The Pennsylvania State University
The Graduate School

EXTREME POINT APPROACH TO SUBSET SELECTION AND DATA AUGMENTATION

A Thesis in
Electrical Engineering
by
Srikanth Banagere Manjunatha

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2021

The thesis of Srikanth Banagere Manjunatha was reviewed and approved by the following:

Viveck Ramesh Cadambe
Associate Professor, Electrical Engineering
Thesis Advisor

Mehrdad Mahdavi
Assistant Professor, Computer Science and Engineering

Kultegin Aydin
Head of the Department and Professor, Electrical Engineering

Abstract

In the recent years, we have witnessed a drastic increase in the dataset size across various disciplines. It is challenging to store and process this large data. Along with the lack of resources, the existing algorithms are proven to be computationally very expensive as this large data is stored on multiple clusters of machines. Although there are many new algorithms developed focusing on tweaking the architecture of deep neural networks to address the problem of massive data explosion, we focus on a more general problem. We develop a method to obtain a representative subset of the training data, such that the convex hulls of various classes are approximately preserved. We mainly focus on classification problems in machine learning. We first study linear classification models, and then extend our ideas to non-linear classifiers. We develop a randomized extreme point algorithm that approximates the extreme points of a given set of points in high dimensions. We show that the performance of the model on the subsets found using this randomized algorithm are competitive with the performance found on the full dataset. Specifically, for a set of N data points in R^d, our algorithm has a computational complexity of $O(MN^2)$ *independent of the dimension d.*

We extend our approach to develop efficient methods for data augmentation part by adding random noise. Data augmentation has proven to make the margin of classification thick, hence making the model more robust. Current methods augment the entire dataset randomly in an isotropic manner. We explore the domain with a focus on augmenting only the extreme points. Specifically, we portray methods to augment the extreme points in specific direction, such that the convex hull of the augmented points along with the extreme points contains the whole dataset. The specific directions are carefully chosen such that the augmented points do not lie within the convex hull of extreme points. Towards this, we demonstrate a further reduction in size while still achieving a similar performance as the solution found on the full dataset. We further extend our ideas to non linear separable dataset, and demonstrate the interpretability property of the chosen extreme points. We demonstrate the effectiveness of our approach for both training and data augmentation on the standard MNIST data set.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{T}$	$\epsilon-$isometric transformation
S_{aug}	Augmented set of S
S_i	A subset of S belonging to a class i
α	hyper parameter that controls the variance of noise
E	A finite set of extreme points returned by our randomized extreme point enumeration algorithm

Acknowledgments

I would first like to thank my thesis advisor Dr. Viveck Ramesh Cadambe, from the Department of Electrical Engineering at Pennsylvania State University. His constant encouragement and guidance was vital in making this research work and thesis a reality. Every meeting we had, motivated and encouraged me to pursue research. Discussing new topics and ideas with him in regard to research and my coursework was very insightful and has undeniably improved my thesis. He consistently provided me the guidance to ensure that I was making positive progress in my research.

I would like to extend my deepest gratitude to my committee member Dr. Mehrdad Mahdavi, from the Department of Computer Science at Pennsylvania State University. Discussions with him about the extensions and future works of my research work was very helpful and provided the right direction for our future work.

I would like to extend my sincere thanks to Dr. Bill Kay, who is a collaborator from Oakridge National Laboratory, Tennessee, USA. His ideas and insights helped in shaping up this research. I would also like to thank all the non-teaching staff of Pennsylvania State University, especially Sherry Dawn Jackson. The door to her office was always open whenever I had any doubts regarding degree requirements and orientation.

Last but not the least, I would like to thank my parents and sister for their selfless and constant encouragement throughout my years of study and research. I am forever indebted to my parents and sister for encouraging me to explore new directions in life and seek my own destiny. I would also like to thank my close friend Kalpana N, who lent moral and emotional support during my Master's study. Special thanks to all my friends at Pennsylvania State University, Aakarshitha, Aishwarya, Kruthika, Srujan and Balachandran for the wonderful times we shared.

Chapter 1
Introduction

A principal problem in modern machine learning applications is *scalability*. Due to the available large data set, the learning models require longer time to saturate, often hours to days. A popular example is of FaceNet, by Schroff et al. [1] where they claim in their paper that the model takes around 2,000 hours to saturate. With the increase in the size of the data set, there is a need for efficient representation of data for easier processing. Following the trend, there is a higher possibility of seeing a boom in the data, and it is practically very hard to train machine learning models on large chunk of data. Many a times, large part of data sets are often redundant and contribute very less to the margin of classification. These collection of redundant points add an extra cost while having very less significance on the model parameters. This motivates the need to study the data distribution and find an effective representation of data.

1.1 Representative subset

Machine learning models are data-driven models, and understanding the data distribution and representation is very important to build good learning models. The data that is presented for the machine learning model to learn from, is called the training data. In the recent years, application of data-driven models is gaining lot of popularity across various disciplines and successfully solving various tasks. Naturally, the inflow of the training data is increasing with a great extent. However, the growing data set poses several challenges.

The cost and storage requirement to maintain the growing large data is huge and tedious. Furthermore, the processing of this large data can be computationally intensive, as well as time consuming. There is a need to develop new algorithms, each time there is an increase in the data, as the existing algorithms become computationally infeasible. To

tackle these challenges, there is a need to develop new algorithms which can effectively find a representation of the data set as well as reduce the size of the training data. In this research work, our prime focus lies in finding a core set that has two main properties:

1. reduce the size of training data

2. effectively preserves the information contained in them

It is very important, especially in the recent years to focus on understanding the data better to build effective models. The recent works on promoting the robustness of the classifier by increasing the margin of the classification, by Rajput et al. [2] Charles et al. [3] and Yang et al. [4] inspired our research work. In our research work, we mainly build our theories and algorithms assuming a linearly separable dataset. We claim that the solution built and designed using the representative subset is equivalent to the solution obtained on the overall data set. We provide necessary proofs to support our claim. Charles et al. [3] explores Adversarial training technique on linearly separable data and focus on providing the bound on the number of iterations required for a large margin of classification by emphasising the relation between the large margin and robustness. Yang et al. [4] emphasize on thicker boundaries while claiming and showing that thin decision boundaries lead to overfitting. Our representative subset focuses on preserving the convex hull of the data set to maintain the performance.

In this research work, we examine the various properties that govern our approach (refer section 2.2). We progress by building our algorithm with a focus of finding a subset which preserves the convex hulls: the extreme points. We demonstrate the influence of extreme points on the margin of classification (refer section 2.2). We build on the classical algorithmic ideas of discrete geometry and extend it to high dimensional data set. We mainly build upon the works of Ottomann et al. [5] to obtain our subset for effective data representation.

We modify the algorithm presented by Ottomann et al. [5] for applications on high dimensional data. We also provide supporting experiments along with proofs on standard MNIST data set, a real world data set (which is almost linearly separable). We show that our algorithm compresses the MNIST data set by over 50% in a few minutes while the same model trained on the whole data takes hours to saturate. The empirical results also show that model trained over the chosen subset (from our algorithm), coupled with data augmentation has similar accuracy as the models trained on all the data.

This constitutes the first phase of our research work, where we focus on developing and obtaining the representative subsets that preserve the effective information of the

whole training data. We evaluate the effectiveness of our approach using a training method, where we train a linear classifier on standard MNIST data and compare the performance.

