting-reducing method for deep learning models [8]. It is another regularizing technique that drops a portion of the connections between neural network layers. However, even dropout can not always counter overfitting and sometimes degrades the performance of a model [9]. Eventually, the model still trains on the entire training data at once, and the robust neural network

structures find their ways to fit on the training set a bit too much. This is where our motivation comes from. It can be a good idea to not let the neural network train on the entire dataset in the first place. Intuitively, we should combine multiple models that train on different portions of the data so that the model can not familiarize itself with all of it at once, and a consensus can be achieved by focusing on the general patterns observed in almost all the portions of the data.

1.2 Research Problem

Our study is loosely inspired by Federated Optimization [10]. The federated optimization technique suggests training in each edge device using a similar network on the data available in that local device. However, after a specific time, each device sends the parameter weights of its model(not the data) to a central server, where the weights are aggregated and averaged. Finally, the averaged weights are sent to all the edge devices again and the loop continues. Federated optimization, however, is aimed at safe utilization of data situated in edge devices, not better generalization.

In our study, we implemented a novel method, Divide2Conquer (D2C). We divided our training data into multiple subsets and trained a model each(all having the same architecture) with the training subsets. After an epoch, or, a few epochs, we performed weighted averaging of all the parameter weights from all the subsets and shared the weights back with the edge models (the models that are used to train each subset of the training data). Then the whole process is repeated for several global epochs.

The K-Fold-Cross-Validation (K-Fold-CV) method for evaluation proposed by Korjus K. et al. [11] has some similarities to our proposed method in the sense that it also creates subsets by partitioning the dataset and makes use of the whole training data. However, this method is different from ours since the erroneous samples still make it to the training phase (K-1/K)-th of the time in K-Fold CV. D2C method partitions data and the whole training process is partitioned too, which means none of the subsets is fed with a particular outlier while training except for one of them, minimizing the impact of these erroneous samples.

D2C method also has similarities with Bagging [12] in the form of using subsets of data. However, in bagging (Bootstrap Aggregating), the data is not completely separated for different models as we proposed. Instead, the technique involves creating multiple subsets of the original dataset by randomly sampling with replacement. This means that some data points may be repeated in a subset, while others may be excluded altogether. Also, in bagging, the aggregation is done by combining the output probabilities, not the model weights.

1.3 Research Objective

The objective of this study is to propose and assess a method that addresses overfitting using a decentralized approach, exploiting the separation of training data. Our experiments encompass multiple datasets across different domains, and each

of the cases suggests dividing the training set into multiple subsets, training them separately, and averaging the parameters after every few epochs can significantly reduce overfitting. To evaluate the performance, the primary comparison was between the performances of the base Neural Network architecture used for the edge models using the entire training data and the performance of our models using the D2C method. We summarize our contributions through this study below:

- We introduce a new method, D2C, that helps in reducing overfitting significantly while being conceptually simple and easy to implement. We also provide the mathematical justification behind this hypothesis.
- The D2C method can be applied on top of many other data augmentation and regularization techniques, and our experiments show that this results in a clear improvement in the model's generalization ability, across various datasets from multiple domains.
- We extensively tune the hyperparameters introduced by this method and report the findings, providing important directions for future applications.
- We also delve into the trade-offs associated with D2C, including its computational overhead and potential diminishing returns for smaller datasets. We explore strategies to mitigate these challenges and provide a comprehensive understanding of D2C's strengths and weaknesses.

1.4 Thesis Organization

The thesis is organized as follows. In chapter 2, we discuss relevant literature. We establish the theoretical justification behind our hypothesis in chapter 3. In chapter 4, we formulate the research problem and delve deeper into the mathematical intuitions behind our hypothesis. In chapters 5 and 6, we discuss our methodology and lay out the experimental specifications. In chapter 7, we evaluate our approach through empirical experiments on multiple datasets and analyze the results. Chapter 8 contains concluding remarks.

Chapter 2

Literature Review

Several authors have addressed the issue of overfitting while performing various deep learning based tasks. The common methods to proposed to resist overfitting include different forms of regularization and ensemble methods.

2.1 Classical Approaches

M. Cogswell et al. proposed a new regularizer called DeCov [13] which helps reduce overfitting in deep neural networks by Decorrelating Representations. Their regularizer inspires diverse or non-redundant representations by minimizing the cross-covariance of hidden activations in Deep Neural Networks. M.K Rusia and D.K Singh came up with two customized CNN models to recognize facial expressions with reduced overfitting [14]. Their study focused on optimizing the effect of hyper-parameters. Some of these hyper-parameters are the learning rate, filter size, activation functions, and neural network architecture.

Dropout [8] was proposed by Srivastava et al. back in 2014, and since then, this technique has been extensively used in very complex neural network architectures successfully. It was indeed an outstanding contribution specifically for deep learning-based models. Batch Normalization [15] proposed by Ioffe and Sergei primarily focuses on better convergence and somewhat contributes to reducing overfitting. J. Kolluri and V.K Kotte came up with $L\frac{1}{4}$ regularization to solve the problems faced by L1 and L2 regularization techniques [16].

Qinghe Zheng et al. presented an implicit regularization approach at the sample level, incorporated into a two-stage training process [17]. Optimizing the decision boundary through the use of a pre-training stage and an implicit regularization training stage, his method was able to achieve state-of-the-art performances in several datasets.

2.2 Ensemble Methods

Ensembles are also often used to improve generalization. The ensemble-based ELVD model [18] managed to outperform the traditional VGGNet and DropoutNet models in terms of reducing overfitting. Ginhua Wen also used the ensemble method for the overfitting-prone task of facial expression recognition and achieved better performances compared to the base models [19]. Zehong Zeng et al. came up with an ensemble framework that incorporates several techniques to prevent overfitting [20]. Min-Gu Kim et al. also proposed parallel ensemble networks to reduce overfitting in ECG data and prevent the degradation of generalization performance as the training progresses [21].

2.3 Distributed Learning Approach

We implemented a variation of our proposed method based on federated optimization preliminarily for facial expression recognition using FedNet [22]. This model uses the technique of dividing the training set into multiple subsets and performing federated averaging. This model achieves excellent results on both CK+ and FER-2013 datasets compared to other state-of-the-art models for these datasets.

Even though many researchers worked on addressing the overfitting problem, it remains a prevalent issue in supervised machine learning. Methods like Early Stopping, Network Reduction, Training Set Expansion, Regularization, and Dropout are effective [1], but they all have their limitations. Our proposed method tries to build an approach that focuses on achieving better generalization and can be implemented on top of other overfitting-reducing techniques. Furthermore, we combine our proposed technique with classification tasks across different domains to demonstrate the overfitting-resistant nature of it.

Chapter 3

Theoretical Framework

In this chapter, first, we discuss the underlying causes of the problem we are trying to solve, Overfitting. Then we lay out our hypothesis about the proposed D2C method and delve deeper into why this method will be able to address the problem of overfitting effectively. Finally, we discuss the expected aftermath of applying our method in the light of typical counter-overfitting approaches. This will help us understand what to expect from a technique that treats overfitting.

3.1 The Underlying Causes Of Overfitting

Overfitting happens when a model fits too well on the data that it trains on. It effectively captures even the random characteristics from the training data, randomness that would be insignificant in real-world applications. In particular, noisy data and outliers can significantly contribute to overfitting in neural networks if not properly addressed. In short, the underlying causes of overfitting are:

1. **Noise in the Data:** Noise refers to random variations or errors present in the data that do not represent meaningful patterns. This noise can arise from various sources, including sensor inaccuracies, measurement errors, or human error during data collection. Neural networks can learn and memorize noise in the training data, especially if the noise is significant compared to the signal. As a result, the model may fit the noise rather than the underlying patterns, leading to poor generalization.

2. **Outliers:** Outliers are data points that significantly deviate from the rest of the dataset. These can be legitimate data points that represent rare events or errors in the data collection process. Outliers can distort the learning process of neural networks, especially if the model tries to accommodate them during training, which complex neural networks often do. If the model overly focuses on fitting outliers, it may fail to capture the underlying patterns of the majority of the data, leading to overfitting.

3. **Model Complexity:** Complex neural network architectures with a large number of parameters have a higher capacity to memorize the training data. If the model is too complex relative to the amount of training data available, it can lead to overfitting. This phenomenon is even more prevalent when there are outliers and noise in the training data.

4. **Insufficient Data:** When the training dataset is small, the model may not capture enough of the underlying patterns and instead memorize the noise in the data. This lack of diversity can lead to overfitting.

The challenge with complex neural networks is that these powerful models can learn not only the representative data but also the noise and outliers and try to fit all the data at once. Many methods have been adopted to address this challenge, but, the model still trains on the entire training data, and the robust neural network structures find their ways to fit on the training set a bit too much.

3.2 The Hypothesis

Our study focuses on the portion of training data that is responsible for overfitting. From the discussions above, we can see that the presence of outliers and noisy data is a fundamental reason behind overfitting. In this study, we investigate a probable method to minimize the effect of these outliers and noisy data.

One way of minimizing the contribution of outliers/noise is **dividing the training data into multiple shards (we can call them subsets) and training each shard separately**. That way, each data point will only occur once in one of the several data shards. The representative samples would be close to each other in the feature space and would be present in each of the data shards more or less uniformly. However, the individual outliers/noise would only be able to impact one of the training processes. **After training, aggregating and averaging can be done** to combine the results of all the models. Averaging would mean that the effect of the outliers/ noise would be reduced by a factor of N, where N is the number of data shards. In this case, this technique should reduce the overfitting significantly. There are many layers to this hypothesis. Some of the theoretical reasons why this method should generalize better than the traditional method are discussed below:

1. Weighted averaging of parameters helps in combining the knowledge learned by different subset/edge models. **However, extreme parameter updates driven by noise or outliers in individual models are moderated when averaging across multiple models**. If a model encounters a noisy sample that deviates significantly from the overall data distribution, it might adjust its parameters excessively to fit that sample, leading to overfitting. However, if the training data is divided into subsets and trained separately, even if one model's parameters are influenced by noisy or outlier samples, their impact is diluted when combined with the parameter weights of other models during averaging. As a result, the averaged parameters should reflect a more balanced representation of the underlying data distribution, mitigating the influence of individual noisy or outlier samples. Overfitting occurs when a model learns to fit the noise or idiosyncrasies present in the training data rather than capturing the underlying patterns. By averaging across multiple models trained on diverse subsets, the learning of more generalizable patterns is encouraged while minimizing the impact of noise and outliers. As a result, this method should be able to generalize better to unseen data and be less susceptible to overfitting compared to individual models, as outliers or noisy examples in one

subset are less likely to affect the overall model significantly when averaged with parameters from other subsets.

2. Another important aspect of the D2C method would be that the edge models being trained on separate subsets should be combined centrally. Typically, ensemble methods combine the models by averaging class probabilities. However, if multiple models have the same architecture, the edge models can be combined more effectively by averaging the parameter weights post-training. This is particularly helpful in combatting overfitting since it helps in smoothing the decision boundary and has a regularizing effect.

   Averaging parameter weights regularizes the model at the parameter level. It moderates extreme updates driven by noise or outliers in individual models, leading to more stable and generalizable parameter values. This helps prevent overfitting by discouraging models from fitting to the noise or idiosyncrasies present in their training subsets. By averaging parameter weights, the ensemble should converge to a shared solution that reflects the collective knowledge learned from different perspectives. This consensus learning approach encourages models to learn generalizable patterns and reduces reliance on individual model predictions that are prone to overfitting. This should encourage effective exploration of the model space by combining knowledge learned from diverse perspectives. Each model trained on a different subset of the data can provide a unique viewpoint of the underlying data distribution. Averaging parameter weights across multiple models might help capture the consensus among these diverse perspectives, leading to a more robust and effective decision boundary.

   Averaging parameter weights should smooth out the decision boundary learned by individual models. **Extreme or noisy parameter updates that result in sharp or jagged decision boundaries in individual models are likely to be moderated when combined through averaging**. The ensemble's decision boundary is more likely to be smooth and continuous, which helps in generalizing well to unseen data and reducing the risk of overfitting.

3. When the training data is divided into multiple subsets, each edge model is trained independently on one of these subsets, with each subset containing a distinct portion of the overall dataset. For example, if there are 10 subsets, each edge model will be trained on a different 10% portion of the data, ensuring that no two models see exactly the same training examples.

   Training each edge model on a different subset of the data provides diversity in the training process. Since each subset may contain a different mix of samples, each edge model is exposed to a slightly different perspective of the dataset. This diversity allows each model to capture unique patterns and variations present in its subset, which may not be fully represented in other subsets. Models trained on diverse subsets are more likely to generalize well to unseen data. Weighted averaging of parameters may help combine the knowledge learned by different edge models. By learning from a broader range of training examples, the averaged combination of these edge models becomes more robust and capable of recognizing patterns that are consistent across different subsets of the data. This improved generalization ability enhances the

ensemble's performance on real-world data, including data that may contain variations not present in the training set.

4. The optimal hyperparameters, such as the number of subsets and epochs before each round of central averaging vary by dataset and domain. The appropriate subset count depends on dataset size and class distribution. If the edge model is complex or the subset count is too high, then the overly reduced size and number of samples per class in each subset can result in significant overfitting in edge models and , which in turn can degrade overall performance even after central averaging. This is similar to what happens in federated learning when the client data is minimal, as pointed out by Zhang et al. [23]. When sufficient data exists per class, more subsets can be beneficial, as noise or outliers are diluted by a factor of **N** (number of training subsets).

In summary, we can conclude that dividing the training data into multiple subsets, training each subset separately, and averaging the parameter weights during the training should have a great impact on addressing the problem of overfitting by moderating the effect of noise and outliers and resulting in a decision boundary that is less prone to sharp, unwanted variations. It should be noted that while this method should help, it doesn't guarantee complete immunity to outliers. Extreme outliers can still affect the central model. However, the effect should be much less. Also, dividing the training set into too many subsets can result in very small data sizes.

In light of these discussions, we can formulate a methodology to address overfitting and minimize the effects of noise and outliers. Our plan is to introduce a method that focuses on four important aspects:

- Firstly, it combines models that have exactly the same architecture but use different training data. We can do it by simply dividing the entire training set.

- Secondly, the method does not combine decision or class probabilities. Instead, it individually trains on all the subsets of training data and then periodically shares the parameter weights for aggregating and averaging.

- Thirdly, this sharing and averaging of weights shared from edge models can go on for several epochs. Repeating the process for several global epochs will allow the model to refine its understanding. Each iteration can refine the central model based on the collective knowledge of edge models (the models being trained on the subsets).

- Finally, the number of subsets and the local training process should be optimized through a systematic process so that as much improvement as possible can be generated using the proposed method.

Despite the trade-offs that might exist, the proposed approach will likely be able to address the issue of overfitting effectively.

3.3 Expected Aftermath of Deploying a Method that Addresses Overfitting

We have already laid out the theoretical background behind our hypothesis that the method we are proposing should be more overfitting resistant. However, what can we expect when such a method is employed? Apart from addressing the issue of overfitting, what other trade-offs and constraints should be considered? That is what we will be discussing in this section. Compared to the traditional approach, we can expect some specific changes when we apply a method that addresses overfitting. These changes are important to understand the results once we apply our proposed method. Some of the important effects that such a method is likely to face are:

1. **Improved Generalization Performance:** The most obvious effect, is the very reason for applying such a technique. After applying the method, we should observe better performance on the validation or test dataset compared to before applying it. The performance metric can be different (accuracy, loss, F1 score, AUC-ROC, Matthew's Correlation Coefficient) across domains or datasets, but the improvement must be there.

2. **Reduced Sensitivity to Noise and Outliers:** Overfitting often occurs when models capture noise or outliers in the training data as if they were genuine patterns. A technique that mitigates overfitting should make the model less sensitive to such noise and outliers, resulting in more robust predictions. This should be reflected in the performance metrics. The decision boundary should be smoothed in this process too.

3. **Stability in the Training Process:** Many techniques used to address overfitting, such as dropout, weight regularization, or early stopping, introduce regularization mechanisms that promote stable and more robust model training. As a result, one can expect more stable training dynamics and reduced sensitivity to hyperparameters. This stability can sometimes be observed in the loss curves, with the curves becoming more consistent and stable, showing clearer trends and less fluctuation.

4. **Slow Convergence:** Many overfitting reduction methods come with slower convergence to the optimum solution. For example, Regularization (L1, L2) adds penalty terms to the loss function, making the learning slower and encouraging smaller parameter values. While it helps prevent overfitting, it can slow down convergence because the model adjusts more gradually. Similarly, Dropout randomly deactivates neurons during training, reducing co-adaptation. Again, this slows down the learning. This whole process introduces stochasticity, which can slow down convergence. The averaging process of weights in our proposed method also penalizes learning, which may slow down the convergence to some extent.

5. **Sensitivity to Hyperparameters:** Some Crucial hyperparameters related to the specific techniques play an important role in getting the desired outcome when applying overfitting-reducing techniques. Setting the right value of these hyperparameters becomes very important in most cases. For example, In L2

regularization, the regularization term added to the loss function is proportional to the squared magnitudes of the model weights. The hyperparameter λ controls the strength of regularization, where higher values of λ lead to a more pronounced shrinking of the weights. A value of λ that is too high can lead to severe underfitting, while too small a value can result in a sub-optimal solution where overfitting is not addressed properly. Similar things can be said about the dropout rate. While deactivating too many neurons can hamper effective training, a low dropout rate may mean a very minimal improvement in generalization performance. Our proposed method will have its own set of hyperparameters which will be important to apply this method effectively.

6. **Bias Variance Trade-Off:** Almost any method that addresses overfitting must face a trade-off between bias and variance. While these methods are usually applied to tackle the problem of high variance, not setting the hyperparameters carefully may lead to high bias, subsequently resulting in poor performance. Finding the sweet spot between high bias and high variance is imperative.

7. **Data Dependency:** There is no "One size fits all" scenario when it comes to these techniques. Models trained on datasets from different domains, of different sizes may perform differently under different conditions. This is why tuning the hyperparameters accordingly becomes very important. In our study, we will test our method using several datasets over multiple domains to test for these specifics.

In general, we can say that apart from the obvious expectation of improved performance on unseen data, we can expect several other aftermaths when we apply our method. These things should be considered when evaluating the method and also to determine how to apply the method proficiently.

Chapter 4

Problem Formulation

In this chapter, we formulate the underlying intuition behind our hypothesis. We must first define overfitting formally. We follow the groundwork laid out by Ghojogh et al. [12]. Let us consider a Dataset D with training set T and test set R so that,

$$T \cup R = D \tag{4.1}$$

$$T \cap R = \emptyset \tag{4.2}$$

Let, T consist of datapoints $\{(x_i, y_i)\}_{i=1}^{n}$ and test set R consist the datapoints $\{(x_i, y_i)\}_{i=1}^{m}$. Also, let f be the function that maps each datapoint in the training set to its corresponding output label(true observations), i.e., $f(x_i) = f_i$, where f_i is the true corresponding output label for x_i. This model is unknown and we wish to train a model that replicates f as closely as possible. The problem is both the ground truth model and the true labels are unknown to us. Realistically, there is always some noise in the training dataset. We assume that the training output is corrupted with some additive noise ϵ_i. So, we can say,

$$y_i = f_i + \epsilon_i \tag{4.3}$$

where ϵ_i is some Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Since the mean of the distribution is zero we can say that $E(\epsilon_i) = 0$, and $Var(\epsilon_i) = \sigma^2$. As the variance of a random variable x can be denoted as $Var(X) = E(x^2) - [E(x)]^2$ (see Appendix A), we have,

$$E(\epsilon_i^2) = Var(\epsilon_i) + [E(\epsilon_i)]^2 \Rightarrow E(\epsilon_i^2) = \sigma^2 + 0 = \sigma^2 \tag{4.4}$$

Our assumption is that the input of the training dataset $\{x_i\}_{i=1}^{n}$ and their noise-added outputs $\{y_i\}_{i=1}^{n}$ are available to us, and our goal is to estimate the true model f by a model $\hat{f}$ in order to predict the labels $\{y_i\}_{i=1}^{n}$ for the input data $\{x_i\}_{i=1}^{n}$. Let the estimated observations be $\{\hat{y}_i\}_{i=1}^{n}$. It is preferable that $\{\hat{y}_i\}_{i=1}^{n}$ are close as possible to $\{y_i\}_{i=1}^{n}$. Mathematically, we can consider this as a minimization problem looking to minimize some error function such as MSE (Mean squared error). The MSE of estimated outputs from the trained model with respect to the dataset outputs are: $\textbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = E((\hat{y}_i - y_i)^2)$. Let us consider a single arbitrary datapoint (x_0, y_0) and the corresponding prediction of the trained model $\hat{f}$ for the input x_0 is $\hat{y}_0$. Ghojogh et al. [12] showed that if the sample instance $(x_0, y_0) \notin T$ i.e., it belongs to the test set R, MSE of the predicted label can be expressed as -

$$\textbf{MSE} = E((\hat{y}_0 - y_0)^2) = E((\hat{y}_0 - f_0)^2) + \sigma^2 \tag{4.5}$$

Here we can observe from the first term of the R.H.S. of the equation that the MSE of the estimation of output labels with respect to the dataset labels can actually be represented by the MSE of the estimation with respect to the true uncorrupted labels $\{f_i\}_{i=1}^n$ plus some effect of the noise that exists in the dataset. Taking *Monte-Carlo approximation* [24] of the expectation terms in Eq. (4.5), we have:

$$\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - f_i)^2 + \sigma^2 \tag{4.6}$$

$$\implies \sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \sum_{i=1}^{m}(\hat{y}_i - f_i)^2 + m\sigma^2 \tag{4.7}$$

Here the term $\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$ is the total error between the model predictions and the dataset labels which we can call the *empirical observed error*, e. On the other hand, $\sum_{i=1}^{m}(\hat{y}_i - f_i)^2$ represents the total error between prediction labels and the true unknown labels, namely E. Hence, $e = E + m\sigma^2$. Thus, the observed error truly reflects the true error because the term $m\sigma^2$ remains constant, which is the theoretical justification for evaluating model performance using unseen test data. But, as Ghojogh et al. [12] showed, if $(x_0, y_0) \in T$, deriving the M.S.E yields:

$$\textbf{MSE} = E((\hat{y}_0 - y_0)^2) = E((\hat{y}_0 - f_0)^2) + \sigma^2 - 2\sigma^2\, E(\frac{\partial \hat{y}_0}{\partial y_0}) \tag{4.8}$$

Using the same approximation as Eq.(4.7) on the training set T, we get:

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - f_i)^2 + \sigma^2 - 2\sigma^2\frac{1}{n}\sum_{i=1}^{n}\frac{\partial \hat{y}_i}{\partial y_i} \tag{4.9}$$

$$\implies \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \sum_{i=1}^{n}(\hat{y}_i - f_i)^2 + n\sigma^2 - 2\sigma^2\sum_{i=1}^{n}\frac{\partial \hat{y}_i}{\partial y_i} \tag{4.10}$$

From Eq.(4.10) we now understand while the model is training the observed empirical error calculated on the training dataset i.e, training error is not a clear representation of the true error because now $e = E + n\sigma^2 - 2\sigma^2\sum_{i=1}^{n}\frac{\partial \hat{y}_0}{\partial y_0}$. Hence, even if we are able to obtain a small value of e, the term $2\sigma^2\sum_{i=1}^{n}\frac{\partial \hat{y}_0}{\partial y_0}$ can grow large as the training progresses and hide the true value of a substantial and large true error E (Figure 4.1). As we can see in the figure, the model starts to overfit as empirical error becomes no longer a true representation of true error when the corresponding complexity term grows and dominates the expression in Eq.(4.10) for empirical error. We can conclude from this analysis that the final term in Eq.(4.8) is a measure of the overfitting of the model. Which is also known as the complexity of a model. Upon a closer look at this final term, we can derive some more interesting insight. $\frac{\partial \hat{y}_0}{\partial y_0}$ is essentially the rate of change of the predicted label $\hat{y}_i$ for the input x_i with respect to the change in dataset label y_i. It makes sense that prediction labels would vary if the corresponding output in the dataset varies as the sample is considered from within the training set. It also reflects how dependent the model is on the training set, as the term grows when the predicted output varies too much with respect to changes in the training samples. This fits the classical definition of

overfitting i.e., fitting too tightly to the training data and thus performing way worse when evaluated with actual test data. Evaluation on test data accurately represents the true error from Eq.(4.7). Figure 4.2 depicts a visualization of this idea.

