




# An Extensive Study on D2C: Overfitting Remediation in Deep Learning Using a Decentralized Approach

Md. Saiful Bari Siddiqui , *Graduate Student Member, IEEE*, Md Mohaiminul Islam , *Graduate Student Member, IEEE*, Md. Golam Rabiul Alam , *Member, IEEE*.

**Abstract**—Overfitting remains a significant challenge in deep learning, often arising from data outliers, noise, and limited training data. To address this, we propose Divide2Conquer (D2C), a novel technique to mitigate overfitting. D2C partitions the training data into multiple subsets and trains identical models independently on each subset. To balance model generalization and subset-specific learning, the model parameters are periodically aggregated and averaged during training. This process enables the learning of robust patterns while minimizing the influence of outliers and noise. Empirical evaluations on benchmark datasets across diverse deep-learning tasks demonstrate that D2C significantly enhances generalization performance, particularly with larger datasets. Our analysis includes evaluations of decision boundaries, loss curves, and other performance metrics, highlighting D2C’s effectiveness both as a standalone technique and in combination with other overfitting reduction methods. We further provide a rigorous mathematical justification for D2C’s underlying principles and examine its applicability across multiple domains. Finally, we explore the trade-offs associated with D2C and propose strategies to address them, offering a holistic view of its strengths and limitations. This study establishes D2C as a versatile and effective approach to combating overfitting in deep learning. Our codes are publicly available at: <https://github.com/Saiful185/Divide2Conquer>.

**Index Terms**—Classification, Deep Learning, Hyperparameter, Overfitting, Regularization, Variance.

## I. INTRODUCTION

DEEP Learning models often do well on the data they train on, but the performance drops massively on unseen data from different distributions. This phenomenon is known as Overfitting. Many methods have been proposed to reduce overfitting over the years [10]. Early Stopping [27] is one of the most intuitive methods for reducing overfitting. However, it limits the model’s learning potential [28]. Network reduction reduces variance by simplifying the model, but it restricts learning of complex features [29]. Data augmentation is a common approach to mitigate overfitting in deep learning applications [32]. However, choosing suitable augmentation

techniques for specific datasets can be challenging, and obtaining additional training data often requires substantial effort. Regularization is a widely used technique to reduce overfitting by penalizing model dependency on training data. Dropout, the most relevant regularization method in deep learning, reduces overfitting by randomly dropping connections between neural network layers [4]. However, even dropout can sometimes degrade the performance of a model due to internal covariance shifts [18]. Eventually, despite employing these techniques, the model still has access to the entire training data, 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 different portions of the data.

We propose a novel method, Divide2Conquer (D2C), which proposes dividing the training data into multiple subsets and training a model (all having the same architecture & hyperparameters) on each subset. A weighted averaging of all the parameter weights from the instance model is performed after a certain number of epochs, and the averaged weights are shared back to each of the models (we denote them as **edge models** in this study). Then, the whole process is repeated for several global epochs.

D2C is loosely inspired by Federated Optimization [2], which involves training identical models on local devices (using the data available on those devices) and periodically aggregating and averaging their trained parameters on a central server. This process ensures safe access to data and preserves privacy, as it only shares the weights, not the data itself. However, unlike D2C, federated learning is not inherently designed to enhance model generalization.

The D2C method also has similarities with Bagging [23] 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 datapoints 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.

Our experiments encompass multiple datasets across differ-

This preprint has been submitted for publication at IEEE Transactions on Big Data. The authors acknowledge the permission granted by IEEE to post this preprint on arXiv. The final version of this paper, once published, will be available at IEEE Xplore.

Md. Saiful Bari Siddiqui and Md. Golam Rabiul Alam are with the Department of Computer Science and Engineering, Brac University, Dhaka, Bangladesh (e-mail: saiful.bari@bracu.ac.bd; rabiul.alam@bracu.ac.bd).

Md Mohaiminul Islam is with the Department of Computer Science and Engineering, United International University, Dhaka, Bangladesh (e-mail: mohaiminul@cse.uui.ac.bd).

ent domains, and each case suggests that employing the D2C method can reduce overfitting significantly. The primary comparison to evaluate the method 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 Divide2Conquer 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 present a mathematical justification for the hypothesis that D2C mitigates overfitting by rigorously demonstrating its capacity to reduce model variance. This formal analysis underscores D2C's potential to improve model robustness across diverse datasets.
- D2C can be applied on top of other data augmentation and regularization techniques, and our experiments across various datasets and tasks show that this results in a clear improvement in the model's generalization ability.
- 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.

The paper is organized as follows. In section II, we discuss relevant literature. We establish the theoretical justification behind our hypothesis in section III. In sections IV and V, we discuss our methodology and lay out the experimental specifications. In section VI, we evaluate our approach through empirical experiments on multiple datasets and analyze the results. We discuss the limitations of D2C in section VII. Section VIII contains concluding remarks.

## II. RELATED WORKS

Several authors have addressed the issue of overfitting while performing various tasks. M. Cogswell et al. proposed a regularizer called DeCov, which helps reduce overfitting in deep neural networks by Decorrelating Representations [5].

Dropout [4] 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 [8] 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  $L^{1/4}$  regularization to solve the problems faced by L1 and L2 regularization techniques [15].

Ensembles are also often used to improve generalization. The ensemble-based ELVD model [3] managed to outperform the traditional VGGNet and DropoutNet models in terms of reducing overfitting. Zehong Zeng et al. came up with an ensemble framework that incorporates several techniques to prevent overfitting [9]. 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 [1].

We implemented a method based on federated optimization preliminarily for facial expression recognition using FedNet [17]. This model achieves excellent results in terms of generalization on both CK+ and FER-2013 datasets. The K-Fold-Cross-Validation method for evaluation proposed by Korjus K. et al. [13] 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. The proposed 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 bar one, minimizing the impact of these erroneous samples.

Overfitting remains a prevalent issue in supervised machine learning despite methods like Early Stopping, Network Reduction, Training Set Expansion, Regularization, and Dropout being effective [10]. Through D2C, we build an approach that focuses on achieving even better generalization and can be implemented on top of other overfitting-reducing techniques.

## III. THEORETICAL FRAMEWORK

### A. The Hypothesis

Overfitting happens when a model fits too well on the data that it trains on. It effectively captures and memorizes even the random characteristics from the training data, randomness that would be insignificant in real-world applications. The presence of outliers and noisy data is a fundamental reason behind overfitting. In this study, we formulate a method to minimize the effect of these outliers and noisy data.

The contribution of outliers/noise can be minimized by dividing the training data into multiple shards 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. Aggregation and averaging can be done periodically to combine all the models. In the best-case scenario, due to averaging, the effect of the individual outliers/noise would be reduced by a factor of  $N$ , where  $N$  is the number of data shards. The key factors behind our hypothesis and considerations regarding this method being able to address overfitting are discussed as follows:

1. Weighted averaging of parameters helps in combining the knowledge learned from different subsets. 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.

So, the averaged parameters should reflect a more balanced representation of the underlying data distribution, mitigating the influence of individual noisy or outlier samples.

2. Averaging the parameter weights is particularly helpful in combating overfitting since it has a regularizing effect. Averaging weights regularizes the model at the parameter level. It moderates extreme weight 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 central model converges 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.

3. Averaging parameters smooths 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 and smoothed when combined through averaging, consequently helping in generalizing well to unseen data and reducing overfitting.

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. Suppose the edge model is complex or the subset count is too high. In that case, 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. This is similar to what happens in federated learning when the client data is minimal, as pointed out by Zhang et al. [34]. When sufficient data exists per class, more subsets can be beneficial, as noise or outliers are diluted by a factor of up to  $N$  (number of training subsets).

In summary, D2C helps to reduce overfitting by mitigating the effects of noise and outliers. However, dividing the training set into too many subsets can lead to insufficient data per subset, which can hamper the overall performance.