1.2 Data Augmentation

Data augmentation, as the name suggests, are ways of augmenting or increasing the data points, such that the augmented data points enable higher performance of the learning models. The augmented data samples have proven to increase the robustness of machine learning models. Data augmentation techniques, in a broader sense, modify or/and add noise to the original data points to produce the augmented samples. These augmented samples are in turn used in training procedure coupled with the original training samples. In modern machine learning applications, data augmentation has proven to be very effective in improving the model's performance. However, the downside to this approach is that the technique essentially increases the overall data set by several folds, based on the number of different augmentations the user decides to implement.

The works of Rajput et al. [2] discusses the various common data augmentation techniques that add noise to the samples, indeed increases the margin of classification and hence making the model more robust. It is noteworthy that this work does suffer from the trade off between performance and the largely increasing data size (due to augmentation techniques). There are several conventional augmentation techniques, namely, Gaussian blurring, translation, flipping, rotation, salt and pepper noise, etc. Customarily, these augmentation techniques are implemented with retaining of the original training data, which results in drastic escalation in the training data set size. As described in the earlier section (refer section 1.1), the growing data size has many challenges and can increase the cost to a large extent. Hence, a naive data augmentation implementation can prove adverse.

As we look carefully at the analysis of Rajput et al. [2], it indicates that the augmented samples near the classification boundary is sufficient to increase the margin of classification. Inspired by this idea, we focus on augmentation techniques that add noise to the coresets that are closer to the margins of classification. In this research work, we develop novel data augmentation techniques that perturb selected data points only in the selected directions. Building upon the works where the extreme points influence the classification boundary, we provide general notion, and supporting proofs to claim that the convex hull of the augmented points necessarily contains the convex hull of the original training

data (refer section 3.3).

Furthermore, we demonstrate the performance of our novel data augmentation technique on the standard MNIST data set and compare the performance with conventional data augmentation techniques. In general, we train a linear classifier employing the two techniques and provide a comparison, claiming that our data augmentation techniques outperform the conventional techniques with an additional reduction in the data set size. We show a further reduction in size by 26% while maintaining a same classification accuracy.

This constitutes the second phase of our research work, where we focus mainly on developing new augmentation techniques which augments in selective directions, effectively maintaining or improving the model's performance. We evaluate our approach by training a linear classifier on standard MNIST data coupled with conventional augmentations against our augmentation technique.

1.3 Related Works

There is a rich history of novel techniques involving data representation and coreset selection in machine learning. Specifically, the survey by Bachem et al., 2017 [6] gives a detail understanding of what coresets are, how to construct them and their implementations and applications on various machine learning algorithms, like clustering and estimation theory. They discuss the effect of large dataset on the existing algorithms, and how the data has to be stored on various clusters of machines. They aim to build coresets such that any solution obtained using the coresets is provably competitive to the solutions obtained on full dataset. They further explore the application and performance of coresets on various clustering algorithms, estimation in mixture models. Similar survey can be found by Jubran et al., 2019 [7], where they give a detailed description of all the related works about coresets with detailed proofs and simple results.

A detailed study about clustering using coresets can be found by Har-Peled & Mazumdar 2004 [8], and a recent study by Huang & Vishnoi in 2020 [9]. Har-Peled & Mazumdar [8] discuss the existence of coresets for problem of clustering in low dimensions. They examine the use of weighted coresets to obtain the clusters that are an approximate by a factor of $(1+\epsilon)$. They develop a faster algorithm with linear time complexity to compute the approximate k-means and k-medians. Clustering algorithms are further explored in detail by Huang & Vishnoi [9], where they aim to solve a more general clustering problem - the (k, z) – clustering problem of finding a subset of k "centers"

which minimizes a function of Euclidean distance of each point to the closest center.

In the course of this research, we present our results for linearly separable dataset and then present ways to extend it to non-linear case. The survey on Bachem et al. 2017 [6](discussed above), Munteanu et al. 2018 [10], Raj et al. 2020 [11] and Mirzasoleiman et al. 2020 [12] best discusses the implementation of linear classifiers with subsamples of the data, and aims to discuss ways to make the training more efficient without losing the performance. Munteanu et al. 2018 [10] examine the importance of coresets for large data analysis and theorize that no sublinear coresets exist for logistic regression. They present a comparative study of uniform sampling and state of the art methods for logistic regression with their novel sensitivity sampling scheme to produce sublinear approximate coresets. A recent study by Raj et al. 2020 [11] explore the approximation of the loss function by the subsamples. They develop a novel sensitivity score indicating the importance of each subsampled data point and its contribution to the total loss. Another recent study by Mirzasoleiman et al. 2020 [12] describes ways to efficiently choose the weighted subset which closely estimates the full gradient by posing it as a maximization problem.

The focus of our research is more aligned with the works of Rajput et al., 2019 [2], Dao et al., 2019 [13] and Kuchnik & Smith, 2018 [14]. Rajput et al. 2019 [2] discuss the importance of augmenting the data samples that are close to the boundaries. They claim that the margin improves in such a case, making the model more robust. Dao et al. 2019 [13], present the understanding of data augmentation as a Markov process and their effects on kernel classifiers. A very close reference to our research focus is the study performed by Kuchnik & Smith, 2018 [14]. They emphasize on the inefficiency of naively augmenting all the data points and how it can lead to an explosion of data which can result in high storage and training costs. They present a study based on loss that effectively chooses the subset which maintains the performance of the model. Our approach, in contrast to the works of Kuchnik & Smith [14], does not require training the model for linearly separable dataset. Our algorithm performs it as a pre-processing stage when the data is linearly separable, and hence faster and more efficient.

Chapter 2
Subset Selection

In this material. we present supporting theorems and proofs to justify our algorithms. to ease the process of fair understanding, let us familiarize ourselves with some basic notations.

For two non-empty sets $S, T \subset \mathbb{R}^D$, we denote

$$d(X, Y) = \inf_{\mathbf{x} \in X, \mathbf{y} \in Y} ||\mathbf{x} - \mathbf{y}||_2$$

If the set S is a singleton $\{\mathbf{x}\}$, we simply denote

$$d(\mathbf{x}, T) = d(\{\mathbf{x}\}, T)$$

. This represents the closest distance between two sets.

For set S, $\texttt{diam}(S)$ represents its diameter, which is the distance between the farthest two points within the set i.e.,

$$\texttt{diam}(S) = \sup_{\mathbf{x}, \mathbf{y} \in S} ||\mathbf{x} - \mathbf{y}||$$

For a set $A \in \mathbb{R}^D$, $int(A)$ denotes its set of interior points. For a finite set $S \subset \mathbb{R}^D$, the convex hull of the set S is denoted by $\texttt{conv}(S)$.

The *extreme points* of a finite non-empty set S is the smallest subset $T \subseteq S$ such that

$$\texttt{conv}(S) = \texttt{conv}(T)$$

The extreme points T is a unique subset of S such that $\texttt{conv}(S) = \texttt{conv}(T)$, and $t \in T \Rightarrow t \notin \texttt{conv}(T \backslash \{t\})$. This emphasizes the improtance of each data point in the extreme subset T. The notation $\mathcal{N}(\mu_{n \times 1}, \mathbf{K}_{n \times n})$ represents the vector Gaussian

distribution with mean vector μ and covariance matrix $\mathbf{K}$.

2.1 Setup

We consider a labeled data set $\mathcal{D}$ that has C classes, denoted as $\mathcal{C} := \{0, 1, \ldots, C-1\}$. The data set is represented as $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\} \subset \mathbb{R}^D \times \mathcal{C}$ denotes N training samples of D dimensional vectors mapped to a class in $\mathcal{C}$, i.e., $\mathbf{x}_i \in \mathbb{R}^D, i = 1, 2, \ldots, N$ and the labels $y_i \in \mathcal{C}, i = 1, 2, \ldots, N$.