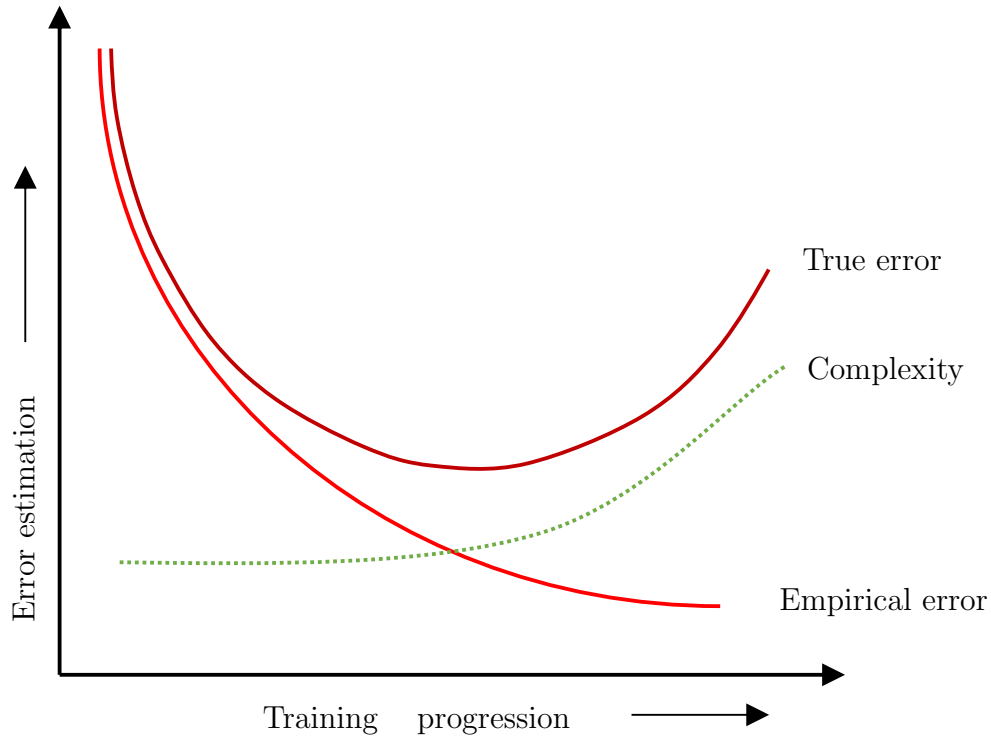

Figure 4.1: As the corresponding complexity term grows and dominates the expression for empirical error, the model starts to overfit.

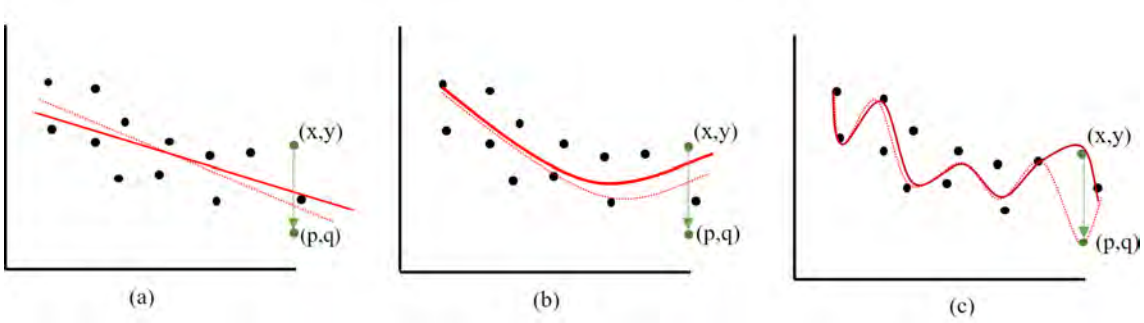

Figure 4.2: A visualization of (a) Underfitting, (b) A good fit, and (c) Overfitting scenarios, illustrating how the model changes if an arbitrary training data point (x, y) is shifted to (p, q) by plotting the trained model before(solid line) and after(dashed line) the shift.

Statistical analysis shows that the distinction between an underfitting model, an overfitting one, and a good fit can be reflected by their bias and variance with respect to the training labels. Ideally, a model with low variance and low bias is the most desirable and usually deemed a good fit. However, during the training phase, sometimes we see a model has low variance and high bias which indicates the model predictions are not good as well as being all over the place, this is the indication of an underfitting model. It usually happens when the model is too simple to capture

the complexity of the data or has not been trained for long enough to reach convergence. On the other hand, we find a model which overfits has high variance and low bias. This is because the model becomes overly complex accommodating for all the outliers in the training data and fails to generalize on unseen test data. Figure 4.2(a) and 4.2(c) show how an underfitting and overfitting model shows low and high variance respectively. Figure 4.2 is a visualization of (a) Underfitting, (b) A good fit, and (c) Overfitting scenarios. The figures illustrate how the model changes if an arbitrary training data point (x, y) is shifted to (p, q) by plotting the trained model before and after the shift. In (a) the underfitted model the change is very minimal. In the (b) Good fit model the change is also relatively little, but in the (c) Overfitted model there is a large shift in the neighboring region of that data point, which reflects the over-dependence on training data.

To see the impact of **D2C** we consider the training set T divided into k equal partitions, namely $|T_1| = |T_2| = ... = |T_k| = |T|/k$. Then, we train the model $\hat{f}_j$ using the j-th data chunk, where $j \in \{1, .., k\}$. As a result, we have k separately trained models. Finally, we aggregate these edge models into one central model $\hat{f}$. This process is repeated for p global epochs. For the sake of simplicity, we consider each $\hat{f}_j$ is a regression model with weight matrix $\boldsymbol{W_j} = \{w_{j1}, ..., w_{jn}\}$ and bias b_j. For an input vector $\boldsymbol{x} = \{x_1, ..., x_n\}$ the prediction of the edge model on the t-th epoch can be described as: $\hat{f}_j(\boldsymbol{x}) = \boldsymbol{W_j}^t \boldsymbol{x}^T + b_j$. In the case of a single global epoch, we now wish to aggregate the weight matrices creating the central weight matrix $\bar{\boldsymbol{W}} = \{\bar{w}_1, ..., \bar{w}_n\}$. Here, the weight matrix for the $(t+1)$-th epoch is:

$$\bar{\boldsymbol{W}}^{t+1} = \frac{1}{k} \sum_{j=1}^{k} \boldsymbol{W_j}^t \text{ and } b^{t+1} = \frac{1}{k} \sum_{j=1}^{k} b_j^t \tag{4.11}$$

Hence, the output of the central model at $(t+1)$-th epoch:

$$\begin{aligned}
&\hat{f}(\boldsymbol{x})^{t+1} = \bar{\boldsymbol{W}}^{t+1} \boldsymbol{x}^T + b^{t+1} \\
&\implies \hat{f}(\boldsymbol{x})^{t+1} = \frac{1}{k} \sum_{j=1}^{k} \boldsymbol{W_j}^t \boldsymbol{x}^T + \frac{1}{k} \sum_{j=1}^{k} b_j^t \\
&\implies \hat{f}(\boldsymbol{x})^{t+1} = \frac{1}{k} \sum_{j=1}^{k} \left(\boldsymbol{W_j}^t \boldsymbol{x}^T + b_j^t \right) \\
&\implies \hat{f}(\boldsymbol{x})^{t+1} = \frac{1}{k} \sum_{j=1}^{k} \hat{f}(\boldsymbol{x})_j^t
\end{aligned} \tag{4.12}$$

Let e_{jt} denote the empirical error of the j-th model while estimating some arbitrary instance on the t-th global epoch. We assume this error to be a random variable with normal distribution having mean zero, i.e., $e_{jt} \sim \mathcal{N}(0, s)$. Hence, $E(e_{jt}) = 0$ and $Var(e_{jt}) = s$. Let us denote the estimation of the product of two trained models using two different data chunks by c, i.e, $E(e_{jt} \times e_{lt}) = c$. So, we have:

$$E(e_{jt}^2) = Var(e_{jt}) + [E(e_{jt})]^2 \Rightarrow E(e_{jt}^2) = s + 0 = s \tag{4.13}$$

Again for $j, l \in \{1, ..., k\}$, where $j \neq l$, we have (see Appendix A for the proof of the used theorem):

$$\begin{aligned} &Cov(e_{jt}, e_{lt}) = E(e_{jt} \times e_{lt}) - E(e_{jt})\, E(e_{lt}) \\ &\implies Cov(e_{jt}, e_{lt}) = c - 0 \times 0 = c \\ &\implies Cov(\hat{f}(\boldsymbol{x})^t_j, \hat{f}(\boldsymbol{x})^t_l) = c - 0 \times 0 = c \end{aligned} \tag{4.14}$$

From Eq.(4.12) and Eq.(4.14), we can deduce (see Appendix A for the proofs of the used theorems):

$$\begin{aligned} Var(\hat{f}(\boldsymbol{x})^{t+1}) &= Var\left(\frac{1}{k}\sum_{j=1}^{k}\hat{f}(\boldsymbol{x})^t_j\right) \\ &= \frac{1}{k^2} Var\left(\sum_{j=1}^{k}\hat{f}(\boldsymbol{x})^t_j\right) \\ &= \frac{1}{k^2}\sum_{j=1}^{k} Var\left(\hat{f}(\boldsymbol{x})^t_j\right) + \frac{1}{k^2}\sum_{i=1,l=1}^{k}\sum_{i\neq l}^{k} Cov\left(\hat{f}(\boldsymbol{x})^t_i, \hat{f}(\boldsymbol{x})^t_l\right) \\ &= \frac{1}{k^2} sk + \frac{1}{k^2} ck(k-1) = \frac{s + c(k-1)}{k} \end{aligned} \tag{4.15}$$

Eq.(4.15) provides us with an interesting insight. If the two edge/subset models are highly correlated i.e., their predictions and empirical error are very close to each other we can assume $s \approx c$. Hence, the variance of the central model at $(t+1)$-th epoch can be reduced to: $Var(\hat{f}(\boldsymbol{x})^{t+1}) \approx (s + s(k-1))/k \approx s$. On the other hand, if the two models are very dissimilar the same expression gives us: $Var(\hat{f}(\boldsymbol{x})^{t+1}) \approx (s + 0(k-1))/k \approx s/k$. As $Var(e_{jt}) = Var(\hat{f}(\boldsymbol{x})^t) = s$, each of the edge models at epoch t has a variance of prediction s, which means if the edge models at epoch t are completely different, the variation of estimate decreases by a factor of k on epoch $(t+1)$. Consequently, if the edge models are identically trained the variance remains almost unchanged. This result is similar to the one proposed by Breiman et al. [25] and Bühlmann et al. [26] through the idea of *'Bagging'*, which is a meta-algorithm aggregating multiple model outputs. The analysis clearly shows that our approach does not at the least worsen the problem of overfitting in this scenario but rather improves the central model on each epoch if certain conditions are met. However, in reality, the edge models are neither completely different nor exactly the same due to the relative distribution of the data subsets and their shared model architecture. So, the degree to which D2C ameliorates overfitting depends on the model architecture and the correlation among the edge models. However, it is easy to see that if the training data is contaminated by outliers and random noise, the learned parameters from the different portions of the data are more likely to be different from each other. This is because the individual outliers and noise are not highly correlated, hence distributing them among the subsets likely results in significantly different samples in the subsets, making the subsequent models different from each other. Our results show that repeating the process in every global epoch reduces overfitting in most cases. A limitation of this analysis is that deep neural networks are mostly black boxes and their architectures vary tremendously, which makes establishing a generalized mathematical representation

of them extremely difficult. Hence, our assumption of a simple regression model is not an exact reflection of the actual implementation. Despite that, it describes an ideal and simple scenario where the proposed method should improve the model. Moreover, it also gave us the insight that this approach can be used in conjunction with other overfitting reduction techniques such as *Dropouts*(Srivastava et al., 2014) [8]. If dropouts are stacked on top of our method in every iteration of local training, the neurons are randomly removed with some probability p(usually $p = 0.5$). This introduction of randomness ensures that edge models are different from each other increasing the chances of improvement when the models are aggregated.

Chapter 5

The Divide2Conquer (D2C) Method

The Divide2Conquer(D2C) method adds a few more steps on top of traditional neural networks. These steps consist of data distribution among the subsets of the training data and performing central averaging at the end of each global epoch. We discuss each of the steps briefly in this chapter.

5.1 Creating Training Subsets

To train multiple Neural Networks parallelly using different subsets of the data, we need to divide the whole training set into multiple shards, and **we also need multiple identical models that will be used to train the different subsets. These models will be denoted as Edge models or Subset Models in this thesis**. At first, random shuffling is applied to the training set. After that, it is divided into multiple subsets. This is done while keeping the class distribution constant. Now, each of these subsets is fed into an Edge Model. All the edge models must have precisely the same neural network architecture, hence the same number of trainable parameters. Otherwise, central averaging would not be possible.

5.2 Training and Averaging of Trained Parameters

Instead of one training loop, the implementation of our method requires two loops. The subset training loop is nested within the global iteration loop for averaging. The process is started by creating a central model with the same architecture as the models used to train each subset. Initially, identical model objects are built and loaded for each training subset. These are the Edge models or Subset Models. **So, in total, we have one Central Model and K Edge models, all of which share an identical architecture, where K is the number of subsets we have divided the training set into.** Then, the weights of the central and edge models are initialized. Inside the inner loop, the subset training is repeated iteratively for each training subset. Then, after a particular number of local epochs, the trained parameters from all the edge models are scaled and then added to the

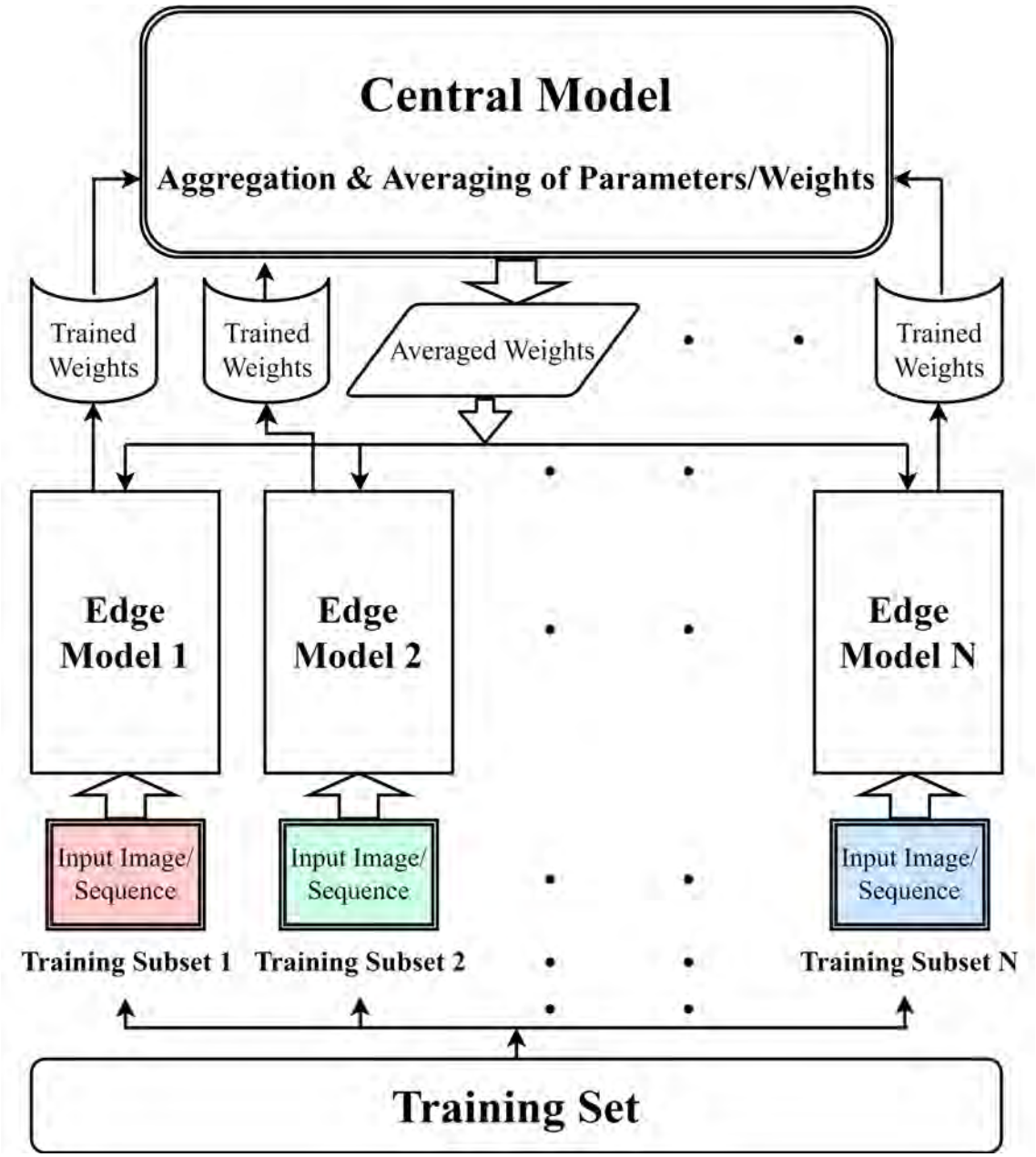

Figure 5.1: D2C at a Glance – Training on each subset, averaging the trained parameters, setting the Central model parameters to these values, and sharing the averaged parameters back to the Edge models to keep the loop running.

local weight list to send to the outer loop, where they are aggregated and averaged. Each edge model trains on its training subset for a particular number of local epochs within each global iteration. This followed by the sharing of subset weights to the central model, concludes local training. All the scaled edge model weights are added up in the outer loop, and weighted aggregation and averaging are performed for each of the parameters. Finally, the central model is modified, and the parameters are updated to these averaged aggregates. At this stage, the initialization phase weights of each edge model are set to the central model's current parameters. An entire global training epoch thus concludes. The next step is to retrain each edge model on the respective training subset. This whole process is then repeated

for several global iterations (global iterations are also denoted as **communication rounds/global epochs** in this study). The averaging process here is simply the execution of weighted parameter averaging. The weights are measured based on the fraction of data instances belonging to each training subset. The weighted averaging formula we're utilizing is given below.

$$f(\omega) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\omega) \text{ where, } F_k(\omega) = \frac{1}{n} \sum_{i \in Pk}^{i} (f_i(\omega)) \tag{5.1}$$

The whole process is summarized in Figure 5.1. Let us take a look at the pseudo-code to implement the whole process below.

Algorithm: Divide2Conquer

function Divide2Conquer(D, N, E, E_{global}, B, lr)
Input:
D: Training dataset
N: Number of subsets
E: Number of local epochs in each subset before central averaging
E_{global}: Number of global epochs
B: Batch size
lr: Learning rate
Output: Central model M_c with aggregated weights

1. Shuffle the dataset D.
2. Divide D into N subsets: $\{D_1, D_2, \ldots, D_N\}$, ensuring class distributions remain constant.
3. Initialize central model M_c with parameter vector θ_c.
4. **for each** subset $i = 1$ **to** N **do**
   (a) Initialize edge model M_i with parameter vector θ_i, where $\theta_i = \theta_c$.
   (b) Set scaling factor $s_i = \frac{|D_i|}{|D|}$, the fraction of the dataset assigned to subset i.
5. **end for**
6. **for each** global epoch $g = 1$ **to** E_{global} **do**
   (a) Initialize weight accumulator $W_{central} = 0$.
   (b) **for each** subset $i = 1$ **to** N **do**
      i. Set $\theta_i = \theta_c$ (initialize edge model M_i with central model's weights).
      ii. **for each** local epoch $e = 1$ **to** E **do**
         A. Train M_i on subset D_i for one epoch using batch size B and learning rate lr.
      iii. **end for**

iv. Obtain updated weights θ_i from M_i after training.

v. Scale weights: $\theta_i = s_i \cdot \theta_i$.

vi. Accumulate scaled weights into $W_{central}$: $W_{central} = W_{central} + \theta_i$.