### B. Mathematical Intuition

To lay out the underlying intuition behind our hypothesis, we must first define overfitting formally. We follow the groundwork laid out by Ghogh et al. [23]. Let us consider a Dataset  $D$  with training set  $T$  and test set  $R$  so that,

$$T \cup R = D \quad (1)$$

$$T \cap R = \emptyset \quad (2)$$

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. However, 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  $\epsilon_i$ . So, we can say,

$$y_i = f_i + \epsilon_i \quad (3)$$

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

$$\mathbb{E}(\epsilon_i^2) = Var(\epsilon_i) + [\mathbb{E}(\epsilon_i)]^2 \Rightarrow \mathbb{E}(\epsilon_i^2) = \sigma^2 + 0 = \sigma^2 \quad (4)$$

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 as 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:  $\mathbf{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \mathbb{E}((\hat{y}_i - y_i)^2)$ . 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  $\hat{y}_0$ . Ghogh et al. [23] 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 -

$$\mathbf{MSE} = \mathbb{E}((\hat{y}_0 - y_0)^2) = \mathbb{E}((\hat{y}_0 - f_0)^2) + \sigma^2 \quad (5)$$

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. (5), we have:

$$\begin{aligned} \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2 &= \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - f_i)^2 + \sigma^2 \\ \Rightarrow \sum_{i=1}^m (\hat{y}_i - y_i)^2 &= \sum_{i=1}^m (\hat{y}_i - f_i)^2 + m\sigma^2 \end{aligned} \quad (6)$$

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 Ghogh et al. [23] showed, if  $(x_0, y_0) \in T$ , deriving the M.S.E yields:

$$\mathbf{MSE} = \mathbb{E}((\hat{y}_0 - y_0)^2) = \mathbb{E}((\hat{y}_0 - f_0)^2) + \sigma^2 - 2\sigma^2 \mathbb{E}\left(\frac{\partial \hat{y}_0}{\partial y_0}\right) \quad (7)$$

Using the same approximation as Eq.(6) on the training set  $T$ , we get:

$$\begin{aligned}
\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - f_i)^2 + \sigma^2 - 2\sigma^2 \frac{1}{n} \sum_{i=1}^n \frac{\partial \hat{y}_i}{\partial y_i} \\
\Rightarrow \sum_{i=1}^n (\hat{y}_i - y_i)^2 &= \sum_{i=1}^n (\hat{y}_i - f_i)^2 + n\sigma^2 - 2\sigma^2 \sum_{i=1}^n \frac{\partial \hat{y}_i}{\partial y_i} \quad (8)
\end{aligned}$$

From Eq.(8) 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  $e = E + n\sigma^2 - 2\sigma^2 \sum_{i=1}^n \frac{\partial \hat{y}_i}{\partial y_i}$ . Hence, even if we can obtain a small value of  $e$ , the term  $2\sigma^2 \sum_{i=1}^n \frac{\partial \hat{y}_i}{\partial y_i}$  can grow large as the training progresses and hide the true value of a substantial and large true error  $E$  (Figure 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.(8) for empirical error. We can conclude from this analysis that the final term in Eq.(7) 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.  $\frac{\partial \hat{y}_i}{\partial y_i}$  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.(6). Figure 2 depicts a visualization of this idea.

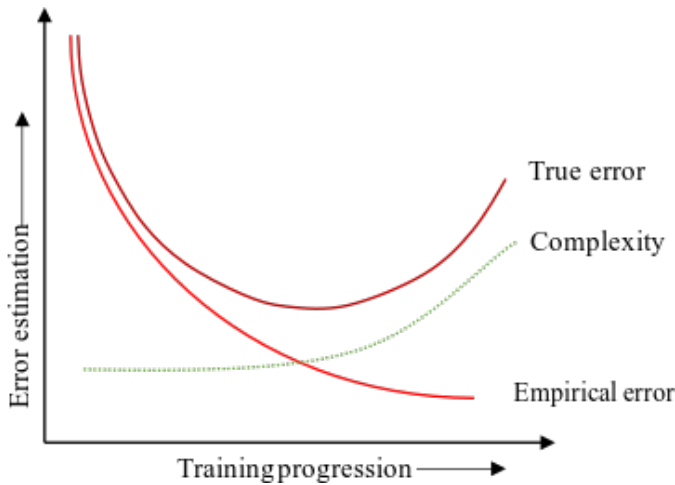


Fig. 1. When the corresponding complexity term grows and dominates the expression in Eq.(8) for empirical error, the model starts to overfit as empirical error becomes no longer a true representation of true error.

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 2(a) and 2(c) show how an underfitting and overfitting model shows low and high variance respectively. Figure 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  $|T_1| = |T_2| = \dots = |T_k| = |T|/k$ . Then, we train the model  $\hat{f}_j$  using the  $j$ -th data chunk, where  $j \in \{1, \dots, k\}$ . 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  $\mathbf{W}_j = \{w_{j1}, \dots, w_{jn}\}$  and bias  $b_j$ . For an input vector  $\mathbf{x} = \{x_1, \dots, x_n\}$  the prediction of the edge model on the  $t$ -th epoch can be described as:  $\hat{f}_j(\mathbf{x}) = \mathbf{W}_j^t \mathbf{x}^T + b_j$ . In the case of a single global epoch, we now wish to aggregate the weight matrices creating the central weight matrix  $\bar{\mathbf{W}} = \{\bar{w}_1, \dots, \bar{w}_n\}$ . Here, the weight matrix for the  $(t+1)$ -th epoch is:

$$\bar{\mathbf{W}}^{t+1} = \frac{1}{k} \sum_{j=1}^k \mathbf{W}_j^t \text{ and } b^{t+1} = \frac{1}{k} \sum_{j=1}^k b_j^t \quad (9)$$

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

$$\begin{aligned}
\hat{f}(\mathbf{x})^{t+1} &= \bar{\mathbf{W}}^{t+1} \mathbf{x}^T + b^{t+1} \\
\Rightarrow \hat{f}(\mathbf{x})^{t+1} &= \frac{1}{k} \sum_{j=1}^k \mathbf{W}_j^t \mathbf{x}^T + \frac{1}{k} \sum_{j=1}^k b_j^t \\
\Rightarrow \hat{f}(\mathbf{x})^{t+1} &= \frac{1}{k} \sum_{j=1}^k (\mathbf{W}_j^t \mathbf{x}^T + b_j^t) \\
\Rightarrow \hat{f}(\mathbf{x})^{t+1} &= \frac{1}{k} \sum_{j=1}^k \hat{f}_j(\mathbf{x})^t \quad (10)
\end{aligned}$$

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,  $\mathbb{E}(e_{jt}) = 0$  and  $\text{Var}(e_{jt}) = s$ . Let us denote the estimation



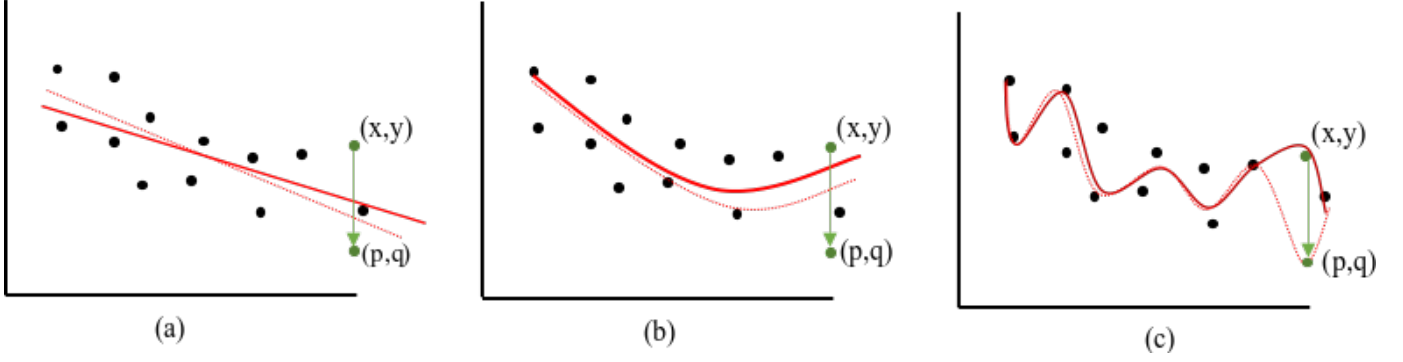


Fig. 2. A visualization of (a) Underfitting, (b) A good fit, and (c) Overfitting scenarios, illustrating how the model changes if an arbitrary training data point  $(x, y)$  is shifted to  $(p, q)$  by plotting the trained model before (solid line) and after (dashed line) the shift.

of the product of two trained models using two different data chunks by  $c$ , i.e.,  $\mathbb{E}(e_{jt} \times e_{lt}) = c$ . So, we have:

$$\mathbb{E}(e_{jt}^2) = \text{Var}(e_{jt}) + [\mathbb{E}(e_{jt})]^2 \Rightarrow \mathbb{E}(e_{jt}^2) = s + 0 = s \quad (11)$$

Again for  $j, l \in \{1, \dots, k\}$ , where  $j \neq l$ , we have:

$$\begin{aligned} \text{Cov}(e_{jt}, e_{lt}) &= \mathbb{E}(e_{jt} \times e_{lt}) - \mathbb{E}(e_{jt}) \mathbb{E}(e_{lt}) \\ \Rightarrow \text{Cov}(e_{jt}, e_{lt}) &= c - 0 \times 0 = c \\ \Rightarrow \text{Cov}(\hat{f}(\mathbf{x})_j^t, \hat{f}(\mathbf{x})_l^t) &= c - 0 \times 0 = c \end{aligned} \quad (12)$$

From Eq.(10) and Eq.(12), we can deduce:

$$\begin{aligned} \text{Var}(\hat{f}(\mathbf{x})^{t+1}) &= \frac{1}{k^2} \text{Var} \left( \sum_{j=1}^k \hat{f}(\mathbf{x})_j^t \right) \\ &= \frac{1}{k^2} \sum_{j=1}^k \text{Var}(\hat{f}(\mathbf{x})_j^t) + \frac{1}{k^2} \sum_{i=1, l=1}^k \\ &\quad \sum_{i \neq l}^k \text{Cov}(\hat{f}(\mathbf{x})_i^t, \hat{f}(\mathbf{x})_l^t) \\ &= \frac{1}{k^2} sk + \frac{1}{k^2} ck(k-1) = \frac{s + c(k-1)}{k} \end{aligned} \quad (13)$$

Relevant proofs related to the derivation of Eq. 12 and 13 are provided in Appendix A. Eq.(13) provides us with an interesting insight. If the two edge 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:  $\text{Var}(\hat{f}(\mathbf{x})^{t+1}) \approx (s + s(k-1))/k \approx s$ . On the other hand, if the two models are very dissimilar the same expression gives us:  $\text{Var}(\hat{f}(\mathbf{x})^{t+1}) \approx (s + 0(k-1))/k \approx s/k$ . As  $\text{Var}(e_{jt}) = \text{Var}(\hat{f}(\mathbf{x})^t) = s$ , 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) [4]. 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.