We assume that every point has a unique label in the data set, that is

$$(\mathbf{x}, y), (\mathbf{x}, y') \in \mathcal{D} \Rightarrow y = y'.$$

In general, a subset selection algorithm `SubsetAlg` takes the data set $\mathcal{D}$ as input and outputs a subset $\mathcal{S} \subseteq \mathcal{D}$ where $n \leq N$ samples.

A classifier is a function $g : \mathbb{R}^D \to \{0, 1, 2, \ldots, C-1\}$, which maps the D-dimensional vectors to a class (one among $\mathcal{C}$). Classifier g is presented with the available data $\mathcal{D}$ as a training procedure for it to learn the data distribution. The training procedure generally involves presenting the whole training data in batches in multiple epochs. We mainly focus and develop our algorithms on the hypothesis space of linear classifiers. A linear classifier is a special case of g, where it consists of hyperplanes

$$\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{C-1} \in \mathbb{R}^{D+1}$$

and the classification function reduces to:

$$g(\mathbf{x}) = \arg \max_{i \in \{0,1,\ldots,C-1\}} \mathbf{w}_i \begin{bmatrix} 1 \\ \mathbf{x}^T \end{bmatrix}$$

where w_0 usually correspond to the bias unit.

A training data set $\mathcal{D}$ is *linearly separable* if and only if there exists a linear classifier g such that

$$(\mathbf{x}, y) \in \mathcal{D} \Rightarrow g(\mathbf{x}) = y.$$

Any classifier g that satisfies the above equation *linearly separates* the data set.

Let us consider a set $\mathcal{D}_\mathbf{x}$ such that, $\{\mathbf{z} \in \mathcal{D} : g(\mathbf{z}) \neq g(\mathbf{x})\}$. For a linear classifier g

that linearly separates data set S, the *margin* of the classifier is defined as follows:

$$\texttt{Margin}(g, \mathcal{D}) = \inf_{\mathbf{x} \in \mathcal{D}} d(\mathbf{x}, \mathcal{D}_{\mathbf{x}})$$

A linear classifier with the maximum possible margin is called a *maximum margin classifier*.

2.2 Our approach to Subset Selection

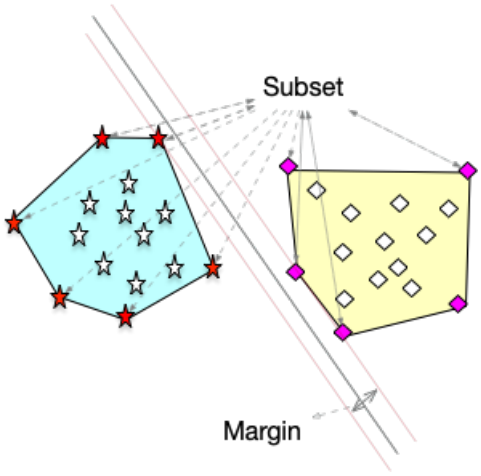

Figure 2.1: Subset Selection via extreme points.

The main aim of our research is to find a representative subset $\mathcal{S}$ of the whole dataset $\mathcal{D}$. A linear classifier g trained on this chosen ideal subset $\mathcal{S}$ should be an approximation of a linear classifier which is trained on the whole dataset. We claim the approximation by the performance of the linear classifier on the common unseen test data. For all our classification experiments, we employ the metric: *accuracy*.

Let $\overline{\mathcal{D}}_i$ be a set of all the D-dimensional vectors which belong to class i. We represent it as follows:

$$\overline{\mathcal{D}}_i = \{\mathbf{x} : (\mathbf{x}, i) \in \mathcal{D}\}$$

for $i = 0, 1, 2, \ldots, C-1$.

For a linear classifier, the membership to any class can be represented as an intersection of a set of hyperplanes in $\mathbb{R}^D$. Let us now examine the conditions when a linear classifier separates the linearly separable dataset $\mathcal{D}$. A linear classifier linearly separates the the linearly separable data set $\mathcal{D}$, if and only if the convex hull of the points in $\overline{\mathcal{D}}_c$ for

each class c, lies in the intersection of the hyperplanes corresponding to class c. Hence, we represent *each class by its subset of extreme points (See Fig. 2.1)*, thus, compactly preserving the convex hull of each class $\overline{\mathcal{D}}_c, c = 0, 1, 2, \ldots, C-1$.

Let $\mathcal{E}_i \subset \overline{\mathcal{D}}_i$ be the set of extreme points of $\overline{\mathcal{D}}_i$. The subset selection algorithm collects all the subsets of the extreme points and returns the union. Let us represent the set of extreme points as $\mathcal{S}_i$. The union set our algorithm returns is as follows:

$$\mathcal{S}_i = \{(\mathbf{x}, i) : \mathbf{x} \in \mathcal{E}_i\}, i = 0, 1, \ldots, C-1.$$

We have three important properties governing our approach. These properties, in general describe the characteristics of the linear classifier and the linear decision boundary, the margin of classification, and the property of our method which has an edge over the works of Kuchnik & Smith, 2018 [14]. The three properties are as follows:

Property 1: Let us start with our linear classification training algorithm `Train( )` that takes in the data set to be trained on. If the input training set is linearly separable, then the classifier `Train`($\mathcal{D}$) linearly separates the dataset after the classifier is fully trained. In that case, `Train`($\mathcal{D}$) linearly separates the input training set $\mathcal{D}$) if and only if the `Train`($\mathcal{S}$) also separates $\mathcal{D}$. We need to note that, `Train`($\mathcal{S}$) is a classifier trained on our chosen set of extreme points $\mathcal{S}$. Figure 2.2 shows a visual representation of Property 1 (property of linear separability) on $2D$ plane. We can extend a similar analogy to high dimensional space as well.

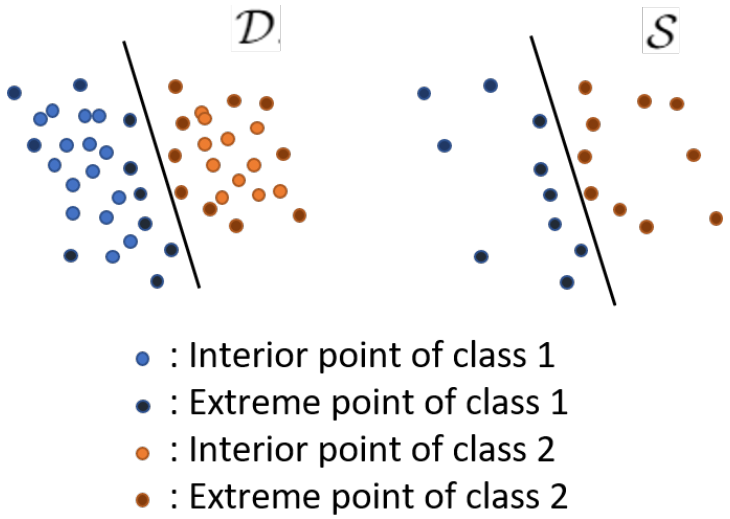

Figure 2.2: A visual representation of Property 1

Property 2: The second property deals with the margin of classification. For any

linear classifier g that separates the linearly separate data set $\mathcal{D}$, we have

$$\texttt{Margin}(g, \mathcal{D}) = \texttt{Margin}(g, \mathcal{S}).$$

This property holds good for the special case too, where `Train( )` generates a maximum margin classifier for a linearly separable data set, $\texttt{Train}(\mathcal{S})$ is also a maximum margin classifier on $\mathcal{D}$. Figure 2.3 shows a visual representation of Property 2 (property of margins) on $2D$ plane. We can extend a similar analogy to high dimensional space as well.