(c) **end for**

(d) Compute averaged central weights:

$$\theta_c = \frac{1}{\sum_{i=1}^{N} s_i} W_{central}.$$

(e) Update central model M_c with new weights θ_c.

7. **end for**

8. **return** central model M_c.

end function

5.3 Global Hyperparameter Tuning

Tuning the overall model consists of two stages. Firstly, the edge model architecture and its hyperparameters should be tuned using a subset/entire training set and the validation set to find a suitable model to train each training subset. Next, that base model is utilized to tune the Central model. The new global hyperparameters that need tuning, in this case, are the **Number of Subsets of the Training Set** and the **Number of Epochs before each round of Central Averaging**. Testing for different values for these hyperparameters, the implementation and the evaluation of our method are done using the best combination. The appropriate number of training subsets can be determined first by varying the **number of training subsets** and observing the performance metrics like test/validation accuracy, F1-score, AUC-ROC, log loss, and also the validation loss curves. The appropriate number of training epochs in each subset before the averaging is done each time is likely to be a small number ($1 \sim 2$) since the edge model can rather quickly overfit the training subset with a reduced size. So, it makes more sense to set a small number of epochs initially and tune the number of training subsets. Once the number of subsets is determined, the same process can be repeated by varying the **number of training epochs** and determining the appropriate value of it.

It should be noted that the appropriate number of subsets and epochs can vary based on the dataset and the domain. Typically, a small number of epochs should result in a more overfitting-resistant model at the expense of more intense training locally. However, the appropriate number of subsets can vary a lot based on the size and the class distribution of the training data. If the edge model architecture is too complex and the number of training subsets is too high, then the overly reduced size and number of samples per class in each subset can result in significant overfitting in edge models, which in turn can degrade overall performance even after central averaging. However, when there is a sufficient amount of data available for each class, the dataset can be partitioned into more subsets. Intuitively, this should help in reducing overfitting, since, as discussed in the previous chapter, the effect of noise

or outliers is reduced by a factor of **N** using our method, where **N = Number of Training Subsets**. This is where other methods to address overfitting like Data Augmentation and Dropout can be useful, as our method can easily be used on top of these.

Chapter 6

Experimental Setup

We implemented our method on three different domains using datasets for classification tasks. They are:

1. **Computer Vision** (Image Classification)
2. **Speech** (Audio Classification)
3. **Natural Language Processing** (Text Classification)

6.1 Datasets

For **image classification**, we used three popular benchmark datasets in CIFAR-10, MNIST, and Fashion MNIST. CIFAR-10 [27] is a widely used benchmark dataset for image classification tasks. It comprises 60,000 RGB images, each with a resolution of 32x32 pixels, distributed across 10 classes. The dataset is split into 50,000 training images and 10,000 test images, with each class containing 6,000 and 1,000 images, respectively. The classes in CIFAR-10 represent common objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. MNIST [28] is a well-known and extensively studied dataset in the field of machine learning. It consists of 70,000 grayscale images of handwritten digits (0-9), each with a resolution of 28x28 pixels. This dataset is divided into 60,000 training images and 10,000 test images. Fashion MNIST [29] was created as a drop-in replacement for the extensively used MNIST, offering a more challenging alternative for benchmarking machine learning algorithms. It also consists of 70,000 grayscale images depicting fashion items such as clothing, shoes, and accessories, each with a resolution of 28x28 pixels. Just like MNIST, the dataset is split into 60,000 training images and 10,000 test images, with 10 classes representing different fashion categories. CIFAR-10 is the most challenging dataset among the three, suitable for evaluating algorithms in complex image classification tasks. MNIST offers a simpler dataset with grayscale images of handwritten digits, commonly used for basic machine-learning demonstrations. Fashion MNIST provides a middle ground between simplicity and complexity. We also focused on a particularly overfitting-prone task, Facial expression recognition. We used the FER-2013 [30] dataset for this. FER-2013 is a large dataset containing more than 35000 training images. It consists of grayscale images depicting facial expressions of seven basic emotions: anger, disgust, fear, happiness,

sadness, surprise, and neutral. This is mostly a balanced dataset containing pictures taken from different angles. The dataset's diversity and quality vary due to its internet-based collection method, which may introduce noise and variability in the images. That is why this dataset is particularly important to our study, which aims to minimize overfitting caused by the mentioned issues and improve the generalization performance.

For **audio classification**, we combined three different datasets to create a combined dataset containing 14 classes (7 basic emotions for genders male and female each). We used TESS [31], CREMA-D [32], and RAVDESS [33] datasets for this task. The CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset) dataset consists of audio and video recordings of actors portraying various emotions in controlled environments, providing rich data for studying emotional expressions. RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) contains audio-visual recordings of actors performing scripted speech and song excerpts while conveying different emotions, enabling research on emotion recognition in both speech and music. TESS (Toronto Emotional Speech Set) comprises audio recordings of professionally trained actors uttering short sentences with neutral and emotional content, offering a standardized dataset for studying emotional speech perception. Combining the datasets gave us a sufficiently large dataset for evaluation. By merging these datasets, we could create a more diverse and comprehensive dataset that encompasses a wider range of emotional expressions, speech styles, and environmental conditions. Also, different sources meant more diversity in both the training set and the test set, which is crucial in determining the generalization performance of the models. It also helps mitigate biases and limitations inherent in individual datasets, leading to more robust and generalizable models. We also used the larger UrbanSound8K [34] for further analysis. This dataset is divided into 10 folds containing related images within each fold, something that gave us an interesting perspective, which we will discuss in the next chapter.

For **text classification**, we used another benchmark dataset, AG NEWS[35]. It is a widely used benchmark dataset for text classification tasks. It consists of news articles collected from the AG's corpus news collection, covering four major categories: World, Sports, Business, and Science/Technology. The dataset contains approximately 127,600 news articles. This dataset is perfectly balanced in terms of class distribution. Its large size, balanced distribution of categories, and relevance to real-world news content make it the perfect candidate for evaluating the generalization ability of models employing our proposed method. This dataset is crucial to testing our method in Big Data scenarios.

6.2 Evaluation Method

To evaluate our proposed D2C method, the direct comparison would be between the performances of the base Neural Network architecture used for the edge models using the entire training data and the performance of the model designed based on the D2C method of using training subsets and central averaging. This comparison gives us a proper indication of whether this novel technique of ours shows an improvement over traditional approaches when it comes to resisting overfitting. A similar evaluation

approach was used in one of the previous works on ensemble models [19] and also while proposing Dropout [8]. The primary objective of our proposed method is to address Overfitting. In this light, the performance metrics we will be prioritizing are:

- **Loss & Accuracy Curve:** A definitive indicator of overfitting. A promising sign would be a non-increasing validation loss curve with an increasing number of global epochs for the model based on our proposed method, even if the loss starts increasing in the case of the base neural network architecture trained the entire training set. Even if the loss curve shows an upward trend, if our method manages to slow down the rate at which the loss increases, it necessarily indicates a reduction in overfitting. For this reason, we will monitor the validation loss curves to evaluate the effectiveness of our method in reducing overfitting. However, the loss curve is not the sole indicator of good or bad performance. Loss and accuracy are sometimes not so linearly correlated, even though it seems counter-intuitive. Often, the accuracy keeps improving despite the loss curve showing an increasing trend. For that reason, we will also keep track of the corresponding validation accuracy curve.

- **Test Accuracy, F1-Score, MCC:** Most of our datasets are balanced. So, accuracy should be a good metric. But we will also keep track of the F1 score to account for any imbalance that might be present in our datasets. For similar reasons, we will also keep track of Matthew's Correlation Coefficient (MCC). Ultimately, loss alone can't represent the performance of a model. Better accuracies can be achieved even at the expense of some overconfident predictions affecting the loss value (from outliers), but that doesn't always mean the model's performance is bad.

- **Log Loss, AUC-ROC:** Another important metric though, is the log loss, especially in our study. This is one of the most important indicators of good generalization of the model, and the model that prevents overfitting better should have a lower validation/test loss after the model converges. Log loss provides a comprehensive and interpretable measure of classification performance, taking into account both the accuracy and confidence of predictions. A lower log loss indicates that the model's predicted probabilities align better with the true class labels. This means the model is more confident and accurate in its predictions, even if the accuracies are similar. A very good indicator of robust and accurate predictions in terms of output probabilities is the Area under the ROC Curve (AUC-ROC). So we will also keep track of AUC-ROC to compare and evaluate our method.

- **Decision Boundary:** In classification tasks, the smoothness of the decision boundary created by the model represents the robustness of the model and is a sign of good generalization. On the other hand, complex, highly inconsistent decision boundaries usually indicate overfitting. So we will also observe the change in decision boundaries before and after applying our proposed method, to assess whether there is a sign of better generalization in the decision boundary when we apply the D2C method.

6.3 Experimental Details

- **Image Classification:** At first, all the images are reshaped and normalized as part of Data Preprocessing. We applied one-hot encoding to the labels to use cross-entropy loss later. We split the data into training and validation sets using Scikit-Learn and applied a 90-10 split. We then divided the training set into multiple subsets. Before constructing the model, processing each training subset into tensors, and batching them was our last step. Our edge model is comprised of 4 Convolutional (32 to 256 kernels progressively) and MaxPooling layer (2X2) blocks followed by one fully connected hidden layer (128 neurons). Each block contained two convolutional and one pooling layer. Batch normalization was applied after each Conv layer, and dropouts were used after each block. We used the Adam optimizer, and the learning rate was 0.001. The edge model/subset model summary is shown in Figure 6.1, and the central averaging process is shown in Figure 5.1.

- **Audio Classification:** At first, we combined the paths of the files of our three datasets into a single CSV file. Then we read the .wav files as time series and applied white noise. Then we converted the 1D time series into 2D Log Mel Spectrograms to capture the audio signals' time and frequency domain features. We treated the resulting data as 2D images, and then the remaining steps were the same as we mentioned in the case of the Image classification tasks.

- **Text Classification:** In this case, we created a corpus using all the words that occur in our entire training dataset. Then we made a word index and word embedding for all the words in the corpus. After that, we created word sequences using that word embedding for each data instance. We applied padding to make each training data point equal in length (80 in this case). In the case of 1D text sequences, we couldn't use the CNN structure we used in previous cases. So, the edge model we used, in this case, consisted of 3 bidirectional LSTM layers and one dense layer followed by the output layer. It took the embeddings as input, and we used the 100-dimensional version of GloVe from Stanford for these embeddings. We used the Adam optimizer, and the learning rate was 0.001. The edge model/subset model summary for text classification tasks is shown in Figure 6.2.

For our experiments, in all cases, we used a 90-10 Training – Validation split and used the readily available test sets provided with the datasets. We used dropouts and batch normalization between the layers in all these models. This is also important because as we mentioned earlier, the Divide2Conquer method can be used on top of other methods that address overfitting.

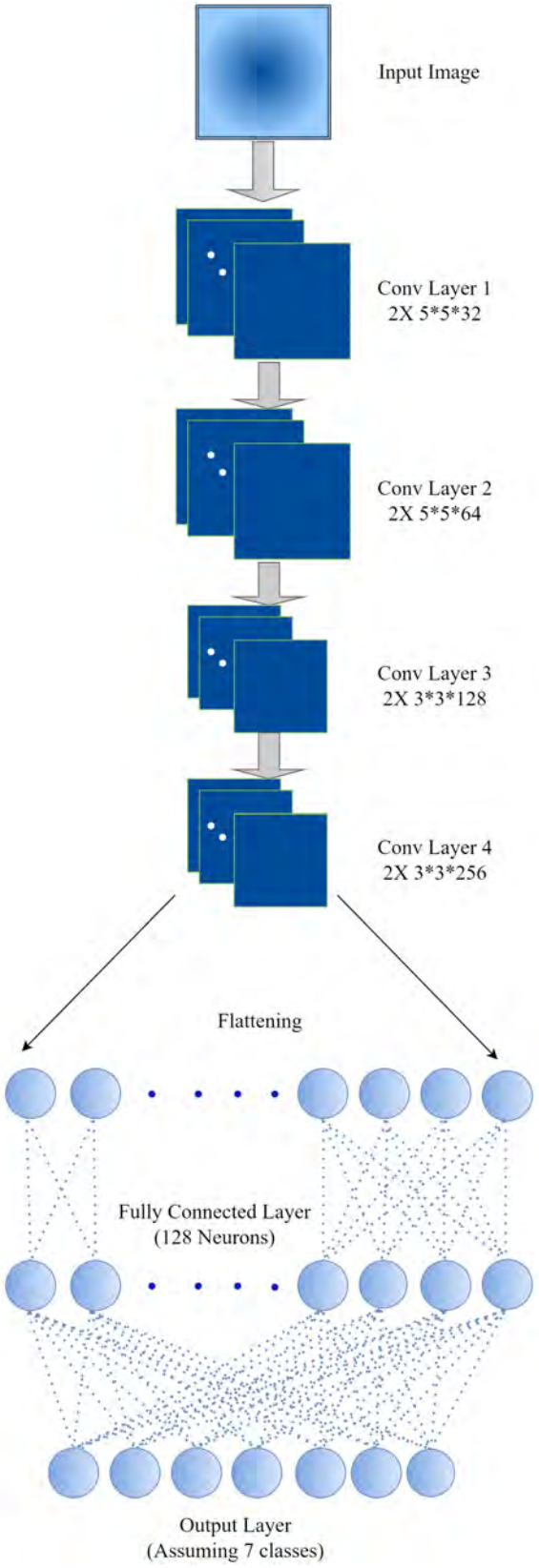

Figure 6.1: Edge Model Architecture that Trains each Training Subset for image and audio classification datasets.

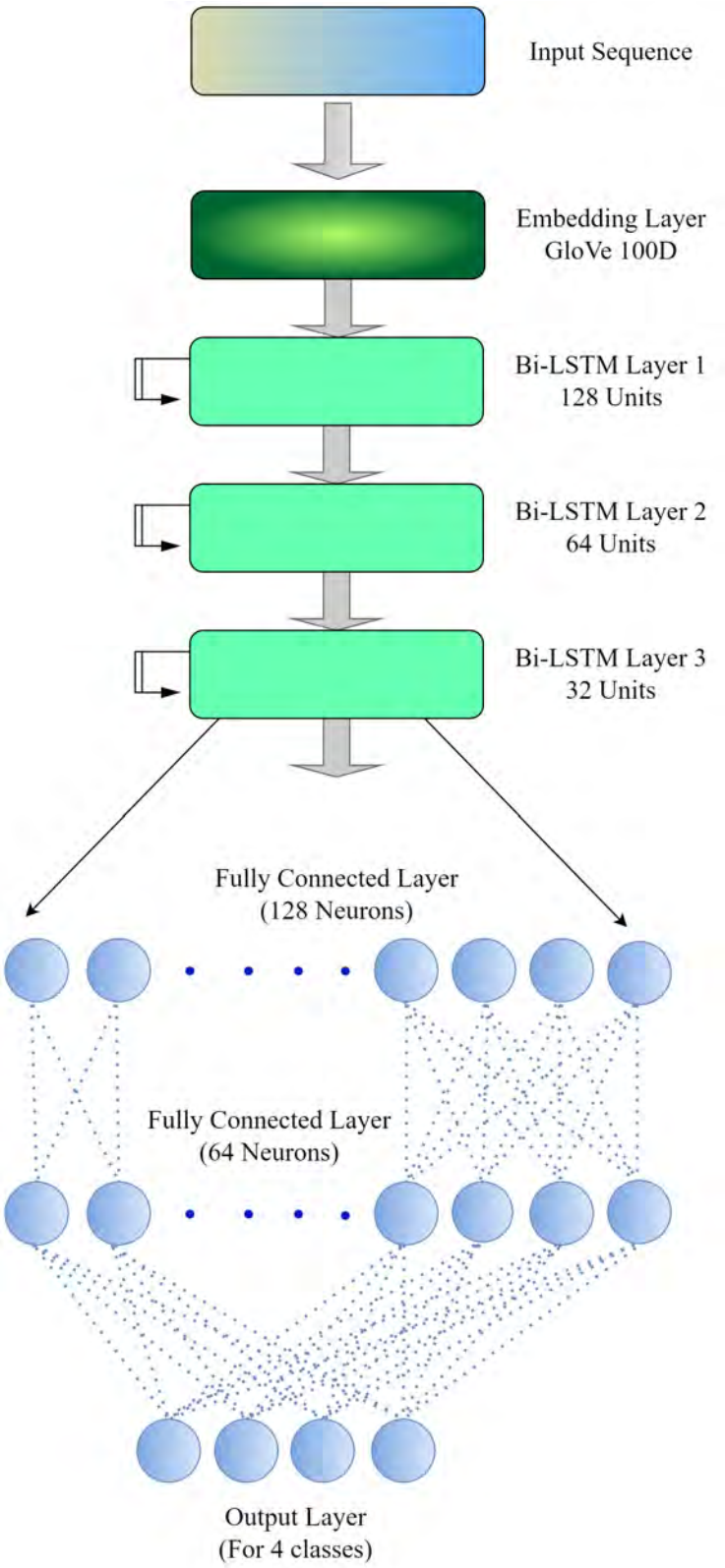

Figure 6.2: Edge Model Architecture that Trains each Training Subset for the text classification dataset.

Chapter 7

Results & Discussion

As discussed in the previous chapter, we will use loss and accuracy curves, Test Accuracy/F1 Score, and log loss to analyze the results and evaluate our method. We also varied the two new global hyperparameters we talked about in chapter 5: the **Number of Subsets of the Training Set** and the **Number of Epochs before each round of Central Averaging**. To refer to them concisely, we will use the variables $\boldsymbol{N}$ and $\boldsymbol{E}$ respectively. This means, in the tables, figures, and discussions of this chapter, $\boldsymbol{N}$ will refer to the Number of Training Subsets, and $\boldsymbol{E}$ will refer to the Number of Epochs before each round of Central Averaging.

7.1 Visualizing Decision Boundary

At first, we will visualize the change in decision boundaries due to applying the D2C method. As we discussed earlier, in the case of overfitting, the decision boundary often becomes overly complex and jagged. Applying a method that addresses overfitting should result in a smoother decision boundary. To test our method, we used a simple binary classification dataset. This dataset was synthetic and created using the Scikit-Learn library. Then the 240-sample large dataset was divided into a training and a test set using a 75-25 split. The neural network architecture we used for these experiments comprised of three hidden layers having 100 neurons each. We did not use any other method that treats overfitting in this case. This helped us understand the difference made by our proposed method alone.

To begin with, we followed the traditional approach and fed the entire training dataset to the model. After training the model for a certain number of epochs, we got the decision boundary shown in Figure 7.1.

As we can see from 7.1, the decision boundary in this plot is highly complex and non-linear. The model creates intricate regions to distinguish between the two classes. The boundary closely follows individual points, resulting in a jagged, convoluted shape. The complexity of the decision boundary suggests that the model is overfitting to the training data. It is trying too hard to separate every point, including noise, rather than focusing on general patterns in the data. Evidence of overfitting is seen in how the model creates small pockets of blue in the red region and vice versa. This indicates the model has likely memorized the data rather than learning generalizable patterns. This overfitting means that this approach is likely to

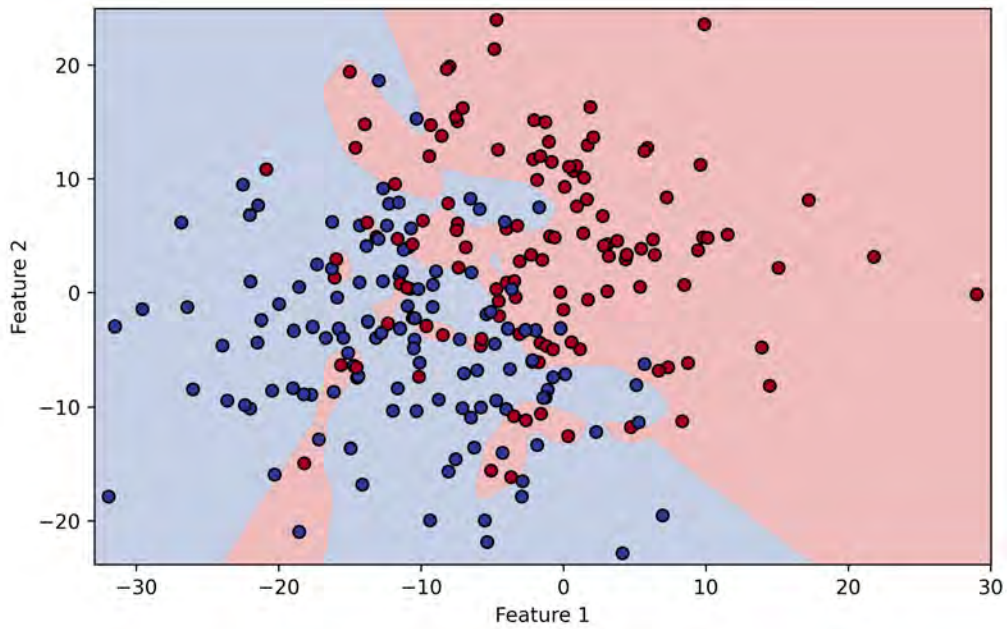

Figure 7.1: Decision Boundary using the traditional approach.

produce very good results on training data, but it is likely to struggle with new, unseen data due to its sensitivity to specific details in the training set. The highly irregular boundary reflects poor generalization ability. The accuracies we got using the training set and the test set also depict the same picture. The training accuracy using this traditional approach was 99.44%, but the test accuracy was just 68.33%, showing significant overfitting. Applying the Divide2Conquer method meant we had to divide our training set into multiple subsets. Then each of the subsets was trained parallelly and after a few epochs, the trained weights from each subset were averaged to get the final weights. Before moving on to the final decision boundary, let us visualize the decision boundaries generated for each training subset in the following figure. We divided the training set into 3 subsets in this case.