#### IV. PROPOSED METHODOLOGY

##### A. 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 in this paper.** At first, the training set is randomly shuffled and then divided into multiple subsets, maintaining consistent class distribution. Each subset is then fed into a separate edge model with

identical neural network architecture and trainable parameters to enable parameter averaging.

### B. Training & Averaging of Trained Parameters

Instead of one training loop, the implementation of our method requires two loops: an inner loop for subset training which is nested within an outer loop that performs central averaging. A central model with the same architecture as the edge models is first initialized. In total, we have one Central Model and  $N$  Edge models, all of which share an identical architecture, where  $N$  is the number of subsets we have divided the training set into. Initially, these identical model objects are built and loaded for each training subset. Within the inner loop, each edge model is iteratively trained on its subset for a specified number of local epochs. Then, the edge models' parameters are scaled based on their data fraction and added to a local weight list. In the outer loop, these scaled parameters are averaged, updating the central model's parameters. Each edge model's weights are then reinitialized to match the central model, completing a global training epoch. This process repeats over several global epochs, with weighted parameter averaging to account for the data fraction of each subset. The whole process is summarized in Figure 3, and the pseudo-code to implement D2C is provided in Algorithm 1.

### C. Global Hyperparameter Tuning

Tuning the overall model consists of two stages. Firstly, the edge model 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 Global 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, log loss, and also the validation loss curve. The appropriate number of training epochs in each subset before the averaging is done each time is likely to be a small number ( $1 \sim 3$ ) since the edge model can 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.

## V. EXPERIMENTAL SETUP

### A. Datasets

For image classification tasks, we employed three widely used benchmark datasets: CIFAR-10 [14], MNIST [7], and Fashion MNIST [12]. CIFAR-10 provides a challenging image classification task with 60,000 32x32 RGB images across 10 categories. MNIST offers a simpler task with

---

### Algorithm 1 Divide2Conquer

---

**Input:** Training dataset  $D$ , Number of subsets  $N$ , Local epochs  $E$ , Global epochs  $E_{global}$ , Batch size  $B$ , Learning rate  $lr$

**Output:** Central model  $M_c$  with aggregated weights

**function** Divide2Conquer( $D, N, E, E_{global}, B, lr$ )

    Shuffle the dataset  $D$

    Divide  $D$  into  $N$  subsets:  $\{D_1, D_2, \dots, D_N\}$ , maintaining class distributions

    Initialize central model  $M_c$  with parameter vector  $\theta_c$

**for**  $i = 1$  **to**  $N$  **do**

        Initialize edge model  $M_i$  with  $\theta_i = \theta_c$

        Set scaling factor  $s_i = \frac{|D_i|}{|D|}$

**end for**

**for**  $g = 1$  **to**  $E_{global}$  **do**

        Initialize weight accumulator  $W_{central} = 0$

**for**  $i = 1$  **to**  $N$  **do**

            Set  $\theta_i = \theta_c$

**for**  $e = 1$  **to**  $E$  **do**

                Train  $M_i$  on  $D_i$  using batch size  $B$  and learning rate  $lr$

**end for**

            Obtain updated weights  $\theta_i$  from  $M_i$

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

            Accumulate scaled weights into  $W_{central}$ :

$W_{central} = W_{central} + \theta_i$

**end for**

        Compute averaged central weights:

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

        Update central model  $M_c$  with new weights  $\theta_c$

**end for**

**return** central model  $M_c$

**end function**

---

70,000 28x28 grayscale images of handwritten digits. Fashion MNIST presents a more complex variation with 70,000 28x28 grayscale images of fashion items. To address the overfitting challenge in facial expression recognition, we utilized the FER-2013 [16] dataset, which comprises over 35,000 grayscale images of seven basic emotions. The diverse and noisy nature of this internet-sourced dataset makes it particularly suitable for evaluating generalization performance.

For audio-based tasks, we created a comprehensive dataset by combining TESS [19], CREMA-D [20], and RAVDESS [21]. This dataset encompasses 14 emotion classes, offering a diverse range of emotional expressions and recording conditions. By merging these datasets, we aim to improve model generalization by mitigating biases and limitations inherent in individual datasets.

To test our method on text data, we employed the AG News [22] dataset, which consists of approximately 127,600 news articles categorized into four balanced classes: World, Sports, Business, and Science/Technology. This dataset pro-

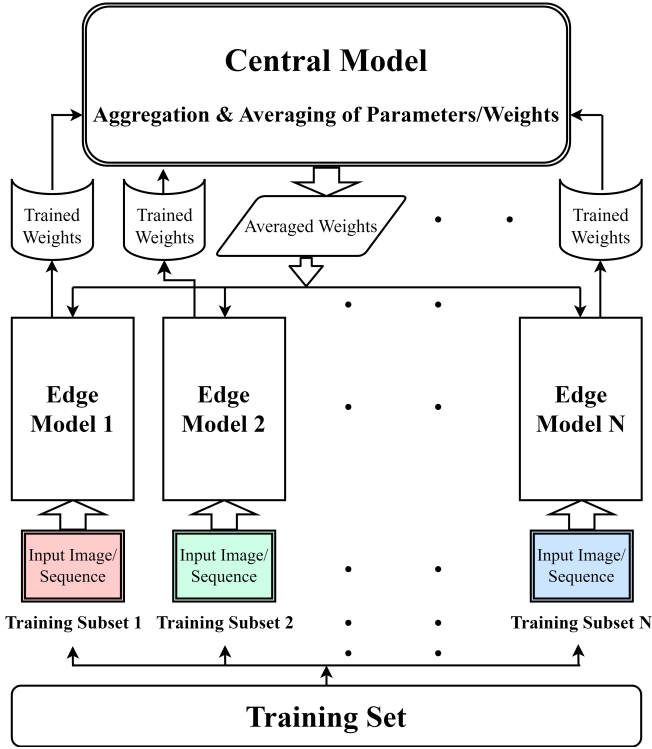


Fig. 3. D2C at a Glance – Training on each subset, averaging the trained parameters, setting the Central model parameters to these values, and sharing the averaged parameters back to the Edge models to keep the loop running.

vides a large-scale, real-world benchmark for evaluating text classification models, particularly in the context of Big Data.

### B. Experimental Details

**1. Image Classification:** At first, all the images are reshaped and normalized. We applied one-hot encoding to the labels to use cross-entropy loss later. We split the training data into training and validation sets using Scikit-Learn. 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. Each of our edge models was 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 identical convolutional layers and one pooling layer.

**2. 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.

**3. Text Classification:** In this case, we created a corpus using all the words occurring in our 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. In the case

of these 1D text sequences, the edge model we used consisted of 3 bidirectional LSTM layers (128, 64, and 32 units) and two dense layers (128 and 64 neurons) 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.

In all cases, we used a 90-10 Training-Validation split, Adam optimizer, and applied dropouts and batch normalization layers. This is important to test for D2C's effectiveness when used on top of other methods that address overfitting. We used the predefined test sets for testing.

## VI. RESULTS AND DISCUSSIONS

We will use loss and accuracy curves, Accuracy, F1 Score, AUC-ROC, and log loss to analyze the results and evaluate our method. We also varied the two new global hyperparameters we mentioned in the previous section: 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  $N$  and  $E$  respectively in this section.

### A. Visualizing Decision Boundary

At first, we will visualize the change in decision boundaries due to applying the D2C method and analyze the findings. We used a synthetic binary classification dataset to simulate a simple decision boundary. This dataset was 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 three hidden layers having 100 neurons each. No regularization or data augmentation was used for decision boundary visualization, to observe the effect of D2C explicitly.

At first, we adopted the centralized, traditional approach and fed the training dataset to the sole model. After training, we got the decision boundary shown in Figure 4a.

As we can see, the decision boundary for the traditional approach is highly complex and non-linear here. 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 means that this approach is likely to produce excellent results on training data. Still, 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.

Then, to see the effect of D2C on decision boundaries, we divided our training set into multiple subsets and trained them parallelly using identical **Edge Models**. Figure 5 shows the decision boundaries we got from each edge model.