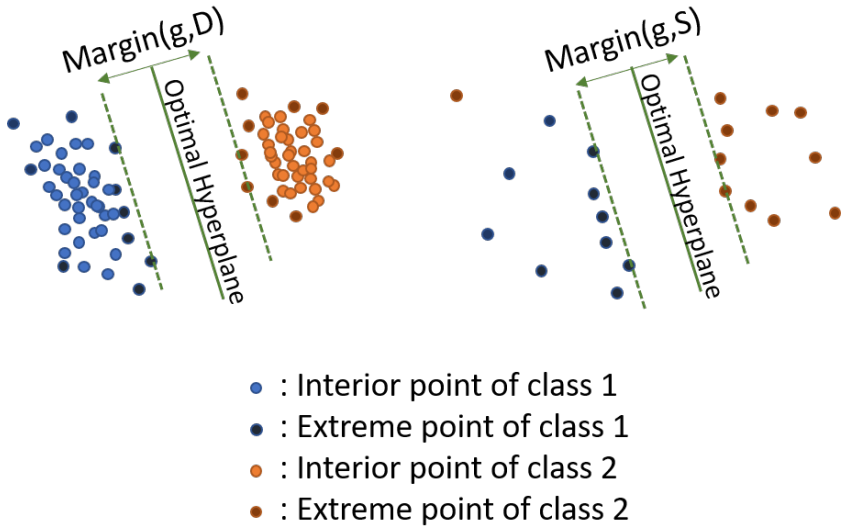

Figure 2.3: A visual representation of Property 2

Property 3: Unlike the works of Kuchnik & Smith, 2018 [14], the extreme point subet selection can be performed as a pre-processing stage and can be done without the use of `Train( )`. This makes our approach faster and more computationally efficient, and hence making our approach better than the reference. For example, as we describe in the coming chapter *Experiments*, we are able to approximate the extreme points for the MNIST data set in a few minutes on a laptop (where training takes hours on the same machine).

Our technical contributions are precise descriptions of Properties $1, 2$ and 3. Properties $1, 2$ are characterized in the next chapter. A key contribution is enabling Property 3 for high dimensional data sets via the development of randomized algorithms for approximately enumerating the extreme points. The extreme point enumeration algorithms are described in the next section. In the coming chapters, we describe the approximation

guarantees obtained by our algorithm with the help of *JL-Lemma*. We describe *JL-Lemma* in detail in the coming sections. In the next section, let us explore the Extrme Point Enumeration algorithm which is developed and built upon the works of Ottmann et al., 1995 [5].

2.3 Algorithms for extreme point enumeration

We reproduce the works of Ottmann et al. [5] for extreme point enumeration. The detailed description of the algorithm can be seen in Algorithm 1. Similar works on extreme point enumeration can be seen in Chan et al. [15] and Clarkson et al. [16]. A key point to remember in the implementation of the algorithm is that the set that we input to the algorithm contains the origin as an interior point.

The algorithm takes a finite set $S \in \mathbb{R}^D$ as input, whose convex hull contains the origin and outputs the set of extreme points of S, which we denote by E. The algorithm uses the following linear program as a subroutine:

$$f_{\mathbf{x}}(Q) = \max_{\mathbf{y} \in \mathbb{R}^D} \mathbf{x}\mathbf{y}^T$$

$$s.t., \mathbf{z}\mathbf{y}^T \leq 1, \mathbf{z} \in Q$$

where Q is an arbitrary subset of S. The above linear program has $|Q|$ constraints, effectively bounding the feasible region to the intersection of a set of halfspaces, where each halfspace corresponds to one point in Q. We need to note a key property of the above linear program: for a set Q that contains a vector $\mathbf{x}$, if

$$f_{\mathbf{x}}(Q) = f_{\mathbf{x}}(\{\mathbf{x}\}),$$

then $\mathbf{x}$ is an extreme point of Q (line 6 in algorithm).

If M is total number of extreme points of S, then the extreme point enumeration algorithm runs for $M|S|$ instances. In Algorithm 1, it is shown in Ottmann et al. [5] that the linear program in line 5 runs for at most $|S|$ times. As the total number of extreme points of S is M, the $|Q|$ has at most M constrains. Hence, the linear program has complexity $O(M)$ due to M constrains. Hence, the overall complexity of executing line 5 is $O(MN)$,. In finding this complexity, we are ignoring the dependence on dimension D.

Ottmann et al. [5] also shows that the line 7 in Algorithm 1 runs for at most M times. This is due to the fact that there are at most M constrains. Corresponding to

one constraint for every element of S, we have the complexity of the linear program of at most $O(|S|)$. Hence, the overall complexity of the line 7 in Algorithm 1 has complexity of at most $O(M|S|)$. Even in the above complexity analysis, we are ignoring the dependence on dimension D. When the number of extreme points are small, the extreme point enumeration algorithm scales well with the size of the data set S.

Two key points to keep in mind during implementation of Algorithm 1 is that the algorithm requires the origin to be an interior point in the convex hull of the finite set S, and all the points in the finite set S are in general position [1]. It is proved in [5] that the Algorithm 1 finds the extreme points of the finite input set S if this setting is taken care.

However, we observe that Algorithm 1 has two major drawbacks:

1. For large dimensions, the complexity of the linear program can be substantial [17, 18]).

2. In addition to the large complexity, we observe numerical instability: We performed synthetic data test (where the extreme points are known a priori) and observed that, even ignoring complexity (for dimensions (> 4), the algorithms are numerically unstable.

We observe that both the problems are associated with the dimensionality D being large.

We implemented the synthetic data test as follows:

We build a $S \in \mathbb{R}^D$ such that we chose N_1 samples from a uniform distribution in $(-1000, 1000)^D$, and added 2^D samples explicitly such that these points are the vertices of the hypercube of $(-1000, 1000)^D$. Naturally, the explicitly added vertices of the hypercube are the extreme points of the set S. Hence, we know the extreme points of S apriori. When we run the Extreme Point enumeration of the Algorithm 1, we observe that the equality conditions in lines 6, 8 and 11 fails. We overcome this drawback with an implementation of a certain tolerance factor. With this modification, the resulting algorithm outputs the desired set of extreme points for small dimensions $D \leq 15$. For dimensions $Dgeq15$, we observe that the instability in the algorithm persists. With respect to the above case, when we choose $N_1 = 500$, the algorithm returns incorrect set of points as extreme, irrespective of finetuning the tolerance factor. We observed that slightly lowering the tolerance factor resulted in failing to choose any points as extreme (lines 6, 8 and 11 never being satisfied), while a slight increase in the tolerance factor resulted in choosing all the points in the set S as extreme points (i.e., all the $2^D + N_1$

[1] Note that a set with a minimum of $D + 1$ points in $\mathbb{R}^D$ is said to be in *general position*, if no more than D points from the set lie on any hyperplane.