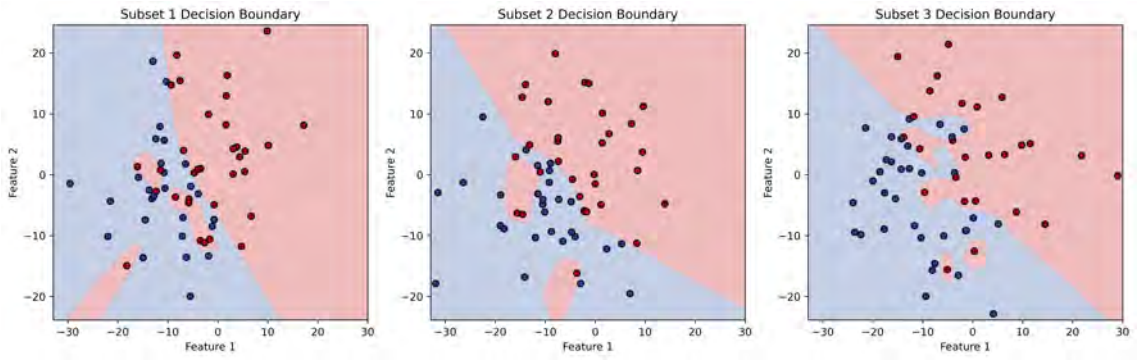

Figure 7.2: Decision Boundary for each training subset after dividing the training data into 3 subsets (N=3).

As we can see from Figure 7.2, the decision boundaries are still complex containing intricate regions following individual noisy data points. However, each of these noisy samples occurs only once in one of the subsets. So, irregularity around the noisy sample does not affect all the subsets, but rather only one of them. Hence, after

averaging the weights, the impact of the individual outlier is minimized. This is true for all the outliers/noisy samples in the training set, so dividing the whole training set means all the noise, and outliers have less impact on the overall model after the averaging is performed. So, the final decision boundary should be smoother and more generalizable when it comes to unseen data. Let us take a look at the decision boundaries we got using different numbers of subsets of the training data.

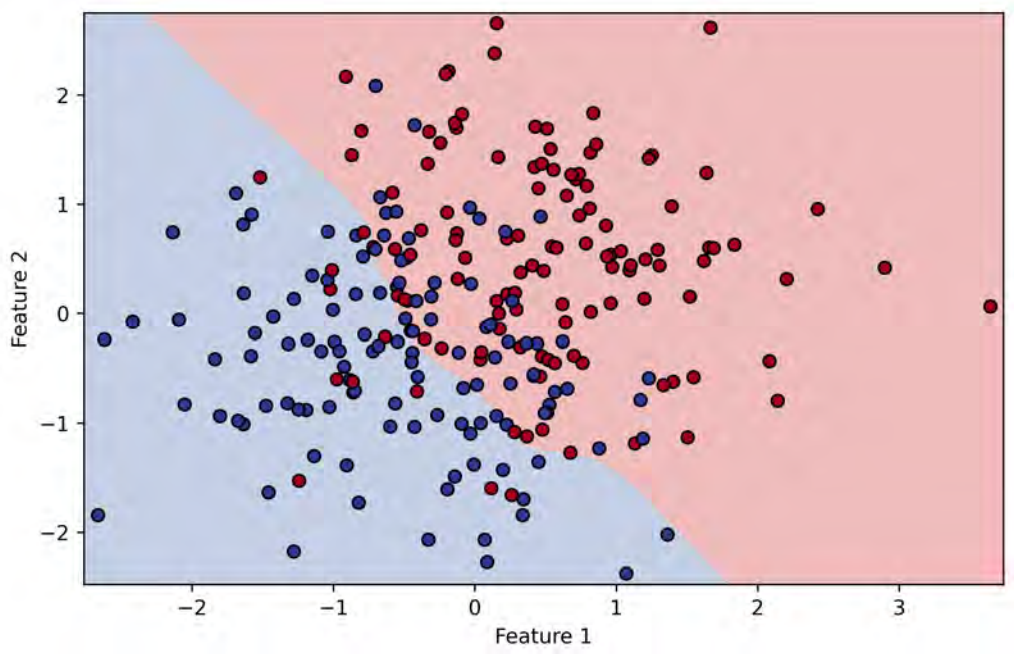

Figure 7.3: Decision Boundary using the D2C method for 2 subsets.

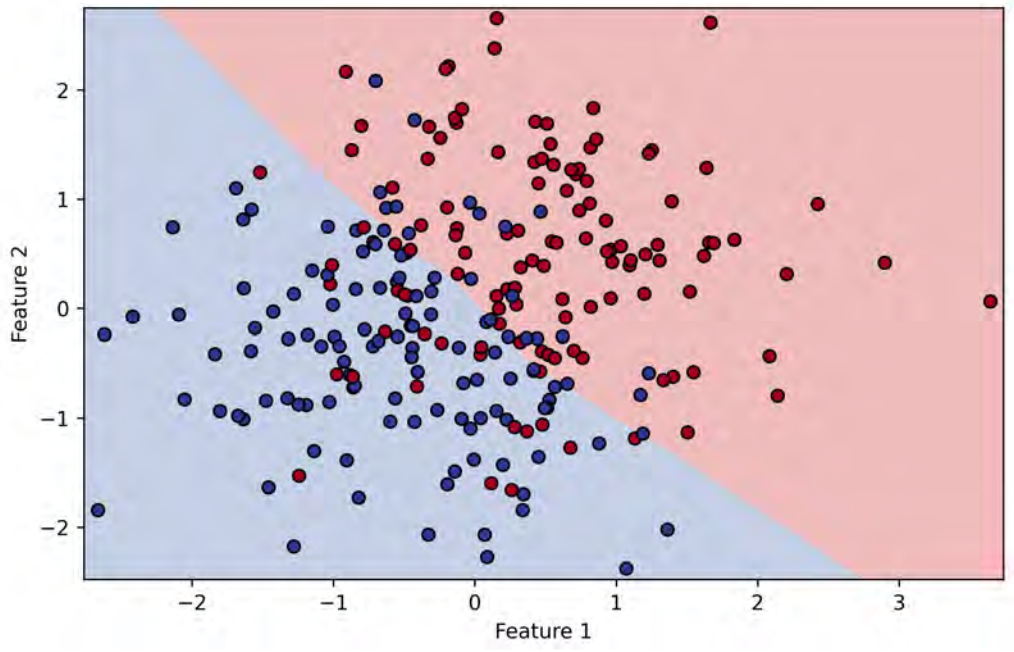

Figure 7.4: Decision Boundary using the D2C method for 3 subsets.

As can be seen from Figures 7.3, 7.4, and 7.5, the decision boundary is much simpler, dividing the feature space almost diagonally. This suggests a less flexible,

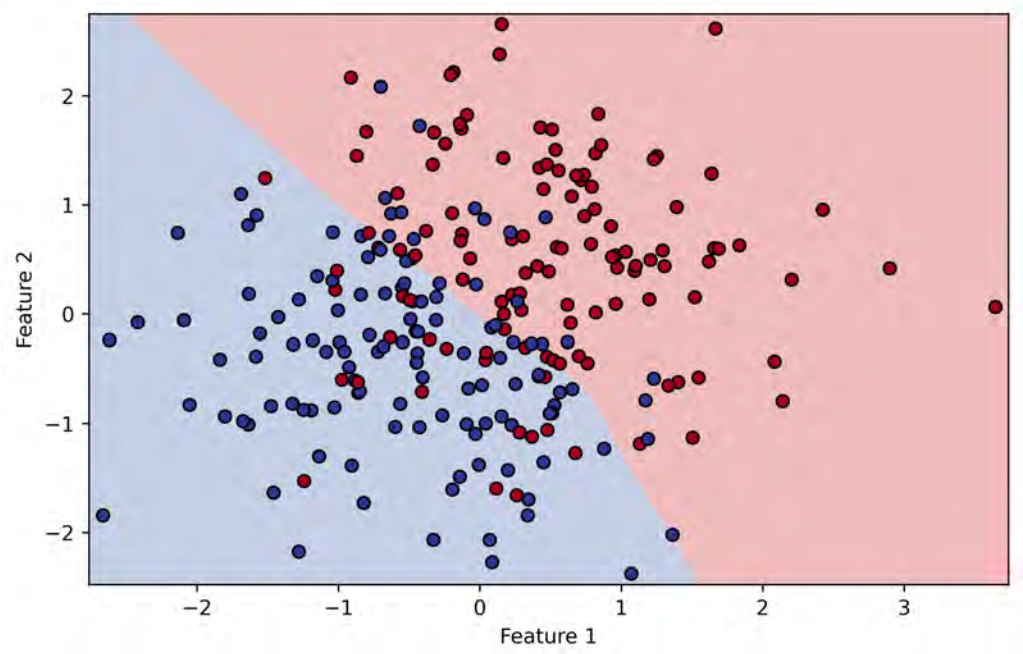

Figure 7.5: Decision Boundary using the D2C method for 4 subsets.

more generalized model, due to averaging the weights from multiple subsets. These boundaries are smooth and do not follow the exact positions of individual data points as closely. The models are more generalized, focusing on the overall trend in the data rather than trying to accommodate every individual point, exhibiting much less overfitting compared to what we saw in the traditional approach using the same model architecture. By averaging the weights from different subsets, the model smooths out the noise in the data and prevents overfitting to specific training samples. This clear improvement in generalization ability stems from the fact that each of the noisy samples in the dataset occurs only once in one of the subsets. So, irregularity around the individual noisy sample does not affect all the subsets, but rather only one of them. Hence, after averaging the weights, the impact of the individual outlier is minimized. We can see further proof of better generalization by comparing the training and test accuracies for each model. These accuracies are summarized in Table 7.1 provided below.

Model Settings	*Training Accuracy*	*Test Accuracy*
No Subsets	**0.9944**	0.6833
N=2 (2 Subsets)	0.7611	0.75
N=3 (3 Subsets)	0.7944	**0.8333**
N=4 (4 Subsets)	0.7889	**0.8333**

Table 7.1: Performance of the Traditional approach vs the D2C method using different numbers of Subsets.

Clearly, **models using the D2C approach are likely to perform better on unseen data, as they avoid overfitting by keeping the decision boundary simple and focusing on capturing the broader, more consistent patterns**

in the data, sacrificing some accuracy on the training set for better generalization. As we can see, we got the highest accuracy on the training set with the traditional approach, which was expected after what we saw in the decision boundary plot. The traditional approach shows extremely high training accuracy, indicating that the model fits the training data almost perfectly. However, the significant drop in test accuracy (0.6833) suggests that the model is overfitting to the training data. It performs poorly on unseen data, meaning it has memorized the training examples rather than learning a generalized pattern. When we use the Divide2Conquer method with 2, 3, and 4 subsets, we observe a clear improvement in this regard. When the data is split into 2 subsets, the training accuracy decreases significantly compared to the traditional approach. However, the test accuracy is now almost on par with the training accuracy (both at around 0.75). This indicates that the model is not overfitting and has better generalization. While its performance on the training set is not as high, the consistency between training and test accuracy suggests improved robustness. With 3 or 4 subsets, both training and test accuracy improve compared to the N=2 approach. The test accuracy (0.8333) surpasses the training accuracy (around 0.79), which is a positive sign of generalization. This indicates that splitting the data into 3 or 4 subsets provides a good balance between reducing overfitting and improving the model's ability to generalize to unseen data.

Splitting the data into subsets leads to a significant reduction in overfitting, as seen by the smoother decision boundary and improved test accuracy. The balanced training and test accuracy indicate that the model learns a more generalizable pattern. So, it can be said that the proposed D2C approach appears to mitigate overfitting effectively and leads to more robust models compared to the traditional approach. However, perhaps the most important characteristic of our proposed method is that it can be used on top of other counter-overfitting measures. For example, we can use dropouts in the edge model architecture and also divide the training set into multiple subsets. In the following sections, we will investigate whether using this method on top of other techniques offers any added advantage or not.

7.2 Analyzing Loss & Accuracy Curves

We will analyze the results obtained from each of our experiments from the 3 different domains. For the loss curves, we used one representative curve from our multiple repetitive runs (to test for repeatability) for each case. The loss curve is a definitive indicator of overfitting. In machine learning, overfitting is marked by a model's increasing validation loss after a certain number of training epochs, indicating that the model has started to capture noise and specific variations in the training set that do not generalize to new data. A model that reaches this overfitting point later, meaning it maintains a lower validation loss over a longer period, suggests it is better regularized or has a complexity that is well-suited to the underlying patterns in the data. A model with a delayed overfitting point continues to find generalizable patterns within the data through many epochs, highlighting that it remains in a generalization phase for longer, rather than immediately entering an overfitting phase. We will analyze whether dividing the training data into more and more subsets delays and slows down the abrupt increase in validation loss as

the training progresses. If the convergence becomes slower, and the abrupt increase of validation loss is minimized as we divide the training set into more and more subsets, it will indicate reduced overfitting.

Also, when a model maintains a lower validation loss during training, it indicates that the model can generalize well to unseen data. Theoretically, overfitting occurs when a model becomes overly complex relative to the data it is learning, allowing it to capture not only meaningful patterns but also specific noise or minor irregularities unique to the training set. This focus on noise causes the model to perform poorly on new data, which is reflected in a higher validation loss or an accelerated rise in validation loss. In contrast, a model that keeps validation loss lower while training likely has a structure that is well-suited to the data's true underlying patterns without becoming excessively sensitive to its unique characteristics. By maintaining a lower validation loss, the model demonstrates an ability to generalize—learning patterns that extend beyond the training set and apply well to validation data. This suggests that the model has achieved a balance, where it is complex enough to capture essential relationships within the data but not so complex that it becomes reliant on details that do not generalize. Consequently, a model that sustains a lower validation loss is theoretically overfitting less, as it shows a stronger alignment with the data's general structure rather than its idiosyncrasies. So, if we can conclude that our proposed method manages to bring down the validation loss during training compared to the traditional method, it will be a definitive sign of better generalization and reduced overfitting. We will also observe whether this reduction in validation loss becomes more apparent as we increase N, the number of subsets. This is because dividing the training set into more and more subsets should help reduce overfitting, as we saw in Eq. (4.15). The more the number of subsets (k in that equation), the less the variance, s/k, for the best-case scenario. Hence, with the decrease in variance should come the decrease in overfitting. Also, as we saw before, the true error, represented by validation loss in this case, is minimized using D2C. We will verify that hypothesis using validation loss and accuracy curves and analyze whether D2C is indeed effective in treating the problem of overfitting. Let us examine the loss curves generated in the three domains we have experimented on.

Image Classification: Let us first take a look at the loss curves generated using the **CIFAR-10** dataset. For now, we will keep the number of epochs before each round of central averaging, $\boldsymbol{E}$, equal to 1. It will ensure a proper comparison between the traditional method and our proposed method while varying the number of subsets, $\boldsymbol{N}$. In our experiments, how much we varied N depended on the performance (Loss curve, Accuracy) trend as we increased N, the number of subsets. In the case of all the datasets, we stopped varying N whenever we observed a definitive performance drop with increasing N. In the case of **CIFAR-10**, we varied N in the range of 1 to 9.

As we can see from Figure 7.6, the validation loss keeps decreasing with each epoch for longer when using our proposed D2C method on the **CIFAR-10** dataset but starts to increase after very few epochs again in the case of traditional NN implementations using the entire training set. This happens due to the overfitting that takes place without the division of subsets and weight averaging. We can also see

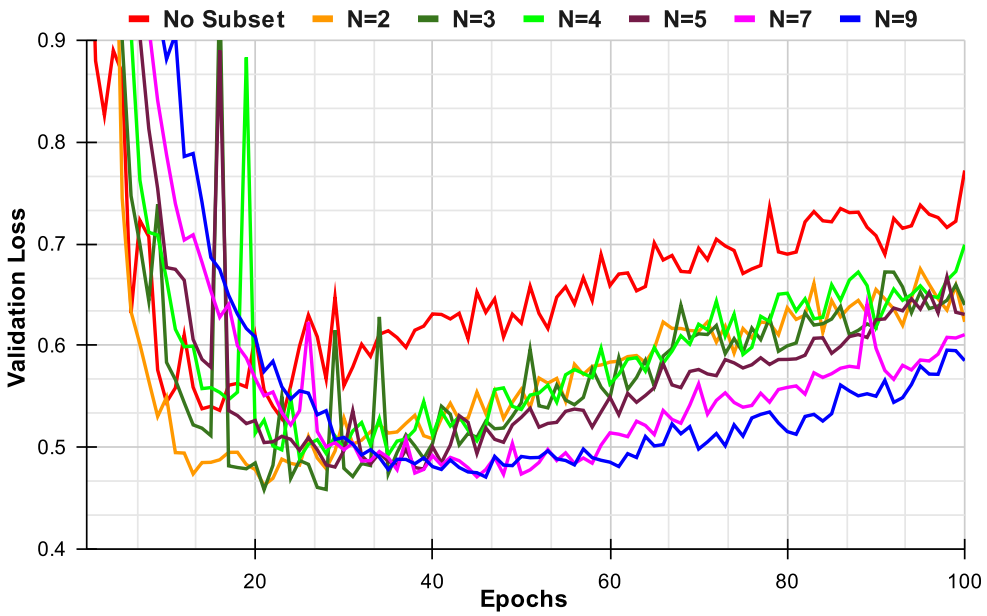

Figure 7.6: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on CIFAR-10.

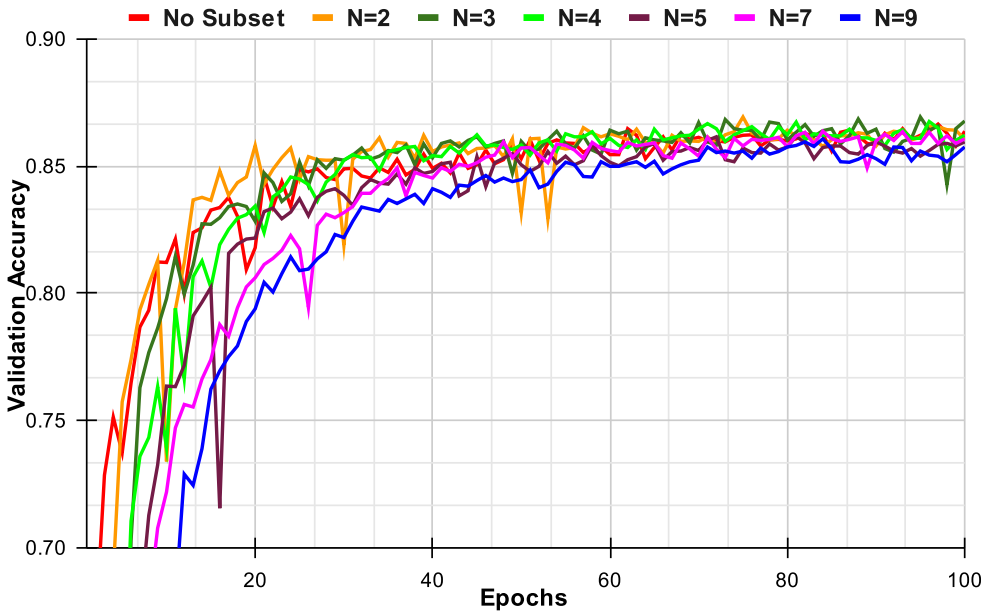

Figure 7.7: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on CIFAR-10.

that dividing the training data into more and more subsets delays and minimizes the abrupt increase in validation loss as the training progresses. For example, the loss curve generated using the traditional approach without dividing the training data (no subsets) reaches its minimum, 0.536, at around 16 epochs and then it keeps increasing and reaches 0.77 by the end of 100 epochs. As we set the number of epochs before each round of central averaging,$E = 1$, each central epoch is equivalent to one epoch in the traditional approach. For the number of subsets,$N = 3$, the loss curve reaches its minimum, 0.462, at around 21 epochs and then it keeps increasing and

reaches0.64 by the end of 100 epochs. When we further divided the training data into more subsets, setting $N = 9$, the loss curve reaches its minimum,0.47, at around 46 epochs and then it keeps increasing and reaches 0.585 by the end of 100 epochs. So, clearly, the convergence becomes slower, and the abrupt increase of validation loss is minimized as we divide the training set into more and more subsets, which indicates reduced overfitting. It should be noted that there is a trade-off between bias and variance here, and even though dividing the training set into more subsets seems to work better against overfitting on the training data, the optimum number of subsets lies somewhere in the middle. The minimum value of loss that each of the cases discussed above also tells a similar story. It should also be noted that our proposed method manages to bring down the minimum value of validation loss during training compared to the traditional method, which is also a promising sign. We will discuss that further in the sections that will follow.
However, the loss curve isn't the only important factor during the training process. The relation between log loss and accuracy may not be so linear. That is why it is important to observe the validation accuracy too. As we can see from Figure 7.7, for all the different values of the number of subsets, N, even though the loss reaches its minimum much earlier, the accuracy shows an increasing trend (even if slightly) almost throughout the entire training process, right up until the 100th epoch. This shows why continuing training for a significant period might be necessary even if the validation loss starts increasing. This also shows why early stopping based on loss curves may not be a good idea in both cases: the traditional approach and the D2C method. In the case of **CIFAR-10**, at the very least, dividing the training set into multiple subsets ensures that the maximum accuracy is attained while keeping the log loss relatively in control, which characterizes a more confident and robust model. We will discuss this further in later sections.