As we can see, the decision boundaries from the edge models are still complex, containing intricate regions following

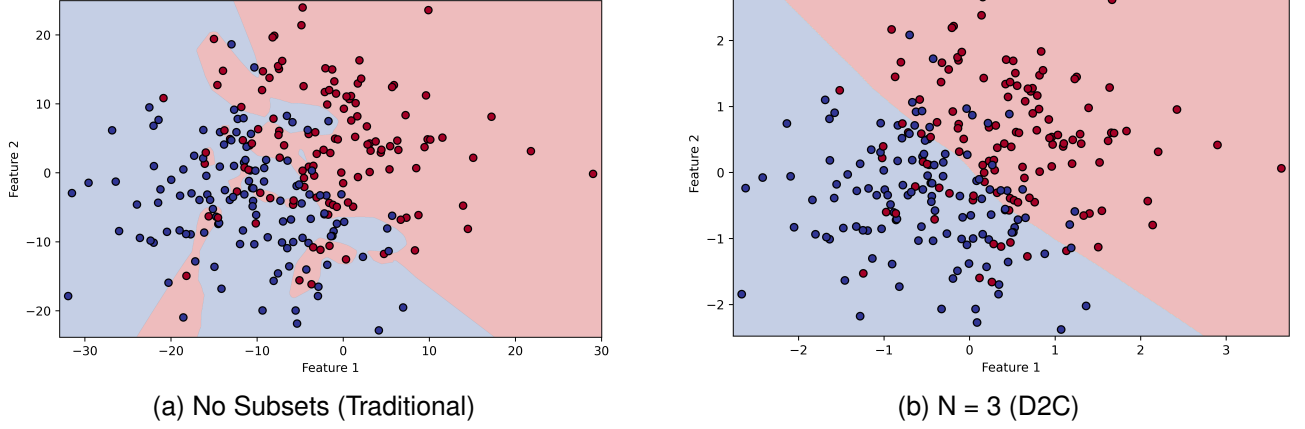


Fig. 4. Decision boundary Visualization with the Traditional approach vs. the Divide2Conquer method.

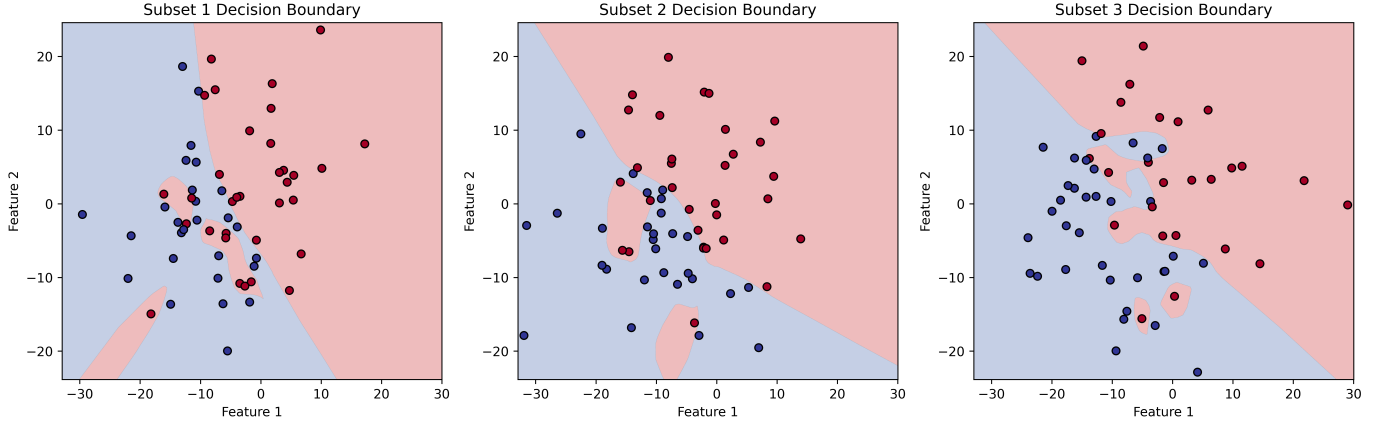


Fig. 5. Decision Boundary from the edge models trained on each subset after dividing the training data into 3 subsets applying the D2C method.

individual noisy data points. However, each of these noisy samples occurs only once in one of the subsets, affecting the corresponding edge model. So, irregularity around an individual noisy sample does not affect all the edge models but rather only one of them. Hence, after averaging the weights, the impact of the individual outliers should be diluted.

To verify, the trained weights from each subset were averaged to get the final weights for the **Central Model**. The resulting decision boundary is shown in Figure 4b.

We can see that the decision boundary from the central model of D2C is much simpler, dividing the feature space almost diagonally. This suggests a less flexible, more generalized model due to averaging the weights from multiple subsets. This boundary is smooth and does not follow the exact positions of individual data points as closely. The model is more generalized, focusing on the overall trend in the data rather than trying to accommodate every single datapoint, 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. An individual noisy sample does not affect all the edge models in D2C, and its impact is diluted through averaging. This leads the central model to create a

decision boundary based on the more common patterns present in all the subsets, ensuring better generalization. Further proof of this generalization is seen in the training and test accuracies for the traditional approach and D2C using different numbers of subsets. We summarize these accuracies in Table I.

TABLE I  
PERFORMANCE OF THE TRADITIONAL APPROACH VS DIFFERENT VALUES OF  $N$  USING THE D2C METHOD.

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

As we can see, we got almost perfect accuracy on the training set with the traditional approach. However, the significant drop in test accuracy (0.6833) proves that the model is overfitting to the training data. When we use D2C with 2, 3, or 4 subsets, we observe a clear improvement and a much better balance between the training and test accuracies. For all these cases, the training and test accuracies are similar, and the test accuracy for 3 or 4 subsets (0.8333) surpasses that of the traditional approach by a significant margin. This



indicates that splitting the data into 3 or 4 subsets provides a good balance between learning patterns from training data and reducing overfitting to improve the model’s ability to generalize to unseen data.

Our observations prove that models using the D2C approach 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. So, D2C can be seen as yet another technique to reduce overfitting. However, several other techniques address overfitting. So, whether using D2C on top of other regularization and augmentation techniques (such as dropouts) improves the performance of a model is the key question in application level. In the following subsections, we will investigate this using the architectures (incorporating regularization methods like dropouts in the edge models) we described in the previous section on several datasets from different domains.

### B. Analyzing Loss & Accuracy Curves

We will analyze the results obtained from our experiments from the 3 different domains. The loss curve is a definitive indicator of overfitting. As we saw in Eq. (6), The validation/test loss truly represents the *true error* of the model. So we will monitor the validation loss curves from the different experiments we performed to compare and analyze if D2C improves the generalizing performance, in the form of minimizing the validation loss as the training progresses.

**Image Classification:** Let us first take a look at the loss curves generated using the **CIFAR-10**, **Fashion MNIST**, **MNIST**, and **FER2013** datasets. For now, we will keep the number of epochs before each round of central averaging,  $E$ , equal to 1. It will ensure a proper comparison between the traditional method and our proposed method while varying the number of subsets,  $N$ . 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  $N$ . For example, in the case of **CIFAR-10**, we varied  $N$  in the range of 1 to 9.

As we can see from Figure 6 and Figure 9, the validation loss keeps decreasing with each epoch for longer when using our proposed D2C method on these datasets but starts to increase after very few epochs again in the case of traditional NN implementations using the entire training set. In Figure 7, we can see that D2C with 9 subsets almost entirely eliminates the trend of an increasing loss after a few epochs. 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

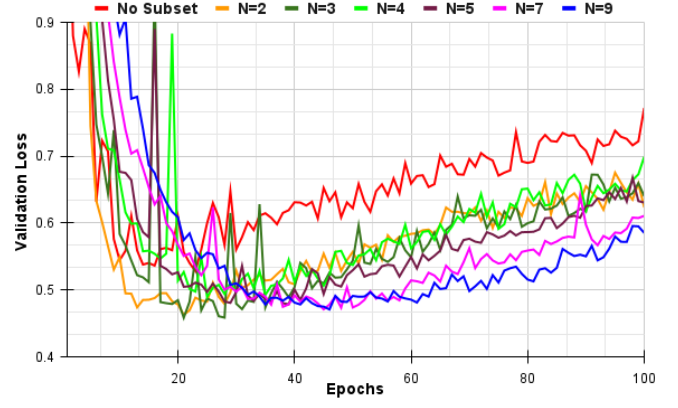


Fig. 6. Validation loss curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on CIFAR-10.

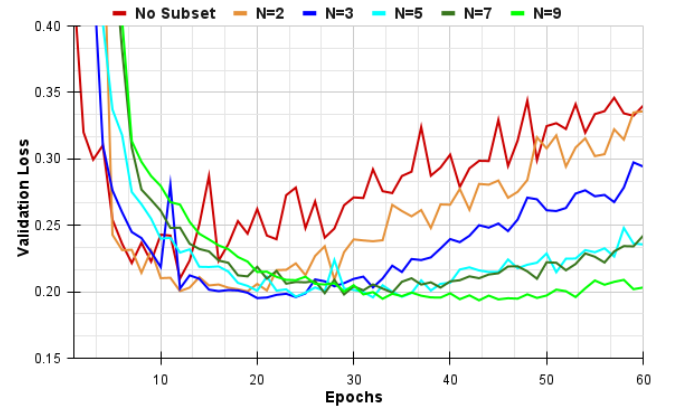


Fig. 7. Validation loss curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on Fashion MNIST.