Algorithm 1 Extreme points enumeration

Inputs: A finite set $S \in \mathbb{R}^D$ whose convex hull interior contains the origin, that is $\mathbf{0} \in \texttt{int}\,(\texttt{conv}(S))$;
Output $E = \texttt{ExtremePoints}(S)$**:** A finite set $E \subset S$ with the set of extreme points.
$E \leftarrow \{\}, S' \leftarrow S$
for $\mathbf{x} \in S'$ **do** $Q \leftarrow E \cup \{\mathbf{x}\}$
    Solve the linear program $f_{\mathbf{x}}(Q)$
    if $f_{\mathbf{x}}(Q) = f_{\mathbf{x}}(\{\mathbf{x}\})$ **then** ▷ $\mathbf{x}$ is an extreme point of Q
        Solve the linear program $f_{\mathbf{x}}(S)$; let $\mathbf{v}$ be its solution.
        if $f_{\mathbf{x}}(\{\mathbf{x}\}) = f_{\mathbf{x}}(S)$ **then**
            $E \leftarrow E \cup \{\mathbf{x}\}$ ▷ $\mathbf{x}$ is an extreme point of S
        else
            $R \leftarrow \{\mathbf{z} \in S : \mathbf{z}\mathbf{v}^T = 1\}$
            $E \leftarrow E \cup R; S' \leftarrow S \backslash R;$
        end if
    end if
end for

points). Additionally, the running time of the algorithm grows very large with even small values of D compared to the dimensionality of the real world datasets.

To overcome both the drawbacks, namely the numerical instability and the high computational complexity and large running time, we modify and employ the approach of `RandomizedExtremePoints` described in Algorithm 2. Towards this, we specifically choose the reduced dimensions lesser than 10, where we have the specific tolerance factor (optimized on the synthetic dataset of dimensions *up to* 10), supporting our choice of extreme points in the lower dimensions. The specifics of the algorithm are described in Algorithm 2.

Our modified randomized algorithm takes in the finite set S, and a lower dimension d, where $d < D$. The algorithm returns the set of extreme points E which is a subset of S. Essentially, the algorithm generates a random $d \times D$ matrix drawn from a Gaussian Distribution $\mathcal{N}(0, \mathbf{I}_{dD \times dD})$ in an i.i.d. manner. The random matrix $\mathbf{T}$ is normalized and the random projection to lower dimension is evaluated. The evaluated projection is passed on to the extreme point enumeration algorithm (Algorithm 1). The pre-images (contained in S) of the obtained extreme points of $U \subset \mathbb{R}^d$ in the lower dimension is returned as the set of extreme points E. We show in Section 3.2 that $\texttt{RandomizedExtremePoints}(S, d)$ approximates the convex hull of S so long as d is chosen appropriately. In all our experiments, we choose the value of d as $5, 8, 10$.

Algorithm 2 Randomized Extreme points enumeration

Inputs: A finite set $S \in \mathbb{R}^D$ whose convex hull interior contains the origin $\mathbf{0} \in \texttt{int}(\texttt{conv}(S))$; a parameter dimension d

Output $E = \texttt{RandomizedExtremePoints}(S, d)$**:** A finite set $E \subset S$ with the set of extreme points.

Random $d \times D$ matrix $\mathbf{T} \sim \mathcal{N}(0, \mathbf{I}_{dD \times dD})$

$R \leftarrow \left\{\sqrt{\frac{1}{d}}\mathbf{T}\mathbf{x} : \mathbf{x} \in \mathcal{S}\right\} \subset \mathbb{R}^d$

$U \leftarrow \texttt{ExtremePoints}(R)$

$E \leftarrow \{\mathbf{x} \in S : \mathbf{T}\mathbf{x} \in U\}$

2.4 Algorithm for Direction-based Data Augmentation

Data augmentation techniques have proven to increase the thickness of the margin of classification (Rajput et al., 2019 [2]) and hence an improvement in the performance of the classifiers. However, this results in the increase in the size of the resulting training data after employing augmentation techniques. The key point which motivates us to develop the modified data augmentation method is that the augmented points which move outward from the convex hull of finite set $S \in \mathbb{R}^D$ influence the margin of classification and the robustness of the classifier. However, naive methods of augmentation may result in the augmented points which move inward from the convex hull of the finite set S. This again is undesirable, as it is resulting in the increase in the overall size of the training data, but not influencing the thickness of the margin of classification. Also, we notice that the original interior points of the convex hull of finite set S does not contribute much to the thickness of the margin of classification.

To address these issues, we employ direction-based data augmentation techniques described in Algorithm 3. We employ the same random matrix $\mathbf{T}$ ($\mathbf{T} \sim \mathcal{N}(0, \mathbf{I}_{dD \times dD})$) used in our randomized extreme point enumeration algorithm (Algorithm 2. We feed this matrix $\mathbf{T}$, the subset S that needs to be augmented, the parameter N_{aug}, which decides the total number of directions along which the augmentations have to be performed, A hyperparameter α that controls the variance of noise which is added as part of the augmentation process. The algorithm outputs set of augmented points for each of the class $S_{aug} \subset \mathbb{R}^D$. Our algorithm employs the following linear program as a subroutine:

$$g_{\mathbf{x}}(\mathbf{T}, Q) = \arg\max_{\mathbf{y} \in \mathbb{R}^D} \mathbf{x}\mathbf{T}\mathbf{y}^T$$

$$s.t., \mathbf{z}\mathbf{T}\mathbf{y}^T \leq 1, \forall \mathbf{z} \in Q$$

Algorithm 3 Direction-based Data Augmentation

1: **Inputs:** A finite set $S \in \mathbb{R}^D$ whose convex hull interior contains the origin, that is $\mathbf{0} \in \texttt{int}\,(\texttt{conv}(S))$; the Random $d \times D$ matrix $\mathbf{T} \sim \mathcal{N}(0, \mathbf{I}_{dD \times dD})$ which was used to find the extreme points in $\texttt{RandomizedExtremePoints}(S, *)$; Number of Augmentations to perform: N_{aug}; Hyperparameter that controls the variance of noise: α

2: **Output** $E = \texttt{DirectedDataAug}(S, \mathbf{T}, N_{aug}, \alpha)$**:** A finite set S_{aug} with the set of augmented samples.

3: $S_{aug} \leftarrow \{\}$

4: **for** $i = \texttt{label} \in \{0, 1, ...(\mathcal{C} - 1)\}$ **do**

5: $\quad S_i \leftarrow \{\mathbf{x} : (\mathbf{x}, i) \in S\}$

6: $\quad S_{aug,i} \leftarrow \{\}$

7: $\quad$ **for** $j \in \{1, 2, ..., (N_{aug} - 1)\}$ **do**

8: $\quad\quad$ Generate a random Gaussian noise: $n_{i,j} \sim \mathcal{N}(0, \mathbf{I}_{D \times D})$

9: $\quad\quad$ Solve the linear program $g_{n_{i,j}}(\mathbf{T}, S_i \cup S_{aug,i})$ and assign: $v \leftarrow g_{n_{i,j}}(\mathbf{T}, S_i \cup S_{aug,i})$

10: $\quad\quad$ **for** $\mathbf{x} \in S_i$ **do**

11: $\quad\quad\quad$ **if** $v^T(T\mathbf{x}) = 1$ **then**

12: $\quad\quad\quad\quad S_{aug,i} = S_{aug,i} \cup \{\mathbf{x} + \alpha * n_{i,j}\}$

13: $\quad\quad\quad$ **end if**

14: $\quad\quad$ **end for**

15: $\quad$ **end for**

16: $\quad S_{aug} = S_{aug} \cup S_{aug,i}$

17: **end for**

The linear program is very similar to $f_{\mathbf{x}}(Q)$ which we saw earlier in the extreme point enumeration. However, in $g_{\mathbf{x}}(\mathbf{T}, Q)$, the vector which resulted in the maximum is returned and the maximization is performed in the projected domain.