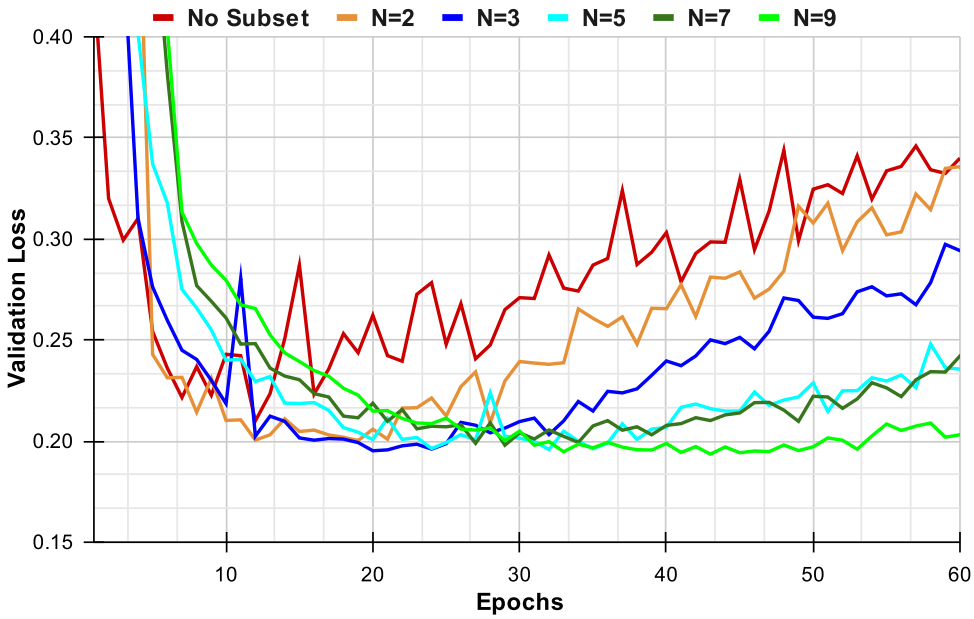

Figure 7.8: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on Fashion MNIST.

Now let us observe the loss curves generated using the Fashion MNIST dataset. Once again, the number of epochs before each round of central averaging, E was kept

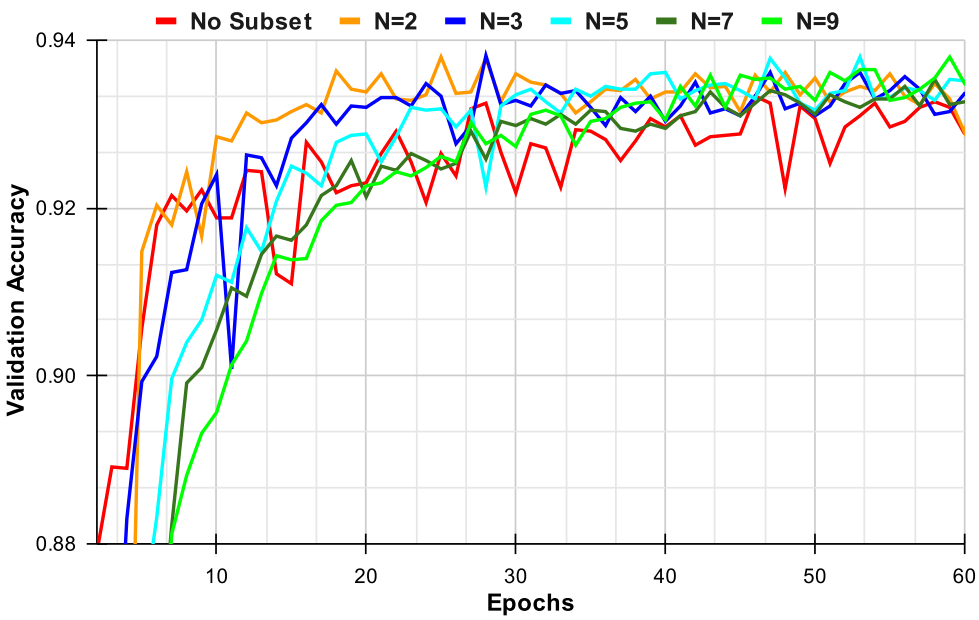

Figure 7.9: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on Fashion MNIST.

constant at 1 and the number of subsets, N was varied between 1 and 9.Figure 7.8 shows that once again the validation loss keeps decreasing with each epoch for longer when using the Divide2Conquer method on the Fashion MNIST dataset but starts to increase abruptly after very few epochs in the case of traditional implementations using the entire training set. In Figure 7.8, we can see that D2C with 9 subsets almost entirely eliminates the trend of an increasing loss after a few epochs. In this case, the pattern is a bit clearer. The more subsets we divide the training set into, the more overfitting-resistant the model becomes, which is evident from the respective loss curves. The validation loss starts increasing quite rapidly with the traditional approach or when the number of subsets is low ($N = 2$). For a bit higher number of subsets ($N = 5, 7$), the loss eventually starts increasing too but the rate is much slower. For an even higher number of subsets ($N = 9$), the phenomenon of the loss curve starting to increase is almost completely mitigated throughout the entire training process of 60 epochs. Once again, the convergence becomes slower and slower with an increasing number of subsets too. The validation accuracy curves from Figure 7.9 show what we saw in the case of **CIFAR-10** once again, the accuracy becomes somewhat stable toward the end of the training phase. But in this case, if we compare the validation accuracy curve for the traditional approach to the ones we got for different numbers of subsets for D2C method, we can see a clear improvement in terms of validation accuracy. Implementations based on our proposed method outperform the traditional implementation with no subsets, which is evident from the validation accuracy curve staying over the one representing no subset for almost the entirety of the training process. While the validation accuracy for the traditional implementation mostly takes up a stable value below 0.93, the implementations using the D2C method take up a stable value of over 0.93. So, in the case of Fashion MNIST, dividing the training set into multiple subsets improves both accuracy and loss compared to the traditional approach, which is expected of a model that addresses overfitting effectively. This will become clearer when we

discuss test accuracy in later sections.

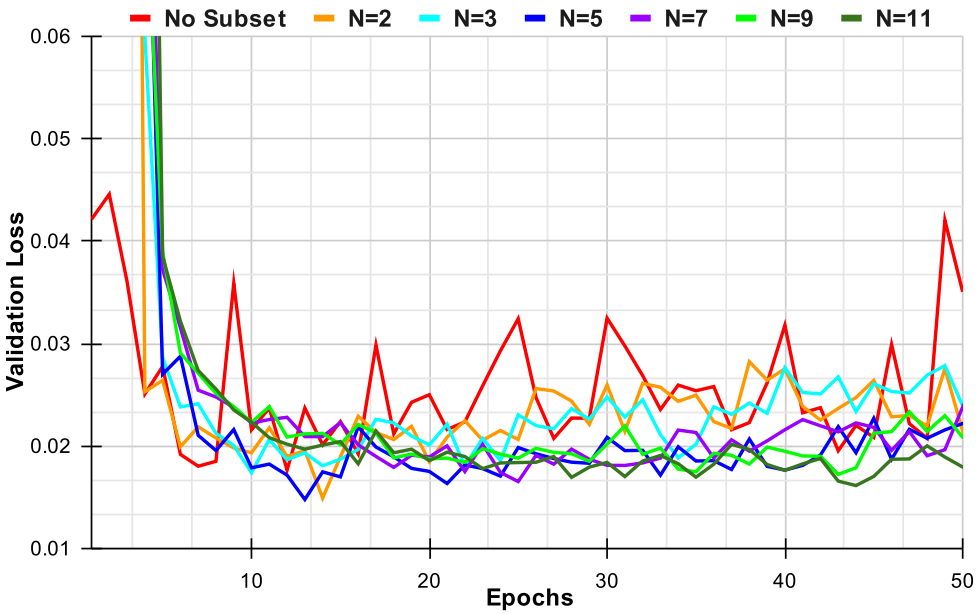

Figure 7.10: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on MNIST.

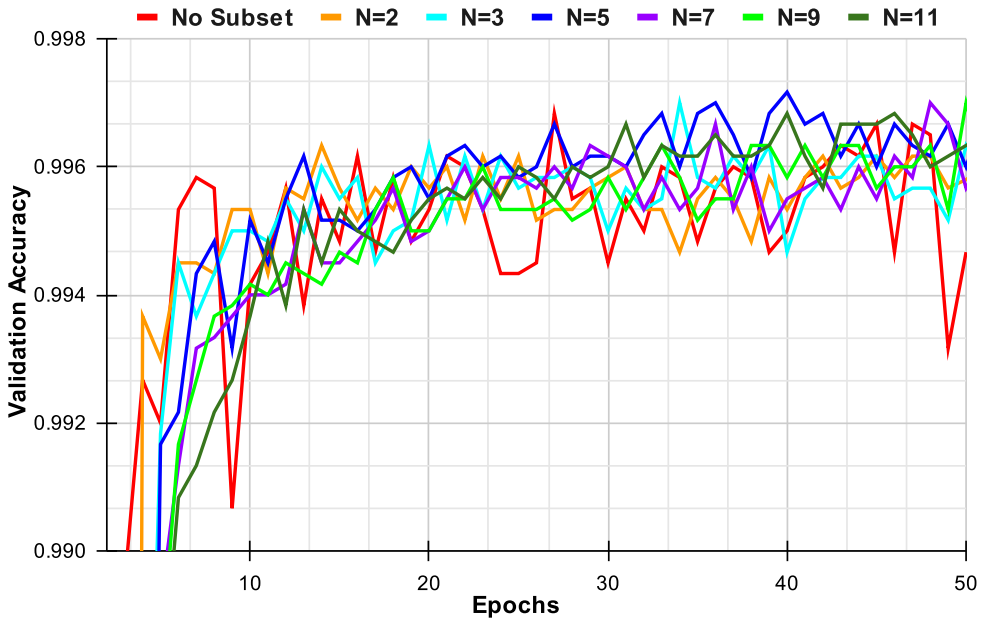

Figure 7.11: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on MNIST.

Next, let us analyze the loss curves generated using the MNIST dataset. The number of epochs before each round of central averaging, E was kept constant at 1 and the number of subsets, N was varied between 1 and 11 in this case. MNIST is comparatively a less challenging dataset, especially in terms of overfitting. That's why we can see in Figure 7.10 that the validation loss doesn't shoot up like the

previous cases for MNIST. For all the values of N, the validation loss becomes somewhat stable after 10-20 epochs. But even in this case, dividing the training set into multiple subsets results mostly in a lower validation loss throughout the training phase compared to the traditional approach (no subset). We can see a similar pattern in the validation accuracy curve from Figure 7.11. The validation accuracy generated using the traditional approach is lower in general throughout the training phase compared to the implementations using our proposed method, for different values of N. So, in the case of MNIST too, dividing the training set into multiple subsets improves both accuracy and loss, which implies less overfitting.
There is another interesting aspect of the validation loss curve here. As we can see from Figure 7.10, the validation loss curve generated using the traditional approach deviates much more than the ones generated using the D2C method. A curve that does not fluctuate much after converging suggests that the model has reached a stable state. This stability is often a sign of robustness, meaning the model is not overly sensitive to slight variations in the data or training process. Dividing the dataset into more and more subsets results in smaller and smaller deviations in the validation loss curve. This was evident in previous loss curves too, to a lesser extent. This proves that dividing the training set into multiple subsets not only improves accuracy and loss but also brings stability and consistency to the loss/accuracy curves.

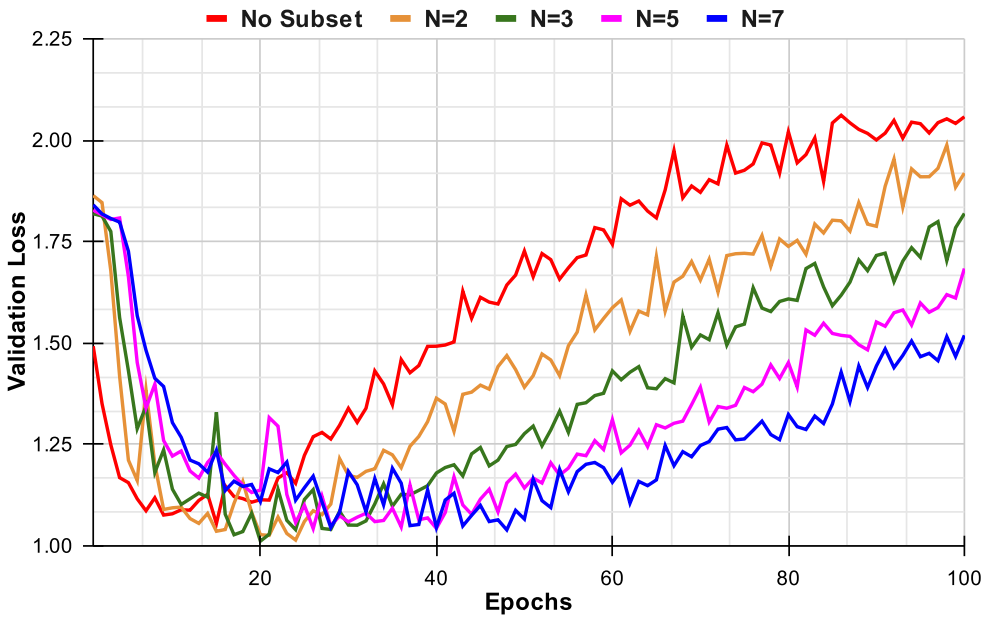

Figure 7.12: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on FER2013.

Apart from these three benchmark datasets, we also used **FER2013** to further test our proposed method. The reason for choosing FER2013 was that facial expression recognition is a particularly overfitting-prone task. So we expected crucial insights from experiments performed on this dataset. Let us take a look at the loss curves generated using the FER2013 dataset. First, the number of epochs before each round of central averaging, E was kept constant at 1 and the number of subsets, N was varied between 1 and 7. Later we also analyzed the impact of changing the

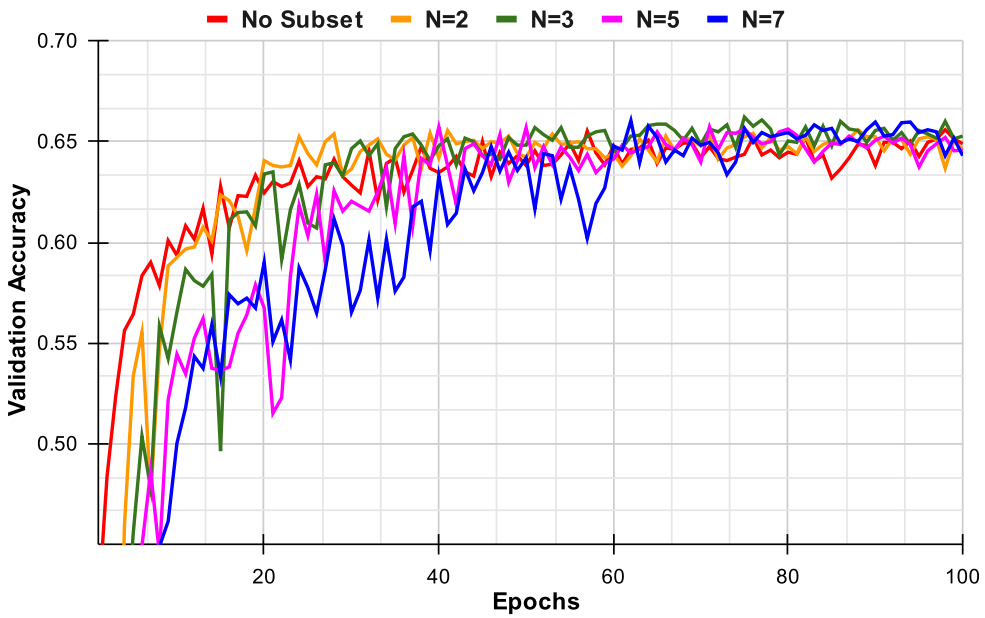

Figure 7.13: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on FER2013.

number of epochs, E, by varying it keeping N constant, and observing the loss and accuracy curves. To compare the curves effectively, in each case, we trained our model in such a way that the total number of global iterations equals the number of global iterations for $E = 1$ divided by E. That way, the total number of local epochs during training was kept constant, as each of the global iterations represented E number of local epochs in the subsets. The loss and accuracy curves from Figure 7.12 and Figure 7.13 show further proof of what we concluded from the previous cases. As expected, the validation loss starts increasing quite abruptly after a few epochs in the case of the implementation based on the traditional approach. The overfitting in this case is much more apparent. The rate of increase in validation loss goes down very rapidly as we increase the number of subsets. This trend is more evident here compared to the other cases. The validation accuracy, however, doesn't come down as the loss increases. Rather counterintuitively, the validation accuracy maintains an increasing trend almost throughout the 100 epochs. We also notice a slight improvement in the peak stable accuracy with the D2C method over the traditional approach. So, once again, dividing the training set into multiple subsets improves both accuracy and loss. All these observations show that our proposed D2C method shows obvious effectiveness in addressing the problem of overfitting, at least in these image classification tasks. Let us also take a look at the validation loss and accuracy curves generated while keeping the number of subsets, N, constant and varying the number of local epochs before each round of central averaging, E. We varied E between 1 to 3 and Kept N constant at 2 and 3 separately.
We can see from Figure 7.14 that when we vary the number of epochs before each round of central averaging, E, there is no clear indication of any kind of change in the trend or the rate of change in the validation losses as the training progresses. This is unlike when we varied the number of subsets, N keeping E constant, where we saw that increasing the number of subsets delayed and minimized the increase in validation loss as the training progressed quite clearly. Also, the convergence got

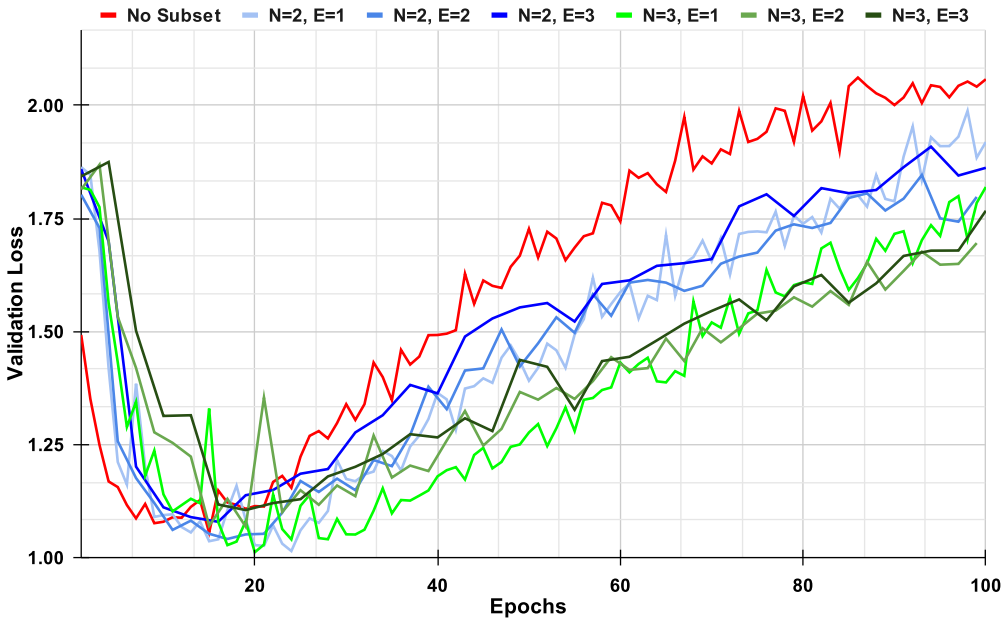

Figure 7.14: Validation loss curves for different numbers of epochs before each round of Central Averaging, E, while keeping the number of subsets, N = 2 & 3.

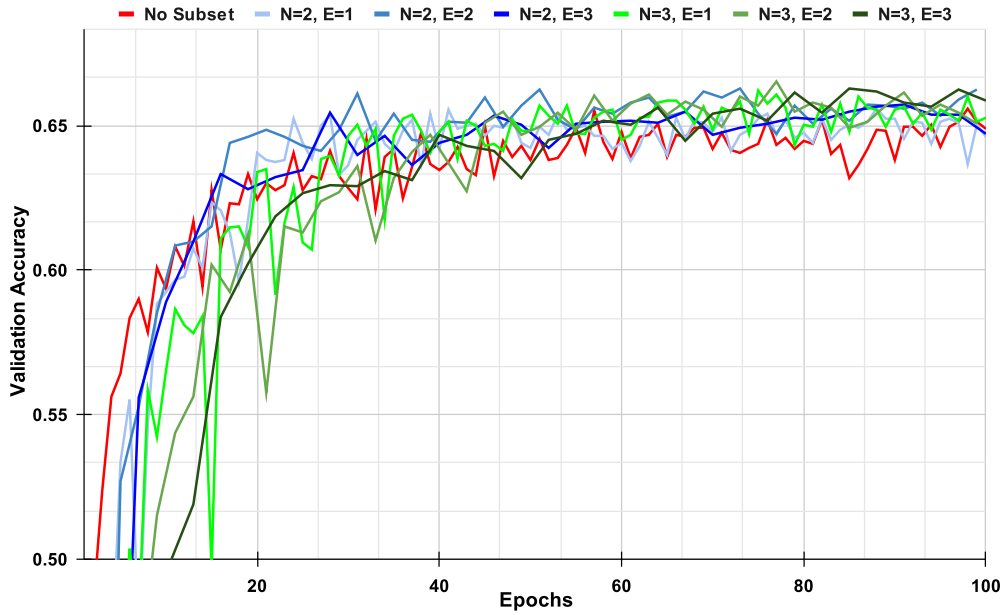

Figure 7.15: Validation accuracy curves for different numbers of epochs before each round of Central Averaging, E, while keeping the number of subsets, N = 2 & 3.

slower. But here, for both $N = 2$ (shades of blue) and $N = 3$ (shades of green), the validation curves are quite similar for different values of E throughout the training phase. From Figure 7.15 it is evident that there is no clear sign of a change in the accuracy either while varying E. In light of the above observations, we can conclude that varying the number of epochs before each round of central averaging, E does not have any obvious implications on log loss as we saw in the case of varying the number of subsets, N. However, an increased number of local epochs before each round of central averaging means more scope for learning from each of the subsets

before averaging, sometimes even overfitting on a particular subset a bit more. We will observe the effect of varying the number of epochs further in the later sections when we observe the final accuracy and log loss in each case. The implication of varying E is likely to be different on different datasets from different domains as we will see in the case of the number of subsets too.