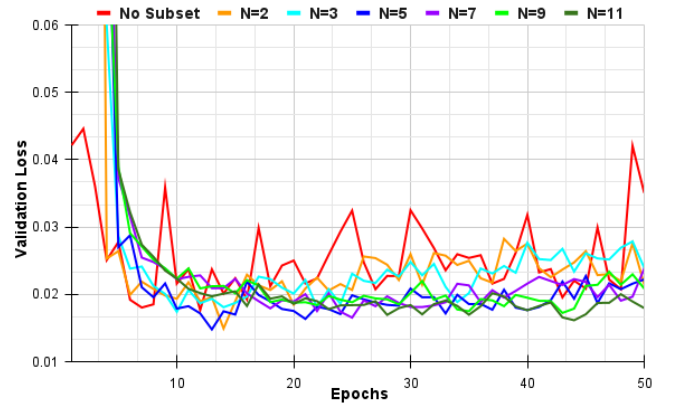


Fig. 8. Validation loss curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on MNIST.

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 can also see that, in all these cases, dividing the training data into more and more subsets delays and slows down the abrupt increase in validation loss as the training progresses. For example, in the case of CIFAR-10, the loss curve generated using the

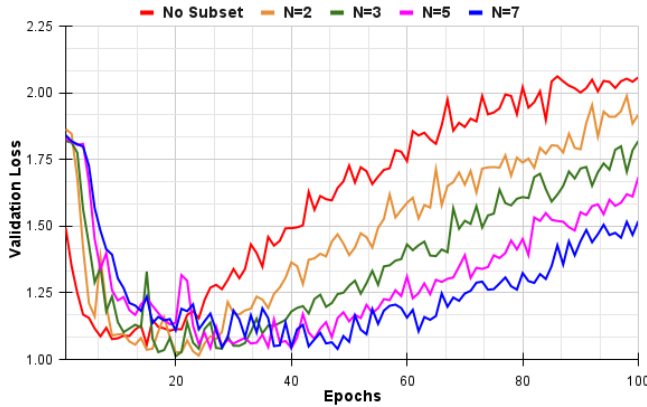


Fig. 9. Validation loss curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on FER2013.

traditional approach without dividing the training data (no subsets) reaches its minimum, 0.536, at around 16 epochs. 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 reaches 0.64 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.

Also, D2C manages to maintain a lower validation loss in general in all cases, including MNIST as we can see from Figure 8. The minimum value of loss that each of the cases discussed above also tells a similar story. The more subsets we divide the training set into, the less the validation loss we can observe from the curves. When a model maintains a lower validation loss during training, it indicates that the model is able to 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, we can conclude that our proposed method manages to bring down the validation loss during training compared to the traditional method, which is a definitive sign of better generalization and reduced overfitting. We can further conclude that this reduction in validation loss becomes more apparent as we increase  $N$ , the number of subsets. The most clear illustration of this phenomenon is seen in the case of FER2013 (Figure 9). The Facial Expression Recognition task using the FER2013 dataset is prone to overfitting to a great extent. Dividing the training set into an increasing number of subsets evidently minimizes that overfitting. So, dividing the training set into more and more subsets helps reduce overfitting, an observation that is consistent with what we saw in Eq. (13). 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 comes the decrease in overfitting. Also, as we saw before, the true error, represented by validation loss in this case, is minimized using D2C, which proves our hypothesis and shows that D2C is indeed effective in treating the problem of overfitting.

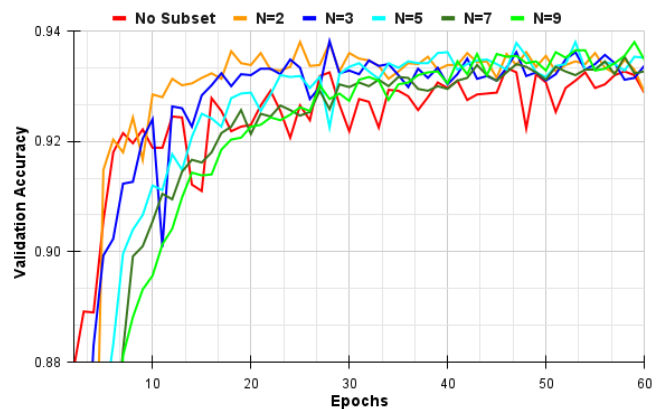


Fig. 10. Validation accuracy curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on Fashion MNIST.

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 10, 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 60th 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. This is something we observed in the cases of all these datasets. Also, we can notice a visible improvement in terms of validation accuracy in Figure 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 with better generalization ability.

There is another interesting aspect of the validation loss curve here. As we can see from Figure 8, 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.

Let us also take a look at the validation loss curves generated using FER2013 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.

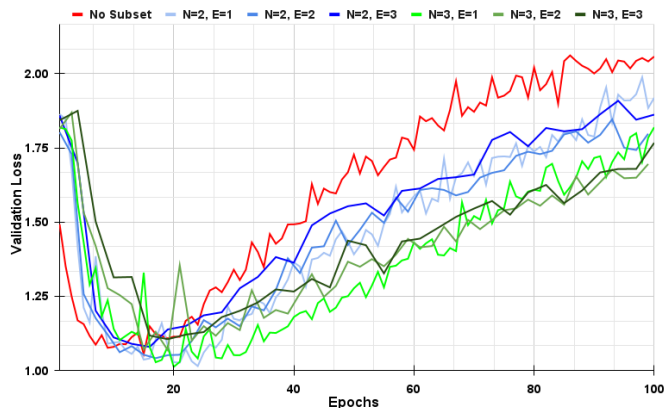


Fig. 11. Validation loss curves for different numbers of epochs before each round of central averaging, while keeping the number of subsets,  $N = 2$  & 3.

We can see from Figure 11 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 slower. But here, for both  $N = 2$  (shades of blue) and  $N = 3$  (shades of green), the trends from the validation curves are quite similar for different values of  $E$  throughout the training phase. 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 subsections 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. We combined the datasets TESS [19], CREMA-D [20], and RAVDESS [21] for this task. Let us take a look at the validation loss and accuracy curves generated using this Combined Audio dataset.

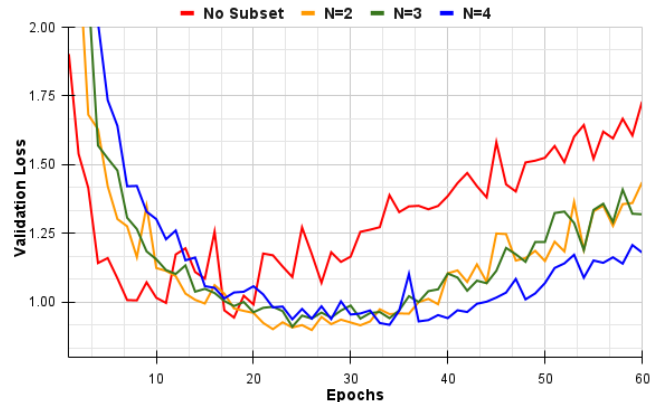


Fig. 12. Validation loss curves for the traditional method & the D2C method for different numbers of subsets,  $N$ , on the Combined Audio dataset.

As we can see from Figure 12, 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, the small size of this dataset may not be ideal in the context of applying D2C. We will analyze it in detail in the next subsection based on several performance metrics.

**Text Classification:** Text classification is an important case study when it comes to testing the Divide2Conquer method. For both image classification and audio classification, 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 section, we will use bi-directional LSTM in this case. We used AG News [22], a benchmark dataset to test our method primarily in the NLP domain. 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 monolithic approach. The loss curve from Figure 13 shows the trend we observed in almost all other cases: dividing the

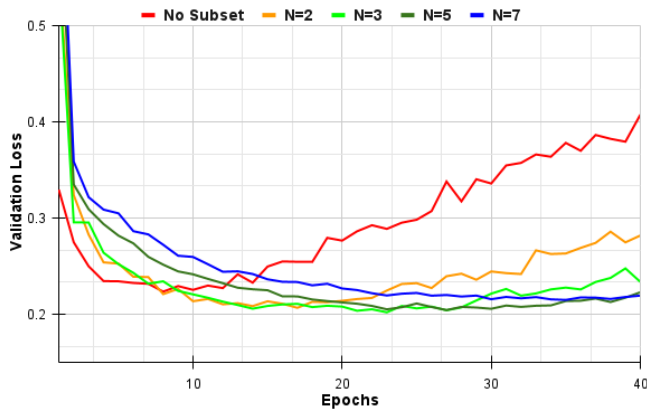


Fig. 13. Validation loss curves for the traditional & the D2C method for different numbers of subsets on AG News.