We input the subset E, a set of the extreme points returned by the Randomized extreme point algorithm (Algorithm 2) as input to our data augmentation algorithm. Hence, we focus on augmenting only the extreme points instead of all the interior points (within the convex hull of S). In the algorithm, a noise vector $n_{i,j} \in \mathbb{R}^D$ is generated from a Gaussian distribution as follows: $n_{i,j} \sim \mathcal{N}(0, \mathbf{I}_{D \times D})$. For v being the solution of the linear program $g_{n_{i,j}}(\mathbf{T}, S_i \cup S_{aug,i})$ we monitor the equality condition in line 11 of Algorithm 3. The key point of our Augmentation algorithm is that we choose only those points that move outward from convex hull (the equality constrain in line 11 of Algorithm 3). The augmentation process is a simple weighted addition of α and $n_{i,j}$ vector with the those chosen points. Due to the numerical instability, we again implement our data augmentation algorithm using certain tolerance factor.

Chapter 3
Analytical Results

The properties that justify our choice of extreme points as representative data set were discussed in earlier chapter, namely Properties 1 and 2. In this chapter (refer section 3.1), we provide results that support these properties, and hence the choice of our representative dataset. In section 3.2, we describe our justification of our randomized algorithm and show that our algorithm indeed chooses the set of extreme points. We employ ***Johnson–Lindenstrauss lemma*** to show that the extreme points are chosen with a high probability and the chosen extreme points are a set of ϵ approximate extreme points. In the last section of this chapter, we explain the justification of our data augmentation process.

3.1 Margin of Linear classifiers

Let us familiarize ourselves with some standard notations:

We have our labeled dataset D and we aim to find a subset $S \subset D$ using our algorithm. The set S is the set of extreme points which preserves the convex hull of D. This subset S is a union of all the subsets S_i over all the classes. These notations are consistent with the notations we employed in the previous chapter to describe our algorithm implementations. We have our training algorithm, `Train`() which takes in the dataset to be trained on and outputs a classifier g, a mapping from the dataset $D \in \mathbf{R}^D$ or $S \in \mathbf{R}^D$ to one among the C classes, $\mathcal{C} := \{0, 1, 2, ..., C-1\}$. We represent margin of classifier g as `Margin(g, . )`. We have the following lemma, which are the elementary properties of convex hulls of polytopes.

Lemma 3.1.1. *A linear classifier g separates data set $\mathcal{D}$ if and only if it separates the subset $\mathcal{S}$.*

Our training algorithm `Train`() is designed to find a linear classifier g that separates any separable data set. If $\mathcal{D}$ is a linearly separable dataset, then the set $S \subset D$ is also linearly separable. If `Train`($\mathcal{S}$) produces a linear classifier g, which separates the dataset S, then `Train`($\mathcal{S}$) linearly separates $\mathcal{D}$ as well. Hence, supporting the property 1 of our approach of choosing extreme points as representative subset.

Lemma 3.1.2. *Consider a linear classifier g that separates data set $\mathcal{D}$. Then*

$$\textit{Margin}(g, \mathcal{D}) = \textit{Margin}(g, \mathcal{S})$$

This lemma provides the relation between the margins of classification. For a classifier g that separates data set S, and in turn separates data set D, where set $S \subset D$, is a set of extreme points, the margins are preserved after we apply our approach. There is an important take away from this lemma. In a special case, where a linear classification algorithm `Train` that finds/ approximates the *maximum margin* classifier g^* on $\mathcal{D}$ can be applied to the set $\mathcal{S} \subset D$ of extreme points to achieve the same effect. That is, `Train`($\mathcal{S}$) will also find/approximate the maximum margin classifier g^*- on $\mathcal{D}$.

Hence, the above Lemmas 3.1.1, 3.1.2 support and justify our approach to subset selection using the extreme point approach. In the next section, we justify our choice of *randomized algorithm*, `RandomizedExtremePoints` routine which is described in detail in Algorithm 2.

3.2 Approximation of Extreme Points

Given a set $S \in \mathbb{R}^D$, we have our transformation to the lower dimension as follows:

$$R = \{\mathbf{Tx} : \mathbf{x} \in S\}$$

where $R \in \mathbb{R}^d$. It is easy to show that $\mathbf{y}$ is an extreme point of R if it follows: $\mathbf{y} = \mathbf{Tx}$ for some $\mathbf{x}$ that is an extreme point of S. Hence, the set our randomized algorithm (Algorithm 2) returns is a subset of all the extreme points of S. In this section, we aim to show that it returns (nearly) all the extreme points of S.

We apply the ideas of Johnson- Lindenstrauss Lemma (J-L lemma) [19] to obtain an expression of what is the probability that our algorithm picks nearly all the extreme points of S. This section is one of the main contributions of this research work. When

the reduced dimension follows

$$d = \tilde{\Omega}\left(\frac{\log|S|}{\epsilon^2}\right),$$

we specifically use the fact that, the distances of the random Gaussian projection of finite set $S \in \mathbb{R}^D$ are preserved up to a factor of ϵ. In this section, we show that if the distances are preserved up to a factor of ϵ, then the set of extreme points are also approximately preserved in the projection to the lower dimension. When the lower dimension d follows $d = \Omega(\frac{\log|S|}{\epsilon^2})$, we apply J-L lemma to show that: with sufficiently high probability, a point $\mathbf{x}$ which is an extreme point of S is not returned by our randomized algorithm `RandomizedExtremePoints`(S) (Algorithm 2) only if the distance of $\mathbf{x}$ from the convex hull of $S - \{\mathbf{x}\}$ is at most $\epsilon \cdot \texttt{diam}(S)$. Towards this, we provide rigorous statements in the remaining part of the section.

Let us consider the definitions of two important concepts: $\epsilon-$ isometric transformation and δ-approximate extreme points, which will help us in our proofs.

ϵ-isometry:

Given a finite set $S \in \mathbb{R}^D$, the linear transformation $\mathbf{T}^{\epsilon}_S : \mathbb{R}^D \to \mathbb{R}^d$ is defined to be an ϵ-isometric transformation of S if it satisfies the following property:

$$\left|\frac{||\mathbf{x}-\mathbf{y}||^2 - ||\mathbf{T}^{\epsilon}_S(\mathbf{x}-\mathbf{y})||^2}{||\mathbf{x}-\mathbf{y}||^2}\right| \leq \epsilon$$

for all $\mathbf{x}, \mathbf{y} \in S$.

For a clean notation, we often omit the subscript S and superscript ϵ in $\mathbf{T}^{\epsilon}_S$. In the remaining part of this section, we employ this clean notation. Hence, when we use $\mathbf{T}$ in the sequel, we inherently mean that $\mathbf{T}$ is an $\epsilon-$ isometric transformation.

δ-approximate extreme points:

Consider a finite set $S \in \mathbb{R}^D$, a subset $E \subseteq S$ is said to be a set of δ-approximate extreme points of S iff

(i) E is a subset of the set of extreme points of S,

(ii) For a point $\mathbf{x} \in \texttt{conv}(S)$ we have:

$$d^2(\mathbf{x}, \texttt{conv}(E)) \leq \delta \texttt{diam}^2(S)$$

Our main results are the following theorem and its corollary.

Theorem 3.2.1. *Consider a finite set $S \in \mathbb{R}^D$. Let*

$$R = \{\mathbf{Tx} : \mathbf{x} \in S\},$$

where $\mathbf{T}$ *is an ϵ-isometric transformation of S. Let $E \subseteq S$ be a subset of the extreme points of S such that*

$$\texttt{conv}(R) = \texttt{conv}(\{\mathbf{Tx} : \mathbf{x} \in E\}).$$

Then E is a set of ϵ-approximate extreme points of S.