Audio Classification: Overfitting is a common problem in audio classification tasks. But it is more apparent when the model is trained on a particular dataset, and then tested on data from a different source. One way of tackling this issue is by combining datasets to increase the diversity in both training and test/validation data. This is the strategy we followed to test our method in the Speech domain. We combined the datasets TESS [31], CREMA-D [32], and RAVDESS [33] for this task. All these are audio-emotion recognition datasets and can be combined to have 14 classes covering 7 emotions and 2 genders (male and female). Unlike CIFAR-10, MNIST, and Fashion MNIST, this combined dataset has an imbalanced distribution. Let us take a look at the validation loss and accuracy curves generated using this Combined Audio dataset.

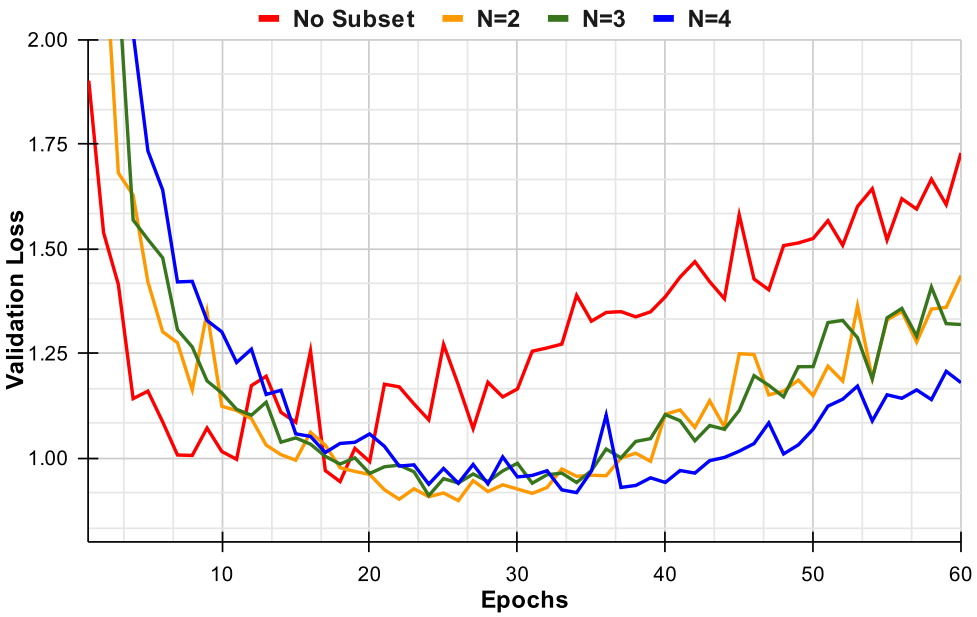

Figure 7.16: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on the Combined Audio dataset.

As we can see from Figure 7.16, there is overfitting to some extent in all cases, which causes the validation loss to creep up after a few epochs but the rate at which this phenomenon takes place is much less when the training set is divided into multiple subsets. Once again, the rate of increase in validation loss goes down as we increase the number of subsets. Even though we converted the 1D time series audio files to 2D Log-Mel spectrograms to form something like an image, the classification based on these Mel-specs differs greatly from traditional image classification. So, the overfitting-resistant nature of models using our proposed method being apparent in this case too is a very promising sign. However, from Figure 7.17, we can see that the improvement is not so apparent when it comes to the validation accuracy curve.

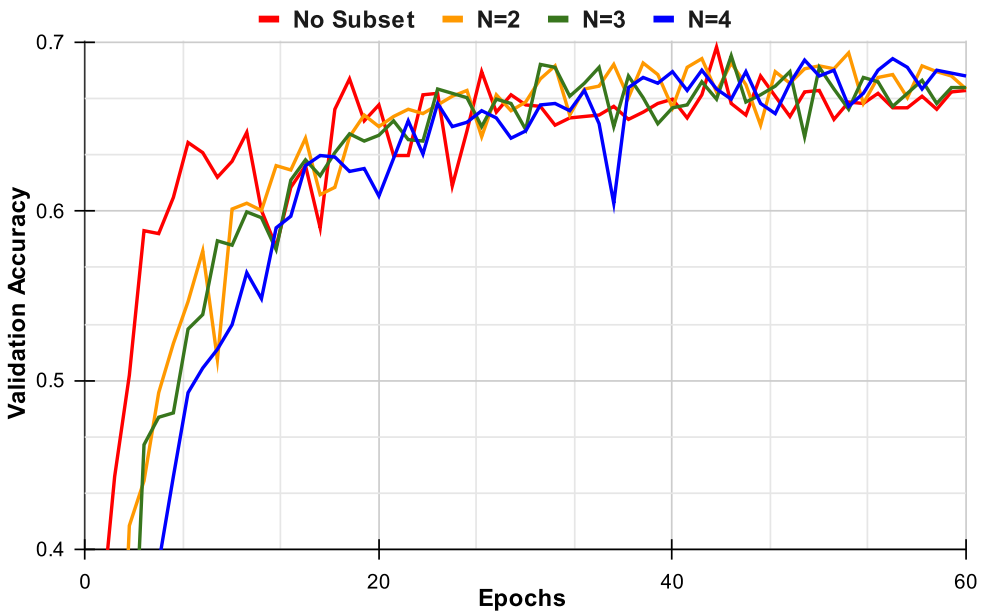

Figure 7.17: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on the Combined Audio dataset.

Rather accuracy reaches its highest with the traditional approach at one point. The reason for this can be logically deduced from the context and the size of the dataset. The combined dataset has only 11632 samples in total, which was further divided into training and validation sets with a 90:10 split. Thus, the resulting training data is much less compared to the datasets we used in the image classification tasks. We discussed earlier that dividing a small dataset into even smaller subsets may cause issues in training, and the model may underperform a bit if each of the subsets can't be fed with enough data. In this case, the number of samples per class becomes too low when we divide this already smaller dataset into more subsets. This might be the reason for the accuracy not showing signs of improvement despite the loss curve showing signs of addressing the issue of overfitting. But there is another thing that we should keep in mind. This Combined Audio dataset is not balanced. So, accuracy may not be an appropriate metric to evaluate in this case anyway. We can use the F1 score to evaluate the performance when the dataset has an imbalanced distribution of classes. This is what we will do in a later section to further evaluate the effectiveness of our model on this dataset. However, in the case of audio classification too, we can say that at the very least, the D2C method ensures that comparable accuracy is attained while keeping the log loss to a lower value, which characterizes a more confident and robust model that classifies more distinctively.
Another interesting case study would be the UrbanSound8K dataset. This benchmark dataset is not too large either with 8732 samples. It comes in 10 folds. 10-fold cross-validation is the recommended method in this case for proper comparisons. However, we will use the 10th fold for testing and combine the 9 other folds for training. In the source of this dataset [34], it is explicitly mentioned that "If you reshuffle the data (e.g. combine the data from all folds and generate a random train/test split) you will be incorrectly placing related samples in both the train and test sets, leading to inflated scores that don't represent your model's performance on unseen data. Put simply, your results will be wrong." It implies that the

samples within the folds might be related to each other, probably through some kind of augmentation. This is important because it also means that the same noisy samples and outliers can be present multiple times in a fold. So, if we divide the training set randomly, there is a high chance that these noisy samples/outliers will be present in all/multiple subsets. So, it would not have the regularizing effect we talked about in the previous chapters as it won't be able to punish these samples properly.

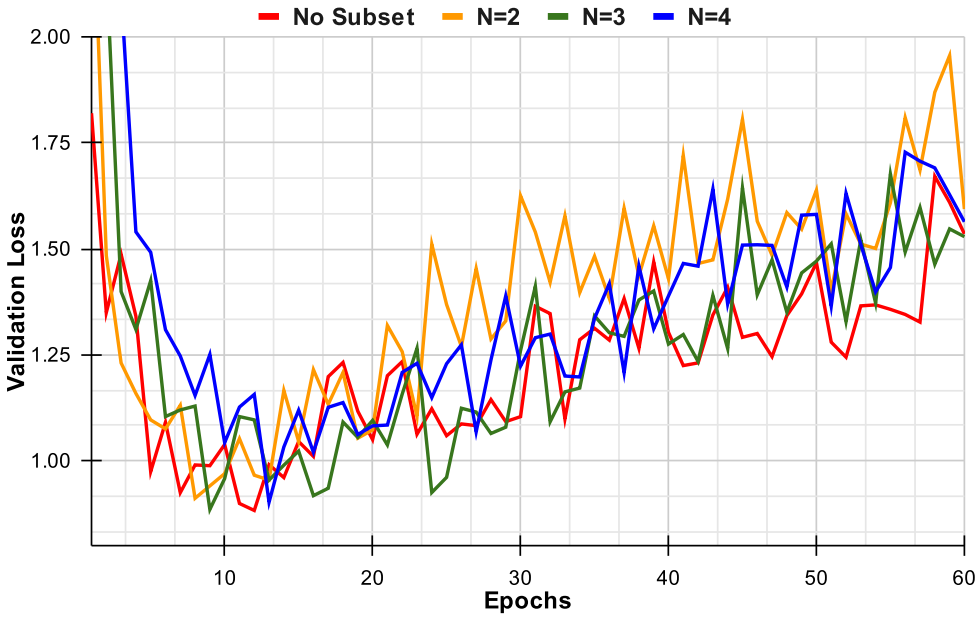

Figure 7.18: Validation loss curves for the traditional method & the D2C method for different numbers of subsets, N, on UrbanSound8k.

To test for it, let us examine the loss curves generated using this dataset. As we can see from Figure 7.18, unlike the previous cases, dividing the training set into multiple subsets did not bring any significant change in the loss curve. Rather, losses with the traditional approach seem to be slightly smaller than the losses we got using the D2C method. Moreover, from Figure 7.19, we can see that dividing the training set into multiple subsets worsens the validation accuracy compared to the traditional approach in this case. We didn't notice this in any of our previous experiments. All these phenomena can be attributed to the fact that the noisy samples/outliers were present in all/multiple subsets. So, it didn't have the regularizing effect we talked about, it didn't minimize the impact of outliers. So, it couldn't address the issue of overfitting like the other cases. This observation brings us to a very important aspect of the D2C method. While dividing the training set into multiple subsets, it must be maintained that the subsets are completely distinct and unique to each other. It should be made sure that there is no data leakage between the subsets. Any sort of data augmentation should be performed on the subsets; after dividing the training set, otherwise related samples will be present in multiple subsets and as a result, overfitting won't be addressed.

Text Classification: Text classification is an important case study when it comes to testing the Divide2Conquer method. For both image classification and audio clas-

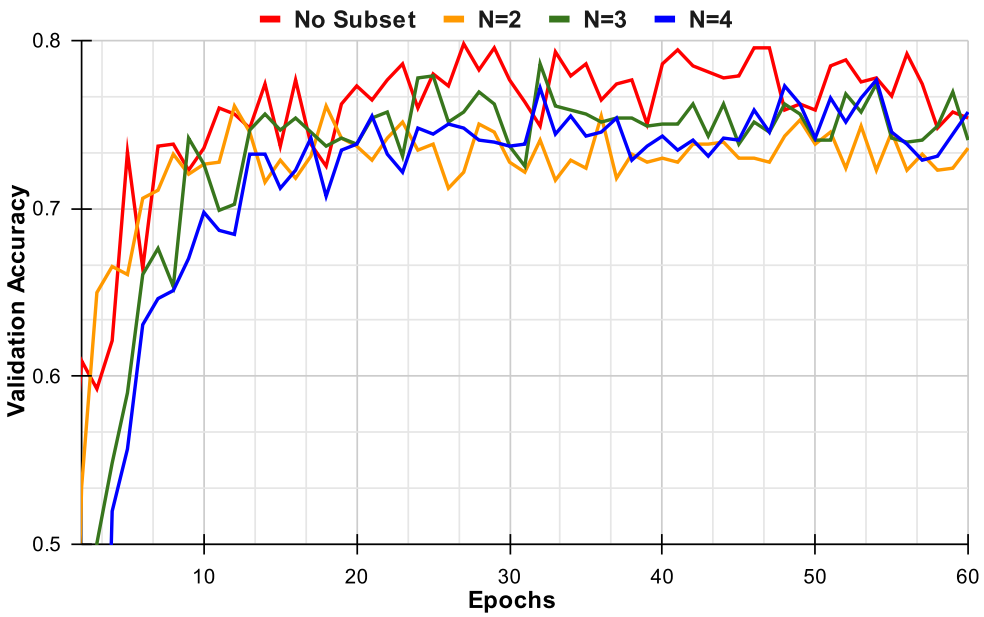

Figure 7.19: Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets, N, on UrbanSound8k.

sification, we used 2D data and CNN to carry out the tasks. But text classification deals with 1D sequences, and as we mentioned in the previous chapter, we will use bi-directional LSTM in this case. These experiments will be crucial to determine the applicability of our proposed method across different domains.

We used AG News [35], a benchmark dataset to test our method primarily in the NLP domain. This is a large dataset containing 120000 samples in the training set, and a separate test set containing 7600 samples. Its large size and balanced distribution mean this is a perfect candidate for applying the Divide2Conquer method. Let us observe the validation loss and accuracy curves generated using the AG News dataset first.

The loss and accuracy curves generated for AG News depict by far the clearest picture in terms of the improvements brought by the proposed method over the traditional approach. The loss curve from Figure 7.20 shows the trend we observed in almost all other cases, which is that dividing the training data into more and more subsets delays and minimizes the increase in validation loss as the training progresses. The validation loss keeps decreasing with each epoch for longer when using the D2C method on the AG News dataset but starts to increase after very few epochs again in the case of traditional NN implementations using the entire training set. The convergence though, slows down a bit in this case too. More importantly, with no subsets, the validation loss shoots up sharply after around 8 epochs, which indicates severe overfitting and impacts the accuracy too as the training progresses as we can see from Figure 7.21. However, when we divide the training set into multiple subsets, the sharp increase in the validation loss is mitigated drastically. Even with only two subsets, the improvement is quite significant. When the number of subsets is increased further to more than two, the loss curve becomes quite stable. From the accuracy curve from Figure 7.21, we can observe by far the clearest indications

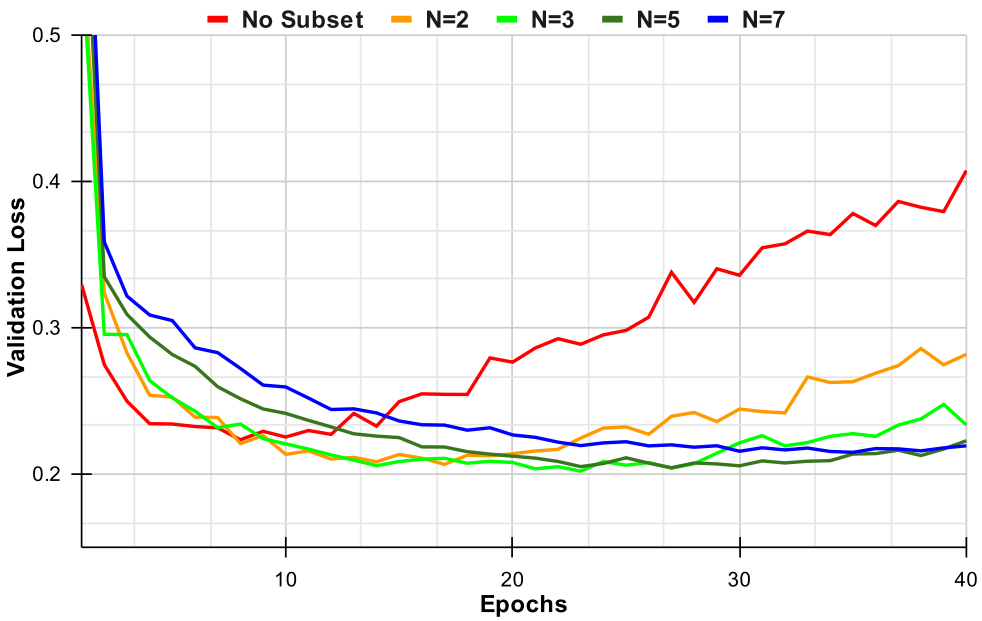

Figure 7.20: Validation loss curves for the traditional & the D2C method for different numbers of subsets, N, on AG News.

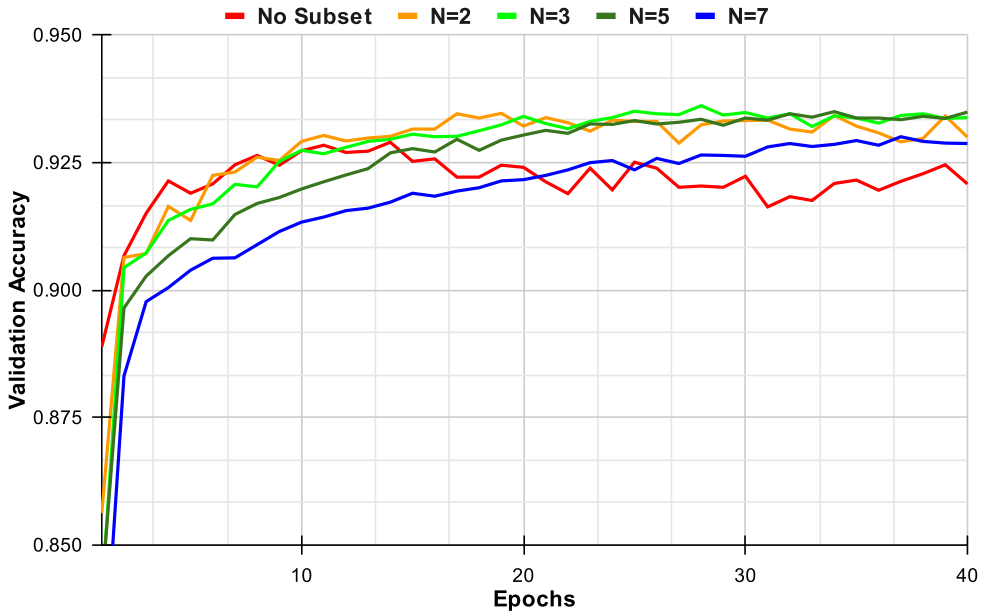

Figure 7.21: Validation accuracy curves for the traditional & the D2C method for different numbers of subsets, N, on AG News.

observed till now of an improvement over the traditional approach. The accuracy crosses the 93% mark and remains stable when the number of subsets is 2, 3 or 5. However, the validation accuracy for the traditional approach increases till the 14th epoch and then it starts to decrease again. This deterioration in accuracy can be linked to the fact that the loss kept increasing drastically and at some point, it started hurting the accuracy too. This is a classic example of overfitting. The experiments on AG News prove that using the D2C method can indeed address overfitting and enhance the performance of deep learning models. So, in the case

of text classification on AG News too, the Divide2Conquer method improves both accuracy and loss compared to the traditional approach, which can be accredited to the overfitting resistant nature of models using our method. This also shows that dividing the training set into multiple subsets works in different domains and a diverse set of tasks. We will analyze it further when we discuss test accuracy and log loss in later sections.

7.3 Evaluation Based on Accuracy, F1 Score, AUC-ROC, MCC and Log Loss

We will evaluate our method based on the results obtained from each of our experiments from the 3 different domains. The test accuracy is the primary evaluation metric in this case. However, since we have some datasets that are not balanced in terms of class-wise distribution, we will observe the F1 Score too in such cases. We also reported the log loss values, even though it was not considered the primary evaluation metric. A lower log loss indicates that the model's predicted probabilities align better with the true class labels. Thus, especially for models with comparable accuracy or F1 score, log loss becomes an important indicator to determine the better model with superior generalization performance. Also, While accuracy only considers whether a prediction is correct or incorrect, AUC-ROC evaluates how well the model can differentiate between positive and negative instances based on their predicted probabilities. That's why we used AUC-ROC too to evaluate our method. For accuracy, log loss, AUC-ROC, and F1 score values, we used the average from our multiple repetitive runs (to test for repeatability) for each case. Each of the experiments was repeated at least 3 to 5 times. Let us take a look at all those results. As we mentioned earlier, all these experiments were conducted using the same parameters and the same base Neural Network architecture which we explained in the previous chapter. We used the provided test sets with each of the datasets for proper comparisons. Almost all of these datasets had predefined test sets. We used shades of red for the traditional approach (No subsets) in the bar charts. The best model was represented using darker shades than the rest of the models having different settings. In the tables, the top row represents the traditional approach and the rest of the rows represent different variations of the D2C approach. The bold entities represent the best performance in the tables.

Image Classification: Let's take a look at the performance metrics from the test sets of all the image classification datasets we primarily used. In this section too, we will vary the two new global hyperparameters: the Number of Subsets of the Training Set and the Number of Epochs before each round of Central Averaging. We will refer to them using N and E respectively. Let us take a look at the performance metrics from the CIFAR-10 dataset using different settings (traditional approach and the D2C method with different values of N and E).
As we can see from Table 7.2, the best accuracy, MCC and AUC-ROC were achieved using the D2C method. The number of subsets, N, was 3 in this case for the best model. The number of epochs before each round of central averaging, E, was just 1. This approach showed an improvement with an accuracy of 86.13% over the traditional approach with no subsets, which yielded an accuracy of 85.96%. The

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subsets	0.8596	0.8591	0.7346	0.98697	0.8440
N=2, E=1	0.8611	**0.8611**	0.7042	0.98693	0.8459
N=3, E=1	**0.8613**	0.8609	0.6682	**0.98704**	**0.8460**
N=4, E=1	0.8586	0.8583	0.6516	0.98690	0.8433
N=5, E=1	0.8552	0.8551	0.6389	0.98628	0.8390
N=7, E=1	0.8539	0.8535	0.6198	0.98640	0.8380
N=9, E=1	0.8531	0.8525	**0.6068**	0.98605	0.8370

Table 7.2: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on CIFAR-10.

accuracy improved as we increased the number of subsets initially, but suffered a consistent drop after it was further increased past 3. As we discussed before, dividing the training set into too many subsets may have negative impacts on training due to a lack of representative samples. The experimental outcomes show further evidence of that in this case.