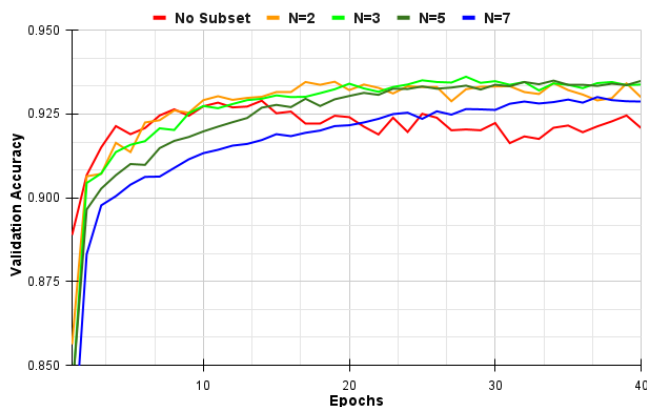


Fig. 14. Validation accuracy curves for the traditional & the D2C method for different numbers of subsets on AG News.

training data into more and more subsets delays and minimizes the increase in validation loss as the training progresses. 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 overfitting and impacts the accuracy too as the training progresses as we can see from Figure 14. However, when we use D2C, 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, almost completely eliminating the increasing trend and showing consistent improvement as the training progresses. The accuracy curve (Figure 14) also shows clear indications that D2C improves over the traditional approach. With the traditional approach, the accuracy too starts deteriorating after around the 14th epoch, indicating severe overfitting, an issue that no longer exists when we use D2C for any value of  $N$  or  $E$ .

The experiments on all these datasets from 3 different datasets prove that using the D2C method can indeed address overfitting and control the abrupt increase of true error in the form of maintaining a lower validation loss during training. This also shows that D2C works well in different domains and a diverse set of tasks.

### C. Accuracy, F1 Score, AUC-ROC, and Log Loss

We evaluated our proposed D2C on test sets using Accuracy, F1 Score, Log Loss, and AUC-ROC to capture a comprehensive view of model performance in terms of correctness, class balance, prediction confidence, and discrimination ability. Reporting these final metrics on test sets was essential to confirm that improvements generalized well beyond the training and validation data, verifying true effectiveness in unseen scenarios. We used the average from multiple repetitive runs (to test for repeatability) for each of these metrics. We used the provided test sets with each of the datasets for proper comparisons. In the tables, the top row represents the traditional centralized 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 benchmark image classification datasets we primarily used using different settings (the traditional approach and the D2C method with different values of  $N$  and  $E$ ).

TABLE II  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING THE D2C METHOD ON CIFAR-10.

Model Settings	Accuracy	F1-Score	Log Loss	AUC-ROC
No Subsets	0.8596	0.8591	0.7346	0.98697
N=2, E=1	0.8611	<b>0.8611</b>	0.7042	0.98693
N=3, E=1	<b>0.8613</b>	0.8609	0.6682	<b>0.98704</b>
N=4, E=1	0.8586	0.8583	0.6516	0.98690
N=5, E=1	0.8552	0.8551	0.6389	0.98628
N=7, E=1	0.8539	0.8535	0.6198	0.98640
N=9, E=1	0.8531	0.8525	<b>0.6068</b>	0.98605

TABLE III  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING THE D2C METHOD ON FASHION MNIST.

Model Settings	Accuracy	F1-Score	Log Loss	AUC-ROC
No Subsets	0.9314	0.9315	0.3467	0.99559
N=2, E=1	0.9323	0.9319	0.3460	0.99549
N=3, E=1	0.9330	0.9329	0.3179	0.99545
N=3, E=2	0.9319	0.9316	0.3303	0.99552
N=3, E=3	0.9337	<b>0.9337</b>	0.3225	0.99590
N=5, E=1	0.9322	0.9321	0.2692	0.99589
N=7, E=1	<b>0.9338</b>	0.9334	0.2467	0.99584
N=7, E=2	0.9305	0.9304	0.2575	<b>0.99600</b>
N=9, E=1	0.9323	0.9323	<b>0.2341</b>	<b>0.99600</b>

As we can see from Table II, Table III, and Table IV, the best accuracy, MCC and AUC-ROC were achieved using the D2C method for all three datasets. For **CIFAR-10**, the number of subsets,  $N$ , was 3 for the best model in terms of accuracy. 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%. Initially, the accuracy improved as we increased the number of subsets, but suffered a consistent drop after it was further increased past 3. As we discussed, dividing the training set into too many subsets may impact training negatively due to a lack



TABLE IV  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING THE D2C METHOD ON MNIST.

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

of representative samples. The experimental outcomes show further evidence of that in this case.

For **Fashion MNIST** too, the best accuracy, MCC, and AUC-ROC were achieved using our proposed Divide2Conquer method. The number of subsets,  $N$ , was 7 in this case for the best model in terms of accuracy. 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  $N = 7$ .

In the case of **MNIST** too, the best accuracy, MCC, and AUC-ROC were achieved using our proposed method. 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. The improvement of 0.07 in error rate over the traditional approach is significant for the **MNIST** dataset. In this case, the least amount of log loss is also achieved with the best model in terms of accuracy.

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 a result of that phenomenon. Even though the best model isn't usually selected based on log loss values, in the case of similar accuracy (for example, the accuracies for  $N = 3, E = 3$  and  $N = 7, E = 1$  in **Fashion MNIST** dataset), the best model can be selected based on log loss. Log Loss evaluates how well the model predicts probabilities, penalizing confident but incorrect predictions, which is critical for understanding model reliability. A lower log loss means a more robust and confident model, so the model with a lower log loss is preferred when the test accuracy or F1 score is similar.

The comparison of subset numbers in the best-performing models across **CIFAR-10**, **Fashion MNIST**, and **MNIST** provides critical insights. For **CIFAR-10**, the optimal model utilized only 3 training subsets, while the best models for **Fashion MNIST** and **MNIST** required 7 and 11 subsets, respectively. Notably, increasing the subset count beyond 3 led to declining accuracy for **CIFAR-10**—a trend not observed with the other datasets. This discrepancy is likely due to

the distinct characteristics of each dataset. While all contain 10 classes, **CIFAR-10**, a smaller dataset with RGB images and complex class representations, presents a more challenging classification task compared to the grayscale images of **Fashion MNIST** and **MNIST**. Consequently, effective training on **CIFAR-10** demands larger sample sizes per subset, and when these subsets become too small, performance degrades. In contrast, **Fashion MNIST** and **MNIST**, identical in dataset size, differ in task complexity, with **Fashion MNIST** presenting a slightly more challenging classification task than **MNIST**. This complexity results in the optimal **Fashion MNIST** model requiring fewer subsets than **MNIST**, as each subset's sample size needs to be sufficient for adequate training.

This analysis suggests a key principle for applying the D2C method effectively: to divide training data into more subsets, datasets must be large enough to allow each subset ample training data. Larger datasets benefit more from divided training, requiring more subsets to achieve optimal performance.

TABLE V  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING THE D2C METHOD ON FER2013.

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

FER2013, a challenging dataset for facial expression recognition, represents a task that is particularly prone to overfitting. As seen in Table V, our proposed D2C method outperformed the traditional approach and achieved the highest accuracy, MCC, and AUC-ROC, with the best-performing model using only 2 subsets ( $N = 2$ ) and 2 training epochs before each central averaging step ( $E = 2$ ). D2C yielded a notable accuracy improvement, reaching 65.67% compared to 64.98% in the traditional model without subsets. Unlike previous datasets, where a single epoch ( $E = 1$ ) was optimal, FER2013 benefited from 2 epochs, likely due to the dataset's complexity, requiring additional training time to capture meaningful features before model averaging.

A decreasing trend in log loss was observed with more subsets, though no consistent pattern emerged with varying epochs ( $E$ ). Additionally, due to FER2013's class imbalance, accuracy alone may not reliably indicate the best model, so, the F1-score, a metric considering class-wise precision, provides a valuable perspective. Interestingly, the optimal F1-score model (3 subsets,  $E = 2$ ) differs from the accuracy-optimized model, with D2C achieving an F1 score of **0.6465** versus **0.642** in the baseline model. This F1-score improvement underscores D2C's effectiveness in addressing overfitting, particularly in class-imbalanced, complex tasks.

**Audio Classification:** Now let us take a look at the performance metrics from the Combined Audio dataset.

TABLE VI  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING D2C ON THE COMBINED AUDIO DATASET.

Model Settings	Accuracy	F1-Score	Log Loss	AUC-ROC
No Subset	<b>0.6953</b>	0.6870	1.3419	<b>0.96570</b>
N=2, E=1	0.6942	0.6879	1.0811	0.96553
N=2, E=2	0.6908	0.6819	1.1636	0.9632
N=3, E=1	0.6920	<b>0.6909</b>	1.2032	0.96416
N=3, E=2	0.6874	0.6830	<b>1.0033</b>	0.96438
N=4, E=1	0.6921	0.6804	1.2840	0.96214

We can see from Table VI that in the **Combined Audio** dataset, the highest accuracy was achieved using the traditional training approach rather than our proposed D2C method. The traditional model attained a 69.53% accuracy, slightly outperforming the D2C approach, which reached 69.42% with 2 subsets. Accuracy declined further as the number of subsets increased, likely due to the smaller dataset size (11,632 samples), limiting effective data availability per subset. As a result, dividing the dataset into smaller portions likely impacted performance by reducing the sample count for each class, a significant challenge given the dataset’s complexity and diversity, as it combines samples from multiple sources.