Corollary 3.2.1.1. *For any finite set $S \in \mathbb{R}^D$ with M extreme points, our algorithm* `RandomizedExtremePoints`(S, d) *(Algorithm 2) outputs an ϵ-approximate extreme point set of S with probability at least $2M^2e^{-d\epsilon^2/8}$.*

The details and proofs of Theorem 3.2.1 and Corollary 3.2.1.1 are provided in Appendix A. Note that we apply J-L lemma to Theorem 3.2.1 to obtain the proof of Corollary 3.2.1.1. There are two important implications from the corollary:

1. Our randomized algorithm `RandomizedExtremePoints`(S, d) (Algorithm 2) returns an ϵ-approximate extreme points set of S if $d = \Omega(\frac{\log M}{\epsilon^2})$ *with high probability.* It is worth noting that, even with the total number of points of S is large, our result only requires d to grow in a logarithmic fashion with respect to the number of the *extreme points* of S, .

2. It is important to note that the corollary establishes a trade-off between the dimension d and the quality of approximation ϵ,, given a fixed probability (bound). In other words, as we move to lower dimensional projection space, the computation and enumerations of the extreme points are faster with increased numerical stability, however, in the projected space, the approximations of the set of extreme points of S are correspondingly coarser.

3.3 Direction-based Data Augmentation

We have already established that Data Augmentations improve the robustness of the classifier by increasing the thickness of the margin of classifier (refer section 2.4). Also, from section 3.1, we have: for a classifier g that separates data set S, and in turn data set D, where set $S \subset D$, a set of extreme points of D, the margins are preserved, i.e., $\texttt{Margin}(g, \mathcal{D}) = \texttt{Margin}(g, \mathcal{S})$. This establishes a relation between the extreme points

and the margin of classifier. Hence, inspired by these observations, we claim that it is sufficient to augment only the extreme points of the set D to increase the thickness of the margin of classification. Towards this, if the augmented points end up inside the convex hull of the D, it is futile and does not conform with our claim. Hence, we develop our data-augmentation algorithm, $\texttt{DirectedDataAug}(S, \mathbf{T}, N_{aug}, \alpha)$ (Algorithm 3) that promote our ideas, where the data points after augmentation move outward of the convex hull of S.

In this section, we show that the augmented points do not lie inside the convex hull of S in the following Theorem, and hence, justify our algorithm.

Theorem 3.3.1. *Consider a finite set $S \in \mathbb{R}^D$, $\mathbf{T}$ is an ϵ-isometric transformation of S, N_{aug} is the total number of augmentations and α is the hyperparameter which controls the variance of noise employed to augment a data point $\mathbf{x}$. Our algorithm $\texttt{DirectedDataAug}(S, \mathbf{T}, N_{aug}, \alpha)$ returns S_{aug}, a set of all the augmented points of S in selected directions across all classes: $\cup_{i=0}^{C-1} S_{aug,i}$, where $S_{aug,i}$ is the set of augmented points in selected directions for a particular class i.*

Then, for $\texttt{conv}(S_i)$ representing the convex hull of S_i (where $S_i = \{\mathbf{x} : (\mathbf{x}, i) \in S\}$, a set of all the vectors $\mathbf{x} \in \mathbb{R}^D$ that are mapped to a particular class i), we have:

$$\texttt{conv}(S_i) \subset \texttt{conv}(S_{aug,i})$$

for all $\alpha > 0$ almost surely, for every class i.

This establishes a strict subset relation between set S_i and $S_{aug,i}$, implying that the augmented points are not contained in, but move outward of the convex hull of S_i. The details and proofs of Theorem 3.3.1 are provided in Appendix A. We employ the constrains of the linear program $g_{\mathbf{x}}(\mathbf{T}, Q)$ to obtain the proof of Theorem 3.3.1.

Chapter 4
Experimental Results

The analytical results support our choice of subset selection and the novel data augmentation techniques (refer chapter 3). In this chapter, we explore the practical implementation and experimental results which follow and conform with the analytical results we obtained. In section 4.1, we explain the setup and an overall summary about the data. Following this, we explain the model architecture (refer section 4.2) and performance evaluation (refer section 4.3). We finally conclude this chapter with results and observations (refer section 4.4).

4.1 Setup

For all our experiments, we choose standard MNIST dataset [20]. The data set comprises of handwritten digits of 0 to 9, with a total of $70,000$ images, where there are around $7,000$ images per class and the images are randomly segregated to training and test data sets. Each image is of size 28×28, represented as a 784×1 vector. The task in hand is to classify each image vector to its corresponding class (that is, 10 different classes). After the partition to training and test data sets, there are $60,000$ training image vectors and $10,000$ test image vectors. We employ a 5-fold cross-validation approach on the $60,000$ training images to generate 5 unique sets of training and cross-validation sets. Hence, we now have 5 sets of training images, each containing $48,000$ image vectors, and every validation set consisting of $12,000$ image vectors.

We employ a pre-processing stage where we divide each image vector in training, cross-validation and test data set by 255.0 to scale down every pixel value (in an image vector) $|p_i| \leq 1$, where $i = \{1, 2, ..., 784\}$. The Training set is shuffled before presenting to a learning model during training. The Validation data $(\mathbf{X}_v(i), Y_v(i))$ is employed to tune the hyper parameters such as learning rate and to terminate the training procedure

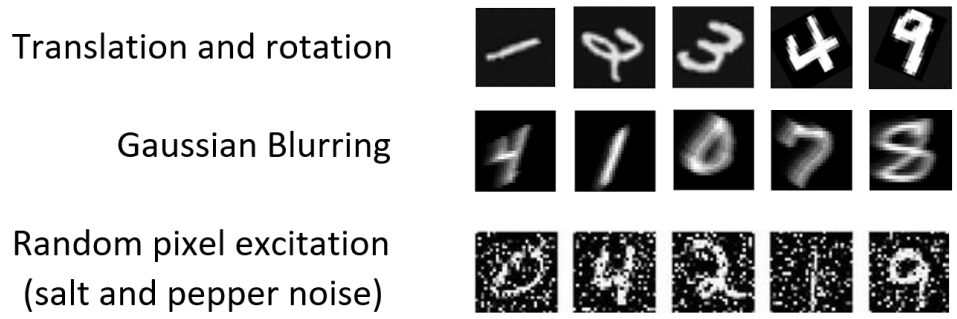

Figure 4.1: Coventional Augmentation Techniques on random MNIST data set images

to avoid over-fitting, and finally the model's performance is evaluated on the Test data $(\mathbf{X}_{te}, Y_{te})$.

4.1.1 Conventional data augmentation techniques

We perform random augmentations on the training data set, by employing 4 different types of augmentations, which include affine transformations like translation and rotation, and augmentations that add noise to the training images such as Gaussian Blurring and random pixel excitation (salt and pepper noise). The original training images are also preserved. Hence, the size of the total data set after augmentations is 4x the number of samples in the original data set. Sample augmentations for each of the different types of augmentation techniques are shown in Figure 4.1

4.1.2 Direction-based data augmentation techniques

We perform directed data augmentations using a noise vector drawn from a random Gaussian distribution: $n_{i,j} \sim \mathcal{N}(0, \mathbf{I}_{D\times D})$. We input the set of extreme points obtained from the `RandomizedExtremePoints`(S, d) (Algorithm 2 as input to the `DirectedDataAug`$(S, \mathbf{T}, N_{aug}, \alpha)$ (Algorithm 3, where the same random $\mathbf{T}$ matrix is used as in `RandomizedExtremePoints`(S, d). The data points that don't lie within the convex hull of S_i after augmentations are chosen and augmented as follows: $\mathbf{x} + \alpha * n_{i,j}$, where α monitors the variance of the noise that is being added. The process of our augmentation approach is depicted in the Figure 4.2.