We can also see that as we increase the number of subsets, N, there is a clear trend of decreasing log loss. As we discussed earlier, increasing the number of subsets further minimizes the effect of noise and outliers. The decrease in log loss is likely to be a result of that phenomenon. Even though the best model isn't usually selected based on log loss values, in the case of almost similar accuracy (the accuracies for $N = 2$ and $N = 3$ are very similar in this case), the best model can be selected based on log loss. A lower log loss means a more robust and confident model, so the model with a lower log loss is preferred when the accuracy or F1 score is similar. Based on that observation, the best model here is the one with 3 training subsets and 1 epoch before each round of central averaging.

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subsets	0.9314	0.9315	0.3467	0.99559	0.9239
N=2, E=1	0.9323	0.9319	0.3460	0.99549	0.9248
N=3, E=1	0.9330	0.9329	0.3179	0.99545	0.9255
N=3, E=2	0.9319	0.9316	0.3303	0.99552	0.9244
N=3, E=3	0.9337	**0.9337**	0.3225	0.99590	0.9264
N=5, E=1	0.9322	0.9321	0.2692	0.99589	0.9247
N=7, E=1	**0.9338**	0.9334	0.2467	0.99584	**0.9265**
N=7, E=2	0.9305	0.9304	0.2575	**0.99600**	0.9228
N=9, E=1	0.9323	0.9323	**0.2341**	**0.99600**	0.9248

Table 7.3: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on Fashion MNIST.

For **Fashion MNIST** too, we can observe in Table 7.3 that the best accuracy, MCC, and AUC-ROC were achieved using our proposed Divide2Conquer method.

However, the best model in this case had more subsets than what we saw in the case of **CIFAR-10**. The number of subsets, N, was 7 in this case for the best model. The number of epochs before each round of central averaging, E, was once again just 1. This approach again showed an improvement with an accuracy of 93.38% over the traditional approach with no subsets, which yielded an accuracy of 93.14%. An increasing trend is visible in accuracy as we increase the number of subsets, till it reaches the best conditions with $N = 7$. As for the log loss, as we can see from Figure **??**, there is a clear trend of decreasing log loss again as we increase the number of subsets, which further justifies the choice of the model with 7 training subsets as the best model.

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subsets	0.9952	0.9952	0.0258	0.99995	0.9947
N=2, E=1	0.9957	0.9956	0.0271	0.99995	0.9952
N=3, E=1	0.9958	0.9958	0.0240	0.99998	0.9953
N=3, E=2	0.9951	0.9950	0.0250	0.99996	0.9945
N=5, E=1	0.9957	0.9956	0.0202	0.99998	0.9952
N=7, E=1	0.9951	0.9950	0.0215	0.99997	0.9945
N=9, E=1	0.9957	0.9957	0.0179	0.99998	0.9953
N=11, E=1	**0.9959**	**0.9959**	**0.0161**	**0.99999**	**0.9955**
N=11, E=2	0.9958	0.9958	0.0170	0.99998	0.9953

Table 7.4: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on MNIST.

In the case of **MNIST** too, we can observe in Table 7.4 that the best accuracy, MCC and AUC-ROC were achieved using our proposed method. The best model in this case had even more subsets than what we saw in the case of **Fashion MNIST**. The number of subsets, N, was 11 in this case for the best model. The number of epochs before each round of central averaging, E, was 1. This approach again showed an improvement with an accuracy of 99.59% over the traditional approach with no subsets, which yielded an accuracy of 99.52%. This improvement of 0.07% in error rate is significant for the **MNIST** dataset. As for the log loss, there is a trend of decreasing log loss again as we increase the number of subsets, albeit with some fluctuations this time. The least amount of log loss is achieved with the best model in terms of accuracy, which featured 11 training subsets and 1 epoch before each round of central averaging.

The comparison between **CIFAR-10**, **Fashion MNIST**, and **MNIST** in terms of the number of subsets in the best-performing model can give us some crucial insights. Let us take a look at the bar charts showing the accuracies for different settings of D2C and the traditional approach for better visualization.

As we can see, the best model for CIFAR-10 had only 3 training subsets, while the best models for **Fashion MNIST** and **MNIST** featured 7 and 11 subsets each. There was a clear trend of decreasing accuracy as we increased the number of subsets past 3 in the case of CIFAR-10. No such phenomenon occurred in the cases of the other two datasets. We can understand these results better if we keep in mind the differences between these benchmark datasets. All these datasets contained the

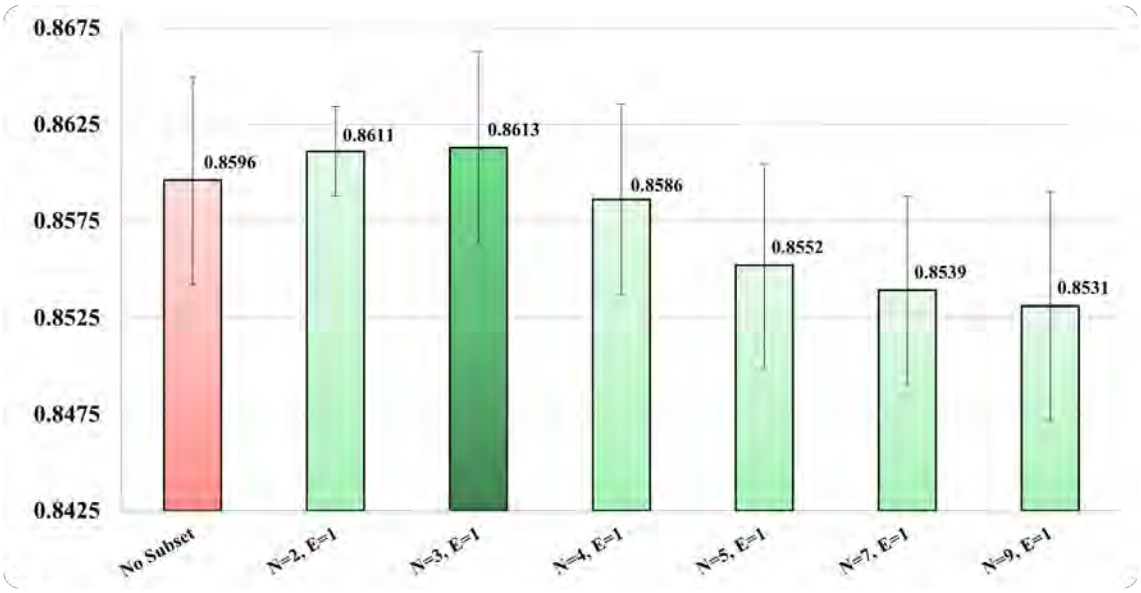

Figure 7.22: Test accuracies for the traditional approach and different values of N and E using the D2C method on the CIFAR-10 dataset.

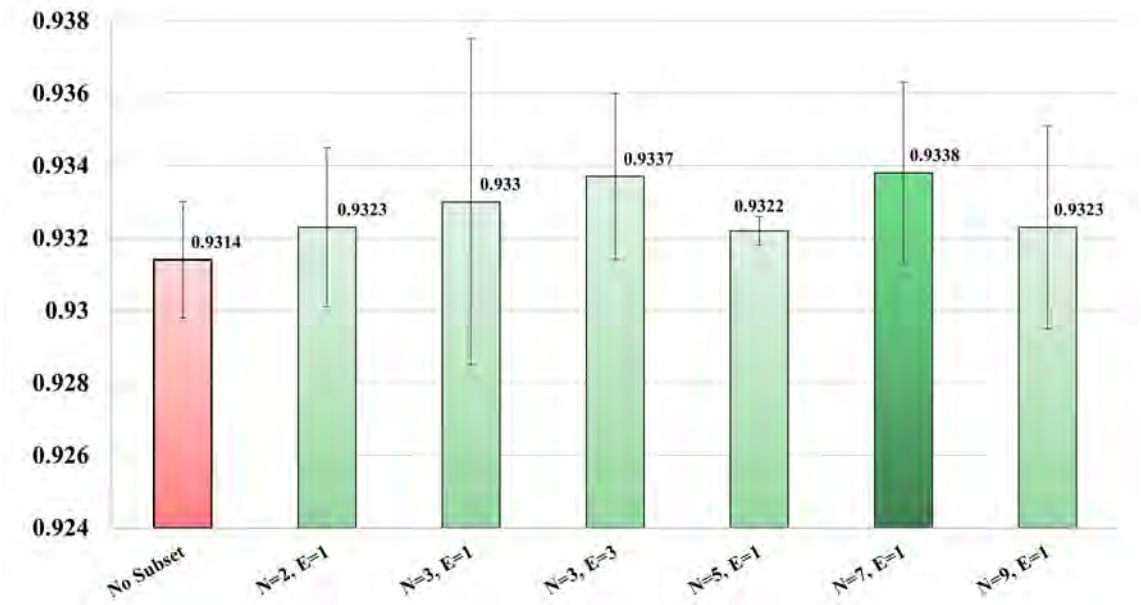

Figure 7.23: Test accuracies for the traditional approach and different values of N and E using the D2C method on the Fashion MNIST dataset.

same number of classes, which is 10. But CIFAR-10 is a smaller dataset compared to **Fashion MNIST** and **MNIST**. It also presents by far the most challenging classification task among these 3 datasets, in the form of RGB images and more complex classes. As a result, for effective training, each edge model for **CIFAR-10** requires more training samples. When we increase the number of subsets, the amount of training data in each subset becomes less. Subsequently, when the subsets become too small, it starts showing negative impacts on the performance of the model. **Fashion MNIST** and **MNIST**, however, are identical in terms of dataset size. Both of them contain greyscale images. But **Fashion MNIST** presents a classification task that is a bit more challenging, compared to the handwritten digit

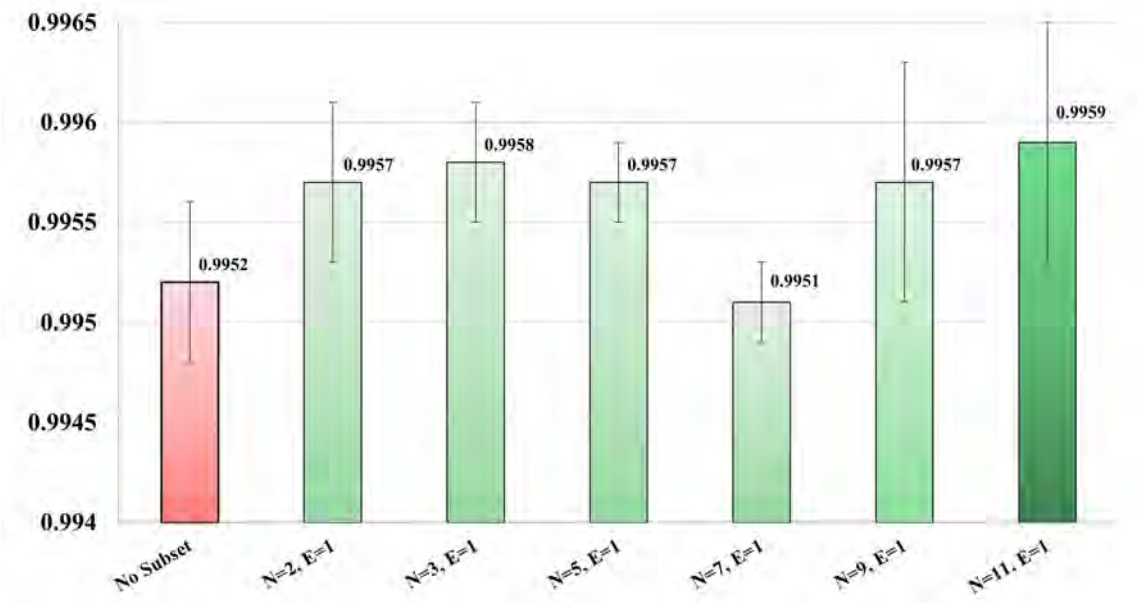

Figure 7.24: Test accuracies for the traditional approach and different values of N and E using the D2C method on the MNIST dataset.

classification in **MNIST**. That's why the number of subsets in the best-performing model for **Fashion MNIST** was less than that of **MNIST**, as the edge models needed more training samples to train optimally. This brings us to a very important conclusion. To apply the D2C method effectively, the dataset needs to be sufficiently large, so that each of the subsets can have enough training data. The larger the dataset, the more effective dividing the training set will be, and the more subsets will be needed to achieve the best possible performance.

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subsets	0.6498	0.6420	2.0112	0.89934	0.5759
N=2, E=1	0.6511	0.6394	1.9101	0.89875	0.5786
N=2, E=2	**0.6567**	0.6444	1.8450	**0.9032**	**0.5853**
N=2, E=3	0.6490	0.6389	1.8718	0.8975	0.5755
N=3, E=1	0.6491	0.6356	1.6540	0.90104	0.5759
N=3, E=2	0.6538	**0.6465**	1.6990	0.89870	0.5817
N=3, E=3	0.6539	0.6428	1.7315	0.9002	0.5820
N=5, E=1	0.6500	0.6368	1.5157	0.90153	0.5772
N=7, E=1	0.6521	0.6432	**1.4821**	0.90148	0.5794

Table 7.5: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on FER2013.

FER2013 is a dataset used for Facial Expression recognition. This task is particularly overfitting prone, and this dataset is quite challenging. In this case, too, we can observe in Table 7.5 that the best accuracy, MCC, and AUC-ROC were achieved

using the D2C method. The number of subsets, N, was only 2 in this case for the best model. The number of epochs before each round of central averaging, E, was 2. The Divide2Conquer approach showed a significant improvement with an accuracy of 65.67% over the traditional approach with no subsets, which yielded an accuracy of 64.98%. Also, as we can see, the optimum number of epochs, E, was 2 in this case, unlike what we saw in the previous datasets where E was always equal to 1 in optimum models. This is because training on **FER2013** is more complicated than the other datasets we used, so just the 1 epoch was not enough to extract meaningful features before each round of central averaging. Hence, more epochs of training before combining the weights from all the subsets resulted in better performance. We can see that there is a trend of decreasing log loss as we increase the number of subsets. However, no such trend is visible consistently when we increase the number of epochs, E. The class distribution of **FER2013**, however, is not balanced. Hence, depending on accuracy only to determine the best model may not be a good idea. So, let us take a look at the F1-scores, a metric that takes into account the class-wise precision.
As we can see from Table 7.5, the best model in terms of F1-score is not the same as the best model in terms of accuracy. The model with 3 training subsets and 2 epochs before each round of central averaging performs the best in terms of F1 score, which means this model performs better when we consider the classes individually. The D2C approach showed a significant improvement in terms of F1 score too, with an F1 score of 0.6465, while the traditional approach with no subsets yielded an F1 score of 0.642. This improvement in the F1-score clearly shows the fact that our proposed method addresses overfitting better than the traditional approach on this dataset.

Audio Classification: Now let us take a look at the performance metrics from the Combined Audio dataset that we used for our audio classification task.

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subset	**0.6953**	0.6870	1.3419	**0.96570**	**0.6692**
N=2, E=1	0.6942	0.6879	1.0811	0.96553	0.6679
N=2, E=2	0.6908	0.6819	1.1636	0.9632	0.6639
N=3, E=1	0.6920	**0.6909**	1.2032	0.96416	0.6656
N=3, E=2	0.6874	0.6830	**1.0033**	0.96438	0.6600
N=4, E=1	0.6921	0.6804	1.2840	0.96214	0.6655

Table 7.6: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on the Combined Audio Dataset.

We can see from Table 7.6 that in the **Combined Audio** dataset, the best accuracy wasn't achieved using our proposed method. The traditional approach of using the entire training set resulted in the best accuracy. The traditional approach yielded a 69.53% accuracy while using 2 training subsets resulted in an accuracy of 69.42%. The accuracy seems to get worse as we increase the number of subsets. We can observe from Table 7.6 that using the D2C method results in less loss, even when the accuracy is a bit worse. So, there is a trade-off here between accuracy and log

loss. While a model with a better accuracy is usually preferred, in case of comparable accuracies, the model with a lower log loss can be an option. However, the **Combined Audio** dataset also has an imbalanced class distribution. So, the true measure of performance and generalization abilities is not accuracy in this case. As we can see from Table 7.6, our proposed method does indeed show a slight improvement over the traditional approach in terms of the F1-score. The number of subsets, N, was 3 in this case for the best model. The number of epochs before each round of central averaging, E, was 1. This best-performing model yielded an F1-score of 0.6909, while the traditional approach with no subsets yielded an F1 score of 0.687. The performance improvement in terms of the F1-score shows that our model generalizes better when we evaluate the model factoring in the performance for each class. Despite this, it is again apparent that dividing the dataset into too many subsets worsens the performance of the models in this case. The reason for this is that the Combined Audio dataset is very small compared to the other datasets we used. However, the complexity it presents is not minimal, especially since this dataset was created by combining multiple datasets having samples collected from different sources. As a result, for effective training, each edge model for the Combined Audio dataset also requires more training samples. When we increase the number of subsets, the size of each training subset becomes smaller, and it starts showing negative impacts on the performance of the model.

This analysis reiterates the conclusion we reached before, that to apply the Divide2Conquer method effectively, the dataset needs to be sufficiently large so that each of the subsets can have enough training data. It is however noteworthy that even with a comparatively smaller dataset, our proposed method still shows some performance improvement in some metrics.

Text Classification: Finally, let us take a look at the performance metrics from the AG News dataset that we used for our text classification task. We used the same Bi-directional LSTM architecture and the same hyperparameters for all the experiments, as mentioned earlier.

Model Settings	*Accuracy*	*F1-Score*	*Log Loss*	*AUC-ROC*	*MCC*
No Subset	0.9243	0.9243	0.2611	0.98764	0.8992
N=2, E=1	0.9291	0.9291	0.2556	0.98873	0.9055
N=3, E=1	0.9288	0.9288	0.2520	0.98872	0.9051
N=3, E=2	0.9275	0.9275	0.2567	0.98852	0.9034
N=5, E=1	0.9293	0.9292	0.2412	0.98881	0.9057
N=5, E=2	**0.9295**	**0.9296**	0.2451	0.98875	**0.9061**
N=7, E=1	0.9271	0.9271	**0.2353**	**0.98916**	0.9028

Table 7.7: Performance Metrics with Traditional Approach vs. Different Settings Using the D2C Method on AG News Dataset.

We can clearly observe from Table 7.7 that the best accuracy, MCC, and F1-Score were achieved using our proposed D2C method. The number of subsets, N, was 5 in this case for the best model. The number of epochs before each round of central

averaging, E, was 2. The D2C method showed a significant improvement with an accuracy of 92.95% over the traditional approach with no subsets, which yielded an accuracy of 92.43%. Using the D2C method decreased the error by more than 0.5% in this case. Also, as we can see from Table 7.7, the trend of a decreasing log loss as we increase the number of subsets is visible here too. The best method in terms of AUC-ROC also happens to be the model with the highest number of subsets (**7**). It also seems that the log loss increases a bit when we increase the number of epochs, E.

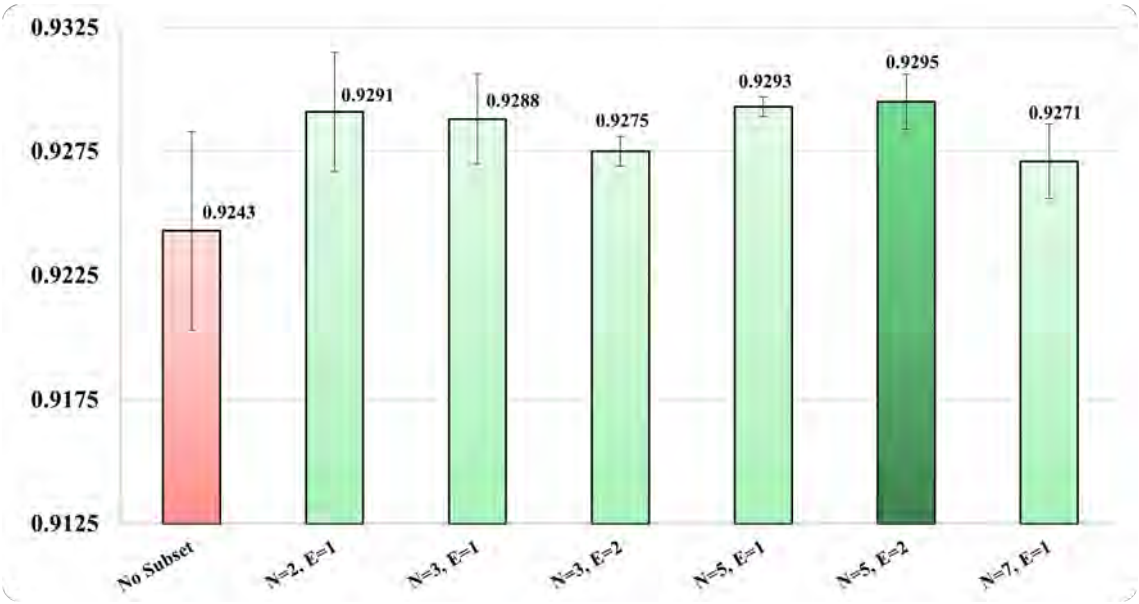

Figure 7.25: Test accuracies for the traditional approach and different values of N and E using the D2C method on the AG News dataset.

The effect of dividing the training set into multiple subsets is the most apparent in this dataset compared to all the previous ones. As can be seen in Figure 7.25, the accuracy improved significantly compared to the traditional approach in every variation of the D2C method (for all values of N and E). The reason for this is quite straightforward. AG News is the largest dataset we have used for our experiments. For each class, we had 30000 samples in this dataset. So, despite dividing the dataset into more and more subsets, each subset had enough data to train on and at the same time take advantage of the regularizing effect that dividing the training set introduces. As a result, the performance of the models improved significantly, addressing the issue of overfitting that usually is prevalent in text classification tasks. The Divide2Conquer method can exploit the large datasets very well in this way, which traditional models are unable to do. This proves the fact that the D2C method can be very impactful in Big Data scenarios.