However, despite marginally lower accuracy, the D2C method showed reduced log loss, indicating a trade-off between accuracy and confidence in predictions. In cases of comparable accuracy, lower log loss can be preferable as it indicates more calibrated predictions. Moreover, given the dataset’s class imbalance, the F1-score is a more reliable metric here. The D2C approach yielded a slight improvement in the F1-score (0.6909 vs. 0.687), indicating better generalization across classes. This suggests that D2C supports robust classification with fewer, more distinctive errors, especially in class-imbalanced, complex tasks like this one. In summary, while D2C did not maximize accuracy, it achieved comparable F1 performance with lower log loss, resulting in a model that generalizes well and offers more confident predictions in challenging, class-imbalanced datasets. However, dividing the dataset into too many subsets does worsen the performance of the models in this case.

This analysis leads us to the conclusion 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.

We can observe from Table VII that, once again, the highest accuracy and F1-score were achieved with 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 VII  
PERFORMANCE METRICS WITH TRADITIONAL APPROACH VS. DIFFERENT SETTINGS USING THE D2C METHOD ON AG NEWS.

Model Settings	Accuracy	F1-Score	Log Loss	AUC-ROC
No Subset	0.9243	0.9243	0.2611	0.98764
N=2, E=1	0.9291	0.9291	0.2556	0.98873
N=3, E=1	0.9288	0.9288	0.2520	0.98872
N=3, E=2	0.9275	0.9275	0.2567	0.98852
N=5, E=1	0.9293	0.9292	0.2412	0.98881
N=5, E=2	<b>0.9295</b>	<b>0.9296</b>	0.2451	0.98875
N=7, E=1	0.9271	0.9271	<b>0.2353</b>	<b>0.98916</b>

Table VII, 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,  $E$ .

The effect of dividing the training set into multiple subsets is the most apparent in this dataset compared to all the previous ones. AG News, the largest dataset in our study, presents an ideal case for D2C, as each class contains 30,000 samples. This abundance of data per class allows for substantial division without sacrificing subset size, which maintains the model’s training stability while leveraging the regularizing effect of partitioning and averaging. Consequently, every variation of D2C (across all values of  $N$  and  $E$ ) demonstrated accuracy gains over the traditional approach, underscoring D2C’s suitability for mitigating overfitting in text classification tasks.

In summary, D2C excels with large datasets like AG News, where ample samples per class allow the model to benefit from both training on robust subsets and reducing overfitting through partitioning and averaging. This validates D2C’s potential for significant impact in Big Data applications, where traditional models often struggle to fully exploit available data.

#### D. Augmentation in D2C

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, 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.

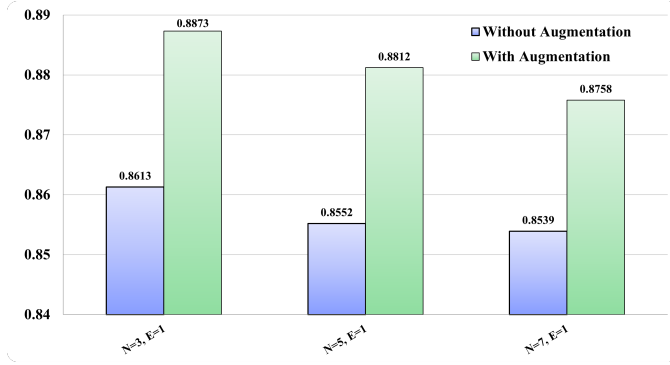


Fig. 15. Test accuracies for different numbers of subsets,  $N$ , using the D2C method on the CIFAR-10 dataset with and without Augmentation.

As we can see from Figure 15, 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.

## VII. LIMITATIONS

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 16 for **MNIST**.

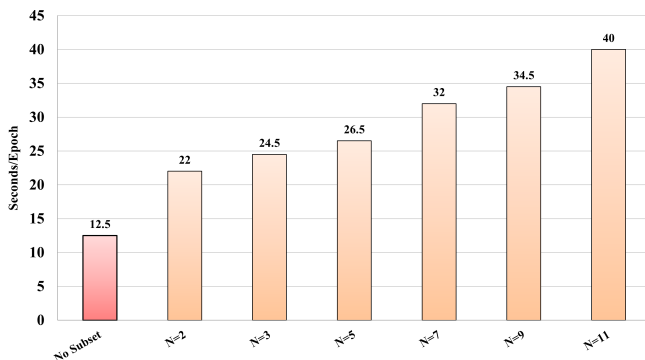


Fig. 16. Time elapsed per epoch for the traditional approach and different numbers of subsets,  $N$ , using the D2C method on the MNIST dataset.

The training was done using a V100 GPU in each of these cases for fair comparisons. As we can see from Figure 16, 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 one 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 one 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.

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.**

## VIII. 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 paper, 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 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.

## ACKNOWLEDGMENTS

This work was supported by the Institute for Advanced Research (IAR), United International University, under Grant UIU-IAR-02-2022-SE-06. The authors sincerely appreciate the funding and resources provided, which were essential to the successful execution of this research.

## APPENDIX A RELATED PROOFS AND DERIVATIONS

Derivation of  $Var(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2$  : Let the random variable  $X$  denote the estimate of  $x$ . The variance

of estimating this random variable is defined as the average deviation of  $X$  from the expected value of our estimate,  $\mathbb{E}(X)$ , where the deviation is squared for symmetry of difference. So, the variance can be formulated as:

$$\begin{aligned} Var(X) &= \mathbb{E}((X - \mathbb{E}(X))^2) \\ &= \mathbb{E}(X^2 + (\mathbb{E}(X))^2 - 2X\mathbb{E}(X)) \\ &\stackrel{(a)}{=} \mathbb{E}(X^2) + (\mathbb{E}(X))^2 - 2\mathbb{E}(X)\mathbb{E}(X) \\ &= \mathbb{E}(X^2) - (\mathbb{E}(X))^2 \end{aligned} \quad (14)$$

Here, (a) is due to the linear nature of the expectation operator, and  $\mathbb{E}(\mathbb{E}(X)) = \mathbb{E}(X)$ , because  $\mathbb{E}(X)$  is not a random variable.

Derivation of  $Cov(XY) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)$  : Let  $X$  and  $Y$  be estimations of two random variables  $x$  and  $y$  respectively. The covariance of  $X$  and  $Y$  is defined as the expected value (or mean) of the product of their deviations from their individual expected values [35]. Hence,

$$\begin{aligned} Cov(X, Y) &= \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y))) \\ &= \mathbb{E}(XY - X\mathbb{E}(Y) - Y\mathbb{E}(X) + \mathbb{E}(X)\mathbb{E}(Y)) \\ &\stackrel{(b)}{=} \mathbb{E}(XY) - \mathbb{E}(X\mathbb{E}(Y)) \\ &\quad - \mathbb{E}(Y\mathbb{E}(X)) + \mathbb{E}(\mathbb{E}(X)\mathbb{E}(Y)) \\ &\stackrel{(c)}{=} \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) \\ &\quad - \mathbb{E}(Y)\mathbb{E}(X) + \mathbb{E}(X)\mathbb{E}(Y) \\ &= \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) \end{aligned} \quad (15)$$

Here, (b) is because expectation is linear and (c) is due to considering,  $\mathbb{E}(cX) = c\mathbb{E}(X)$ . Now if we consider two continuous random variables  $X$  and  $Y$  which are independent, i.e.,  $X \perp Y$ , then by the definition of expectation we have:

$$\begin{aligned} \mathbb{E}(XY) &= \iint xyf(x, y)dxdy \\ &\stackrel{\perp}{=} \iint xyf(x)f(y)dxdy \\ &= \int yf(y) \int xf(x)dxdy \\ &\stackrel{(d)}{=} \int yf(y)\mathbb{E}(x)dy \\ &= \mathbb{E}(X) \int yf(y)dy \\ &= \mathbb{E}(X)\mathbb{E}(Y) \end{aligned} \quad (16)$$

Here, (d) is because  $\mathbb{E}(X) = \int xf(x)dx$  by definition. From 15 and 16 we immediately have  $Cov(X, Y) = 0$  when  $X \perp Y$ .

Derivation of  $Var(\sum_{i=1}^k X_i) = \sum_{i=1}^k Var(X_i) + \sum_{1 \leq i \neq j \leq k} Cov(X_i, X_j)$ :

Let us consider the random variables  $X$  and  $Y$  and arbitrary constants  $a, b$ . Using 14 we can write,

$$\begin{aligned} Var(aX + bY) &= \mathbb{E}((aX + bY)^2) - (\mathbb{E}(aX + bY))^2 \\ &= (a^2 \mathbb{E}(X^2) + b^2 \mathbb{E}(Y^2) + 2ab \mathbb{E}(XY)) \\ &\quad - (a \mathbb{E}(X) + b \mathbb{E}(Y))^2 \\ &= (a^2 \mathbb{E}(X^2) + b^2 \mathbb{E}(Y^2) + 2ab \mathbb{E}(XY)) \\ &\quad - a^2 (\mathbb{E}(X))^2 - b^2 (\mathbb{E}(Y))^2 \\ &\quad - 2ab \mathbb{E}(X)\mathbb{E}(Y) \\ &\stackrel{(e)}{=} a^2 Var(X) + b^2 Var(Y) + 2ab Cov(X, Y) \end{aligned} \quad (17)$$

Here, (e) follows directly from 14 and 15. So Using 16 and 17 and generalizing for  $k$  random variables we can say:

$$Var(\sum_{i=1}^k X_i) = \sum_{i=1}^k Var(X_i) + \sum_{1 \leq i \neq j \leq k} Cov(X_i, X_j) \quad (18)$$

## REFERENCES

- [1] Kim, M., Choi, C. & Pan, S. Ensemble networks for user recognition in various situations based on electrocardiogram. *IEEE Access*. **8** pp. 36527-36535 (2020)
- [2] Konečný, J., McMahan, B. & Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *ArXiv Preprint ArXiv:1511.03575*. (2015)
- [3] Shanmugavel, A., Ellappan, V., Mahendran, A., Subramanian, M., Lakshmanan, R. & Mazzara, M. A novel ensemble based reduced overfitting model with convolutional neural network for traffic sign recognition system. *Electronics*. **12**, 926 (2023)
- [4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal Of Machine Learning Research*. **15**, 1929-1958 (2014)
- [5] Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L. & Batra, D. Reducing overfitting in deep networks by decorrelating representations. *ArXiv Preprint ArXiv:1511.06068*. (2015)
- [6] Zheng, Q., Yang, M., Yang, J., Zhang, Q. & Zhang, X. Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. *IEEE Access*. **6** pp. 15844-15869 (2018)
- [7] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. **86**, 2278-2324 (1998)
- [8] Ioffe, S. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv Preprint ArXiv:1502.03167*. (2015)
- [9] Zeng, Z., Liu, Y., Lu, X., Zhang, Y. & Lu, X. An Ensemble Framework Based on Fine Multi-Window Feature Engineering and Overfitting Prevention for Transportation Mode Recognition. *Adjunct Proceedings Of The 2023 ACM International Joint Conference On Pervasive And Ubiquitous Computing & The 2023 ACM International Symposium On Wearable Computing*. pp. 563-568 (2023)
- [10] Ying, X. An overview of overfitting and its solutions. *Journal Of Physics: Conference Series*. **1168** pp. 022022 (2019)
- [11] Rusia, M. & Singh, D. An efficient CNN approach for facial expression recognition with some measures of overfitting. *International Journal Of Information Technology*. **13**, 2419-2430 (2021)
- [12] Xiao, H. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *ArXiv Preprint ArXiv:1708.07747*. (2017)
- [13] Korjus, K., Hebart, M. & Vicente, R. An efficient data partitioning to improve classification performance while keeping parameters interpretable. *PloS One*. **11**, e0161788 (2016)
- [14] Krizhevsky, A., Nair, V., Hinton, G. & Others The CIFAR-10 dataset. *Online: Http://www. Cs. Toronto. Edu/kriz/cifar. Html*. **55**, 2 (2014)
- [15] Kolluri, J., Kotte, V., Phridviraj, M. & Razia, S. Reducing overfitting problem in machine learning using novel L1/4 regularization method. *2020 4th International Conference On Trends In Electronics And Informatics (ICOEI)(48184)*. pp. 934-938 (2020)



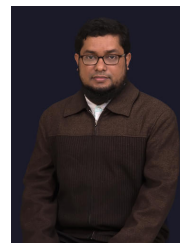
- [16] Zahara, L., Musa, P., Wibowo, E., Karim, I. & Musa, S. The facial emotion recognition (FER-2013) dataset for prediction system of micro-expressions face using the convolutional neural network (CNN) algorithm based Raspberry Pi. *2020 Fifth International Conference On Informatics And Computing (ICIC)*. pp. 1-9 (2020)
- [17] Siddiqui, M., Shusmita, S., Sabreen, S. & Alam, M. FedNet: Federated Implementation of Neural Networks for Facial Expression Recognition. *2022 International Conference On Decision Aid Sciences And Applications (DASA)*. pp. 82-87 (2022)
- [18] Li, X., Chen, S., Hu, X. & Yang, J. Understanding the disharmony between dropout and batch normalization by variance shift. *Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition*. pp. 2682-2690 (2019)
- [19] Dupuis, K. & Pichora-Fuller, M. Toronto emotional speech set (tess)-younger talker\_happy. (Toronto: University of Toronto, Psychology Department, 2010,2010)
- [20] Cao, H., Cooper, D., Keutmann, M., Gur, R., Nenkova, A. & Verma, R. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE Transactions On Affective Computing*. **5**, 377-390 (2014)
- [21] Livingstone, S. & Russo, F. 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*. **13**, e0196391 (2018)
- [22] Zhang, X., Zhao, J. & LeCun, Y. Character-level convolutional networks for text classification. *Advances In Neural Information Processing Systems*. **28** (2015)
- [23] Ghojogh, B. & Crowley, M. The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. *ArXiv Preprint ArXiv:1905.12787*. (2019)
- [24] Ghojogh, B., Nekoei, H., Ghojogh, A., Karray, F. & Crowley, M. Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review. *ArXiv Preprint ArXiv:2011.00901*. (2020)
- [25] Breiman, L. Bagging predictors. *Machine Learning*. **24** pp. 123-140 (1996)
- [26] Bühlmann, P. & Yu, B. Analyzing bagging. *The Annals Of Statistics*. **30**, 927-961 (2002)
- [27] Morgan, N. & Bourlard, H. Generalization and parameter estimation in feedforward nets: Some experiments. *Advances In Neural Information Processing Systems*. **2** (1989)
- [28] Prechelt, L. Early stopping-but when?. *Neural Networks: Tricks Of The Trade*. pp. 55-69 (2002)
- [29] Fürnkranz, J. Pruning algorithms for rule learning. *Machine Learning*. **27** pp. 139-172 (1997)
- [30] Bramer, M. Using J-pruning to reduce overfitting in classification trees. *Knowledge-Based Systems*. **15**, 301-308 (2002)
- [31] Yip, K. & Gerstein, M. Training set expansion: an approach to improving the reconstruction of biological networks from limited and uneven reliable interactions. *Bioinformatics*. **25**, 243-250 (2009)
- [32] Sun, Y., Wang, X. & Tang, X. Deep learning face representation from predicting 10,000 classes. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 1891-1898 (2014)
- [33] Wen, G., Hou, Z., Li, H., Li, D., Jiang, L. & Xun, E. Ensemble of deep neural networks with probability-based fusion for facial expression recognition. *Cognitive Computation*. **9**, 597-610 (2017)
- [34] Zhang, Xu, Yinchuan Li, Wenpeng Li, Kaiyang Guo, and Yunfeng Shao. "Personalized federated learning via variational Bayesian inference." In *International Conference on Machine Learning*, pp. 26293-26310. PMLR, 2022.
- [35] Park, K., Park, M. & Others Fundamentals of probability and stochastic processes with applications to communications. (Springer,2018)



**Md. Saiful Bari Siddiqui** (Graduate Student Member, IEEE) received the B.Sc. degree in Electrical and Electronic Engineering from the Bangladesh University of Engineering and Technology in 2021 and the M.Sc. degree in Computer Science and Engineering from Brac University, Dhaka, Bangladesh, in 2024. He is currently a Lecturer in the Department of Computer Science and Engineering at Brac University, where he is involved in teaching, mentoring and research. He has previously served as a Research Assistant at the Centre for Cognitive and Data Sciences (CCDS), Independent University, Bangladesh, contributing to interdisciplinary projects at the intersection of cognitive science and data analytics. His research interests include machine learning, deep learning, overfitting remediation strategies, and applications of artificial intelligence in medical imaging, industrial systems, and real-world datasets. His work emphasizes decentralized training frameworks for scalable and robust AI solutions. He has also been involved in collaborative research projects supported by organizations like the Institute for Advanced Research (IAR), United International University.



**Md. Mohaiminul Islam** (Graduate Student Member, IEEE) received the B.Sc. degree in Computer Science and Engineering from the Bangladesh University of Engineering and Technology (BUET) in 2021. He is currently a Lecturer in the Department of Computer Science and Engineering at United International University (UIU), Dhaka, Bangladesh, where he is involved in teaching, mentoring and research. His research interests include machine learning applications, federated learning, cybersecurity and privacy, trustworthy machine learning, and distributed systems. His work emphasizes developing secure and reliable AI frameworks for distributed environments. He has published research in renowned conferences and is actively involved in collaborative studies exploring the intersection of AI and diverse real-world challenges.



**Md. Golam Rabiul Alam** (Member, IEEE) received the B.S. degree in computer science and engineering and the M.S. degree in information technology, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2017. He was a Postdoctoral Researcher with the Computer Science and Engineering Department, Kyung Hee University, from March 2017 to February 2018. He is currently a Full Professor of computer science and engineering with the Department of Computer Science and Engineering, BRAC University, Bangladesh. His research interests include healthcare informatics, mobile cloud and edge computing, ambient intelligence, and persuasive technology. He is a member of the IEEE IES, CES, CS, SPS, CIS, KIIE, and IEEE ComSoc. He received several best paper awards at prestigious conferences.