7.4 Augmentation in the D2C Approach

Our experiments show that dividing training data into multiple subsets and applying central averaging results in less overfitting and an improvement in performance metrics like accuracy, F1 score, etc. However, we also saw that the training subsets

need to be sufficiently large so that each of the subsets can provide enough training diversity to the edge models, otherwise, the performance suffers a drop. When the dataset is large, the proposed method works very well, but when the dataset is smaller, this may cause issues in achieving optimum performance.

One way to counter these issues is data augmentation. This is a widely used technique to expand a dataset and prevent overfitting. To minimize the effect of outliers and noise, and to prevent data leakage among the subsets, the augmentation must take place **after** dividing the dataset into multiple subsets. The augmentation is then applied to each of the subsets. That way, the augmented samples from one subset won't get mixed with another subset. This approach ensures the separation of data in each of the subsets as well as the expansion of training data providing enough samples for all the subsets, thus improving the overall performance. We tested this augmentation approach on the **CIFAR-10** dataset for $3, 5$ & 7 subsets. The augmentation methods we used were rotation, horizontal flip, and shifting along both axes. Let us take a look at the accuracies we achieved in each case and compare them with the accuracies we got before without applying augmentation.

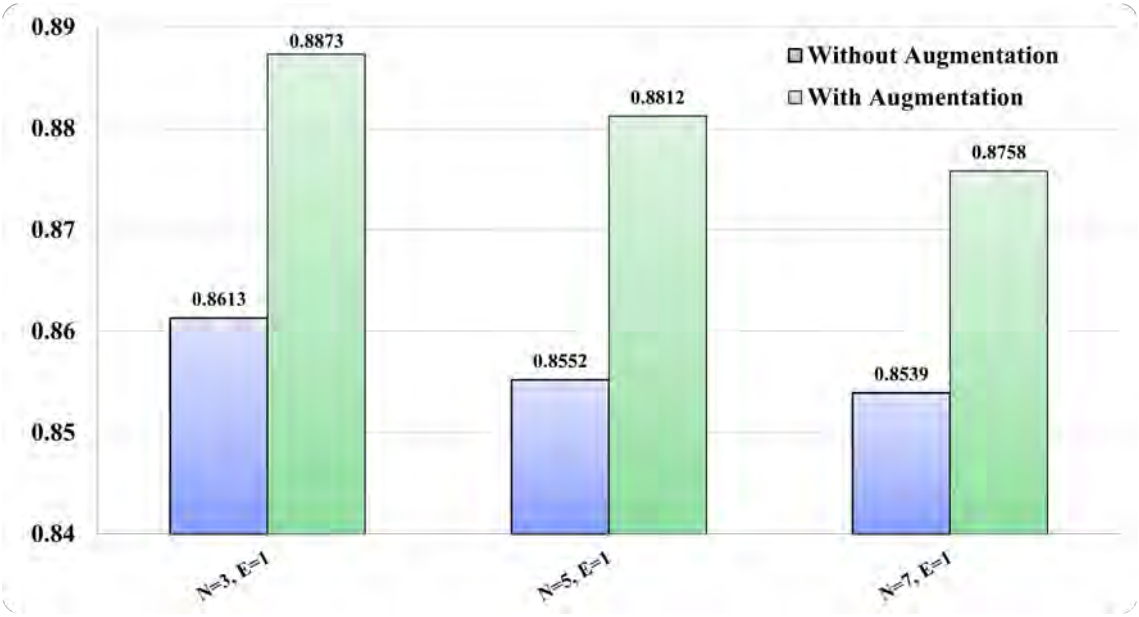

Figure 7.26: Test accuracies for different numbers of subsets, N, using the D2C method on the CIFAR-10 dataset with and without Augmentation.

As we can see from Figure 7.26, the accuracy improved significantly in each case after we applied data augmentation to the subsets of the training data. The accuracy increased from 86.13% to 88.73% for 3 subsets. In the case of 5 subsets, the improvement was even more, reaching 88.12% from just 85.52%. In the case of 7 subsets too, the accuracy increased from 85.39% to 87.58%. This shows that data augmentation techniques can be successfully applied when using the D2C method, and they can effectively treat the problems that arise when the size of the subsets becomes too small.

7.5 Training Time & Inference Time

We have shown that our proposed D2C method treats overfitting and shows an improvement over the traditional approach in multiple domains and datasets. However, all these come at a cost. Dividing the training set into multiple subsets and training them all separately means repeating the whole training process N times (N = number of subsets), albeit with less data. This results in an increased training time. Let us take a look at the following Figure 7.27 for **MNIST**.

Figure 7.27: Time elapsed per epoch for the traditional approach and different numbers of subsets, N, using the D2C method on the MNIST dataset.

The training was done using a V100 GPU in each of these cases for fair comparisons. As we can see from Figure 7.27, the time needed to train the models is the least in the traditional approach, as expected. As we increase the number of subsets, N, the time needed to complete each epoch of training increases. However, it is important to note that the increase in training time is not abrupt. It doesn't increase proportionally with N. For example, the time taken to train for 1 epoch was 12.5 seconds in the case of the traditional implementation. When the number of subsets, N, was 11, it took 40 seconds to train for 1 global epoch. The training time increased significantly, but it was nowhere near 11 times the time taken in the traditional approach. So, even though the training time increases, it doesn't shoot up abruptly. It is important because such an abrupt increase in training time could mean a big issue when it comes to applying the Divide2Conquer method. We also tested the impact on training time for increasing the number of local epochs, E, before each round of central averaging. Let us take a look at the Figure 7.28.
As we can see in Figure 7.28, increasing the number of local epochs, E, expectedly increases the time needed to complete each global iteration of training. In this case, also, the change is not proportional to E. For both N=3 and N =11, the time needed increased as we increased E from 1 to 2, but it didn't reach double the time needed in the case of $E = 1$. As we saw before in Figures 7.10 and 7.11, the convergence happens early too if we increase E. So, the training time becomes even less of an

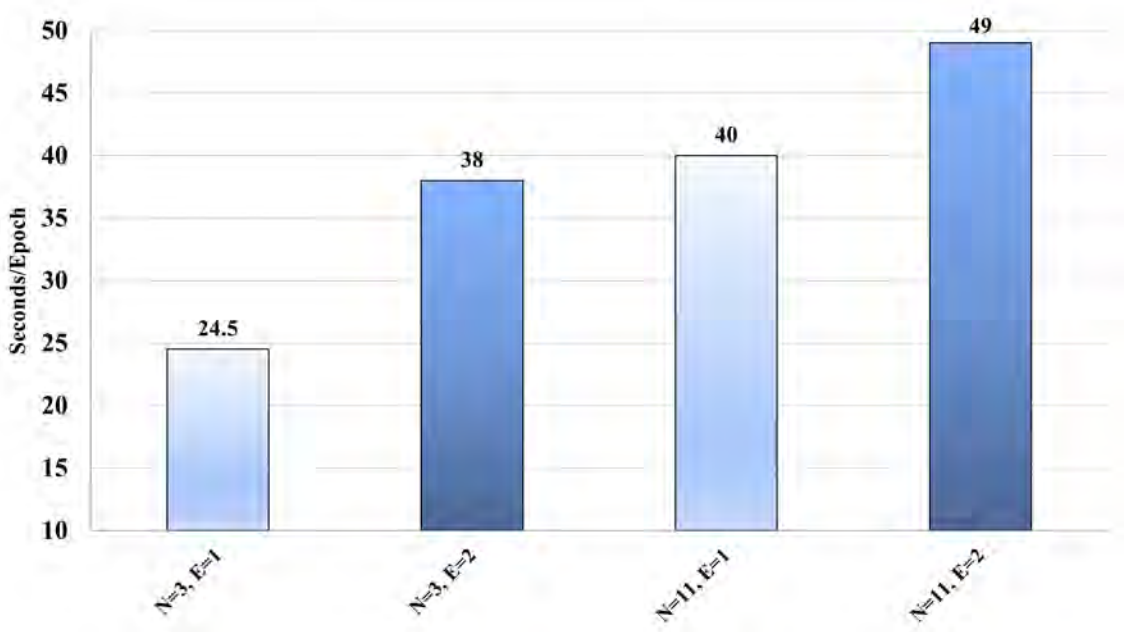

Figure 7.28: Time elapsed per epoch for the traditional approach and different numbers of subsets, N, using the D2C method on the MNIST dataset.

issue.

Lastly, it's important to note that even if the training time is more when using the D2C method, it doesn't make any difference at all in the post-training deployment stage, as the edge models (like the one used in the traditional approach) and the central model have the same number of weights, architecture, and computational complexity. So, clearly, even if **the Training time increases while using the Divide2Conquer method, the Inference time remains exactly the same as the traditional approach**.

Chapter 8

Conclusion

Our study demonstrates that dividing training data into multiple subsets and averaging the weights iteratively can treat overfitting and produce significant performance gains through stability and better generalization. In this study, we applied our proposed Divide2Conquer method and tested it on multiple datasets from different domains. This method outperformed the traditional approach in all the datasets in terms of key metrics and showed significantly better resistance against overfitting. The D2C method also showed a consistent ability to mitigate the increasing trend of validation loss and to produce smoother decision boundaries. These results prove our hypothesis of this method being able to help generalize better and resist overfitting to be true. Moreover, the Divide2Conquer method brings even more improvement when the dataset is larger, outlining its promise in Big Data scenarios, especially since the D2C method can be applied on top of other generalization techniques. Thus, the proposed D2C method opens the door for future research and experiments using this in several domains.

Bibliography

[1] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022 022, Feb. 2019. DOI: 10.1088/1742-6596/1168/2/022022.

[2] N. Morgan and H. Bourlard, "Generalization and parameter estimation in feedforward nets: Some experiments," in *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 1989, pp. 630–637.

[3] L. Prechelt, "Early stopping — but when?" In *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67, ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_5. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_5.

[4] J. Fürnkranz, "Pruning algorithms for rule learning," *Machine Learning*, vol. 27, pp. 139–172, 1997.

[5] M. Bramer, "Using j-pruning to reduce overfitting in classification trees," *Knowledge-Based Systems*, vol. 15, no. 5, pp. 301–308, 2002, ISSN: 0950-7051. DOI: https://doi.org/10.1016/S0950-7051(01)00163-0.

[6] K. Yip and M. Gerstein, "Training set expansion: An approach to improving the reconstruction of biological networks from limited and uneven reliable interactions," *Bioinformatics (Oxford, England)*, vol. 25, pp. 243–50, Dec. 2008. DOI: 10.1093/bioinformatics/btn602.

[7] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1891–1898. DOI: 10.1109/CVPR.2014.244.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.

[9] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2677–2685. DOI: 10.1109/CVPR.2019.00279.

[10] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization:distributed optimization beyond the datacenter," *ArXiv*, 2015. arXiv: 1511.03575 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1511.03575.

[11] K. Korjus, M. Hebart, and R. Vicente, "An efficient data partitioning to improve classification performance while keeping parameters interpretable," *PLOS ONE*, vol. 11, e0161788, Aug. 2016. DOI: 10.1371/journal.pone.0161788.

[12] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial," *ArXiv*, vol. abs/1905.12787, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:170078761.

[13] M. Cogswell, F. Ahmed, R. B. Girshick, C. L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," *Computing Research Repository*, vol. abs/1511.06068, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:2222933.

[14] M. K. Rusia and D. K. Singh, "An efficient cnn approach for facial expression recognition with some measures of overfitting," *International Journal of Information Technology*, vol. 13, pp. 2419–2430, 2021.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, Lille, France, 2015, pp. 448–456.

[16] J. Kolluri, V. K. Kotte, M. Phridviraj, and S. Razia, "Reducing overfitting problem in machine learning using novel l1/4 regularization method," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, 2020, pp. 934–938. DOI: 10.1109/ICOEI48184.2020.9142992.

[17] Q. Zheng, M. Yang, J. Yang, Q. Zhang, and X. Zhang, "Improvement of generalization ability of deep cnn via implicit regularization in two-stage training process," *IEEE Access*, vol. 6, pp. 15 844–15 869, 2018. DOI: 10.1109/ACCESS.2018.2810849.

[18] A. B. Shanmugavel, V. Ellappan, A. Mahendran, M. Subramanian, R. Lakshmanan, and M. Mazzara, "A novel ensemble based reduced overfitting model with convolutional neural network for traffic sign recognition system," *Electronics*, vol. 12, no. 4, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12040926.

[19] G. Wen, Z. Hou, H. Li, D. Li, L. Jiang, and E. Xun, "Ensemble of deep neural networks with probability-based fusion for facial expression recognition," *Cognitive Computation*, vol. 9, pp. 597–610, 2017.

[20] Z. Zeng, Y. Liu, X. Lu, Y. Zhang, and X. Lu, "An ensemble framework based on fine multi-window feature engineering and overfitting prevention for transportation mode recognition," in *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing*, Cancun, Quintana Roo, Mexico: Association for Computing Machinery, 2023, pp. 563–568, ISBN: 9798400702006. DOI: 10.1145/3594739.3610756.

[21] M.-G. Kim, C. Choi, and S. B. Pan, "Ensemble networks for user recognition in various situations based on electrocardiogram," *IEEE Access*, vol. 8, pp. 36 527–36 535, 2020. DOI: 10.1109/ACCESS.2020.2975258.

[22] M. S. B. Siddiqui, S. A. Shusmita, S. Sabreen, and M. G. R. Alam, "Fednet: Federated implementation of neural networks for facial expression recognition," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, 2022, pp. 82–87. DOI: 10.1109/DASA54658.2022.9765165.

[23] X. Zhang, Y. Li, W. Li, K. Guo, and Y. Shao, "Personalized federated learning via variational bayesian inference," in *International Conference on Machine Learning*, 2022, pp. 26 293–26 310.

[24] B. Ghojogh, H. Nekoei, A. Ghojogh, F. Karray, and M. Crowley, "Sampling algorithms, from survey sampling to monte carlo methods: Tutorial and literature review.," *arXiv: Methodology*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:226227376.

[25] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, ISSN: 0885-6125. DOI: 10.1023/A:1018054314350.

[26] P. Bühlmann and B. Yu, "Analyzing bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927–961, 2002. DOI: 10.1214/aos/1031689014. [Online]. Available: https://doi.org/10.1214/aos/1031689014.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *ArXiv*, vol. abs/1708.07747, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:702279.

[30] L. Zahara, P. Musa, E. Prasetyo Wibowo, I. Karim, and S. Bahri Musa, "The facial emotion recognition (fer-2013) dataset for prediction system of micro-expressions face using the convolutional neural network (cnn) algorithm based raspberry pi," in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, 2020, pp. 1–9. DOI: 10.1109/ICIC50835.2020.9288560.

[31] M. K. Pichora-Fuller and K. Dupuis, *Toronto emotional speech set (TESS)*, 2020. DOI: 10.5683/SP2/E8H2MF.

[32] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE Transactions on Affective Computing*, vol. 5, no. 4, pp. 377–390, 2014. DOI: 10.1109/TAFFC.2014.2336244.

[33] S. R. Livingstone and F. A. Russo, "The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english," *PLoS ONE*, vol. 13, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:21704094.

[34] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, Orlando, Florida, USA: Association for Computing Machinery, 2014, pp. 1041–1044, ISBN: 9781450330633. DOI: 10.1145/2647868.2655045.

[35] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, Montreal, Canada, 2015, pp. 649–657.

Appendix A: Relevant Proofs and Derivations

Proof: $\mathbf{Var}(X) = E(X^2) - [E(X)]^2$

The variance of a random variable X is defined as:

$$\text{Var}(X) = E\left[(X - E(X))^2\right]$$

where $E(X)$ is the expected value (mean) of X.

Expanding the square inside the expectation:

$$(X - E(X))^2 = X^2 - 2X \cdot E(X) + [E(X)]^2$$

Thus, the variance becomes:

$$\text{Var}(X) = E\left[X^2 - 2X \cdot E(X) + [E(X)]^2\right]$$

Using the linearity of expectation, we can separate the terms:

$$\text{Var}(X) = E(X^2) - E(2X \cdot E(X)) + E([E(X)]^2)$$

Since $E(X)$ is a constant, we have:

$$E(2X \cdot E(X)) = 2E(X) \cdot E(X) = 2[E(X)]^2$$

and

$$E([E(X)]^2) = [E(X)]^2$$

Thus, the expression simplifies to:

$$\text{Var}(X) = E(X^2) - 2[E(X)]^2 + [E(X)]^2$$

Simplifying further:

$$\text{Var}(X) = E(X^2) - [E(X)]^2$$

Hence, we have proved that:

$$\text{Var}(X) = E(X^2) - [E(X)]^2$$

Proof: $\mathbf{Cov}(X, Y) = E(X \cdot Y) - E(X)E(Y)$

The covariance of two random variables X and Y is defined as:

$$\text{Cov}(X, Y) = E\left[(X - E(X)) \cdot (Y - E(Y))\right]$$

where, $E(X)$ and $E(Y)$ are the expected values (means) of X and Y, respectively.

Expanding the product inside the expectation:

$$(X - E(X)) \cdot (Y - E(Y)) = X \cdot Y - X \cdot E(Y) - E(X) \cdot Y + E(X) \cdot E(Y)$$

Thus, the covariance becomes:

$$\text{Cov}(X, Y) = E\left[X \cdot Y - X \cdot E(Y) - E(X) \cdot Y + E(X) \cdot E(Y)\right]$$

Using the linearity of expectation, we can split this into four terms:

$$\text{Cov}(X, Y) = E(X \cdot Y) - E(X \cdot E(Y)) - E(E(X) \cdot Y) + E(E(X) \cdot E(Y))$$

Since $E(X)$ and $E(Y)$ are constants, we can simplify:

$$E(X \cdot E(Y)) = E(Y) \cdot E(X), \quad E(E(X) \cdot Y) = E(X) \cdot E(Y), \quad E(E(X) \cdot E(Y)) = E(X) \cdot E(Y)$$

Thus, the expression simplifies to:

$$\text{Cov}(X, Y) = E(X \cdot Y) - E(X) \cdot E(Y)$$

Hence, we have proved that:

$$\text{Cov}(X, Y) = E(X \cdot Y) - E(X) \cdot E(Y)$$

Proof: $\mathbf{Var(cX) = c^2 Var}(X)$

The variance of a random variable X is defined as:

$$\text{Var}(X) = E[(X - E[X])^2]$$

Now, consider the random variable cX. We want to find $\text{Var}(cX)$:

$$\text{Var}(cX) = E[(cX - E[cX])^2]$$

We know that:

$$E[cX] = cE[X]$$

Thus, we can rewrite the variance:

$$\text{Var}(cX) = E\left[(cX - cE[X])^2\right]$$

Factoring out c from the expression inside the expectation gives:

$$\text{Var}(cX) = E\left[(c(X - E[X]))^2\right]$$

Using the property of expectations that $E[aY] = aE[Y]$ for a constant a, we can rewrite this as:

$$\text{Var}(cX) = E\left[c^2(X - E[X])^2\right]$$

Now, since c^2 is a constant, we can take it outside the expectation:

$$\text{Var}(cX) = c^2 E\left[(X - E[X])^2\right]$$

Recognizing the definition of variance again, we have:

$$\text{Var}(cX) = c^2 \text{Var}(X)$$

Thus, we have proved that:

$$\text{Var}(cX) = c^2 \text{Var}(X)$$

Proof: $\mathbf{Var}\left(\sum_{i=1}^{k} X_i\right) = \sum_{i=1}^{k} \mathbf{Var}(X_i) + \sum_{1 \le i \ne j \le k} \mathbf{Cov}(X_i, X_j)$

To prove the formula for the variance of the sum of k random variables $X_1, X_2, \ldots, X_k$, we start from the definition of variance and covariance.

Let $X_1, X_2, \ldots, X_k$ be random variables. The variance of their sum is given by:

$$\text{Var}\left(\sum_{i=1}^{k} X_i\right) = E\left[\left(\sum_{i=1}^{k} X_i - E\left[\sum_{i=1}^{k} X_i\right]\right)^2\right]$$

First, we compute $E\left[\sum_{i=1}^{k} X_i\right]$:

$$E\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} E[X_i]$$

Substituting this into the variance expression:

$$\text{Var}\left(\sum_{i=1}^{k} X_i\right) = E\left[\left(\sum_{i=1}^{k} X_i - \sum_{i=1}^{k} E[X_i]\right)^2\right]$$

Next, let $\mu_i = E[X_i]$. We can rewrite the expression:

$$\text{Var}\left(\sum_{i=1}^{k} X_i\right) = E\left[\left(\sum_{i=1}^{k} (X_i - \mu_i)\right)^2\right]$$

Expanding the square:

$$= E\left[\sum_{i=1}^{k} (X_i - \mu_i)^2 + 2 \sum_{1 \le i < j \le k} (X_i - \mu_i)(X_j - \mu_j)\right]$$

By the linearity of expectation, this becomes:

$$= \sum_{i=1}^{k} E\left[(X_i - \mu_i)^2\right] + 2 \sum_{1 \le i < j \le k} E\left[(X_i - \mu_i)(X_j - \mu_j)\right]$$

Recognizing the definitions of variance and covariance:

$$= \sum_{i=1}^{k} \text{Var}(X_i) + \sum_{1 \le i < j \le k} (\text{Cov}(X_i, X_j) + \text{Cov}(X_j, X_i))$$

Since $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$, we can combine the covariance terms:

$$= \sum_{i=1}^{k} \text{Var}(X_i) + \sum_{1 \le i \ne j \le k} \text{Cov}(X_i, X_j)$$

Thus, we have proven that:

$$\text{Var}\left(\sum_{i=1}^{k} X_i\right) = \sum_{i=1}^{k} \text{Var}(X_i) + \sum_{1 \le i \ne j \le k} \text{Cov}(X_i, X_j)$$

Appendix B: Supplementary Materials & Codes

Our codes and some supplementary materials are publicly available at: https://github.com/Saiful185/Divide2Conquer.