4.1.3 Data Outline

For 5-fold cross validation, we have $i = \{0, 1, 2, 3, 4\}$. After the data subset selection and pre-processing, we obtain the following:

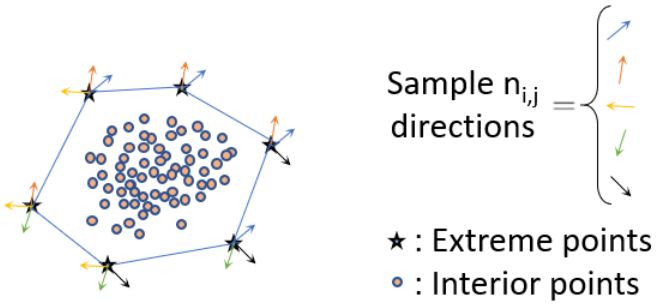

Figure 4.2: Depiction of our method of Directed Augmentation Techniques

- $(X_{tr}(i), Y_{tr}(i))$**:** The training data set (corresponding to i^{th} cross-validation set) and its ground truths that are used for baseline model training and evaluation.
- $(\mathbf{X}_{tr,a}(i), Y_{tr,a}(i))$**:** The training data set (corresponding to i^{th} cross-validation set) after augmentation and its ground truths that are used for baseline model training and evaluation.
- $(\mathbf{X}_{e}(i), Y_{e}(i))$**:** Extreme samples of the training set (corresponding to i^{th} cross-validation set) and its ground truths used to train and evaluate *our method*.
- $(\mathbf{X}_{e,a}(i), Y_{e,a}(i))$**:** Augmented extreme samples of the training set (corresponding to i^{th} cross-validation set) and its ground truths used to train and evaluate *our method*.
- $(\mathbf{X}_{r}(i), Y_{r}(i))$**:** Random samples of the training set (corresponding to i^{th} cross-validation set) and its ground truths used to train and evaluate the models. The number of samples chosen in random is same as the number of extreme samples.
- $(\mathbf{X}_{r,a}(i), Y_{r,a}(i))$**:** Random samples of the training set (corresponding to i^{th} cross-validation set) after augmentations and its ground truths used to train and evaluate our method.
- $(\mathbf{X}_{v}(i), Y_{v}(i))$**:** The i^{th} cross-validation set and its corresponding ground truths used for hyper parameter adjustment and stopping the training (to avoid over fitting) for all the experiments we conduct.
- $(\mathbf{X}_{te}, Y_{te})$**:** The test set and its corresponding ground truths used for final evaluation of all our experiments.

4.2 Model Architecture

For all our experiments, we implement linear classifier g. For a fair comparison, we employ the same linear classifier architecture (the simplest model architecture possible to achieve reasonable classification accuracy) for training and evaluation of our baseline, subset selection approaches and experiments on our data augmentation techniques. We implement our linear classifier model as a neural network with 784 input units and 10 output units corresponding to the 10 output classes. Our linear model has no hidden layers and no intermediate non-linear activation function layers after the input layer. We process the output layer through a softmax layer to obtain a probability value of what class the image vector might belong to. Each image vector is classified to its corresponding class based on the output unit that has the highest probability value. We train the model using categorical cross entropy loss using an Adam optimizer with a learning rate adjusted to reach an optimal value, based on the performance of the model on the cross-validation data set $(\mathbf{X}_v(i), Y_v(i))$. The model with the best validation accuracy is evaluated on the test data $(\mathbf{X}_{te}, Y_{te})$ The categorical cross entropy loss is described in Equation 4.1.

$$\begin{aligned} f(s)_i = \text{softmax}(s_i) &= \frac{e^{s_i}}{\sum_j^C e^{s_j}} \\ \text{Cross Entropy Loss} &= -\sum_i^C \{t_i * \log(f(s)_i\} \end{aligned} \tag{4.1}$$

where, t_i and $f(s)_i$ are the ground truth and the probability score (obtained by the softmax function of the neural network output s_i) respectively, for each class i in the total classes C.

A simple linear model is presented in Figure 4.3, where there are $7,850$ weight connections that map the image vectors $\mathbf{x} \in \mathbb{R}^D$ to one among C classes.

4.3 Metrics and Evaluation Criteria

The performance of our linear model is evaluated using classification accuracy which is defined in the Equation 4.2. The classification accuracy can be summarized as the ratio

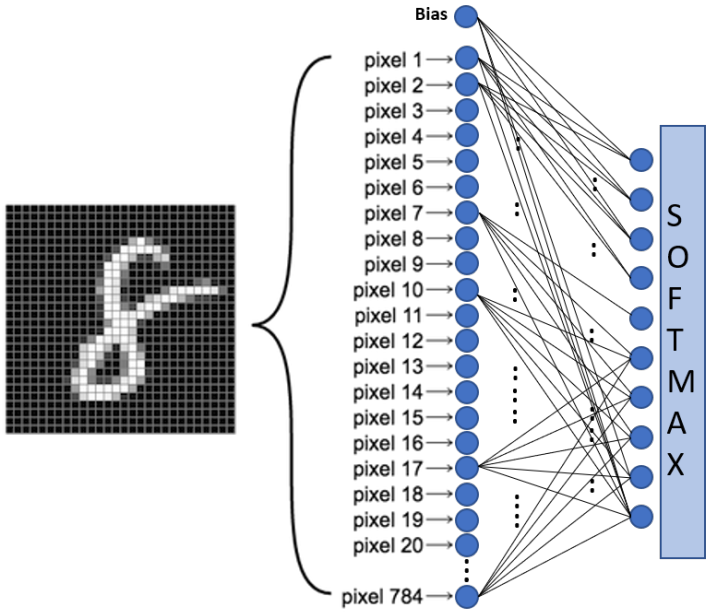

Figure 4.3: A simple linear classifier designed as a neural network with no hidden layers and no non-linear layers

of the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total number of predictions}} \tag{4.2}$$

For a comparison of cost of training among different experiments, we evaluate all our models using **total number of images used for training**. As the same linear classifier architecture is employed, the trainable parameters are fixed to $7,850$.

4.4 Experiments and Results

We perform rigorous experiments on linear classifier and the final classification accuracy result reported in the format { mean accuracy $\pm$ standard deviation } using the 5 sets of using $(X_{tr}(i), Y_{tr}(i))$ in a circular fashion. The input to the model is a 784 dimensional vector which is mapped to one of the ten classes (0 to 9). The performance of each of the experiments performed are tabulated in Table 4.1. The various experiments performed and the results are discussed in the following subsections.

4.4.1 Baseline

We employ two sets of experiments and consider these experiments as our baseline, namely,

1. Linear model trained on all the original training data $(X_{tr}(i), Y_{tr}(i))$ to convergence (48,000 training images and 12,000 cross validation images). In this setting, we observe from table 4.1 that the linear model achieves a mean test accuracy of 92.02%.

2. Linear model trained on the augmented training data $(X_{tr,a}(i), Y_{tr,a}(i))$ to convergence (192,000 training images and 12,000 cross validation images). In this setting, we observe from table 4.1 that the linear model achieves a mean test accuracy of 92.91%.

4.4.2 Extreme Point-based Subset Selection:

We perform the procedure `RandomizedExtremePoints` (Algorithm 2) on each class of the data set $(\mathbf{X}_{tr}(i), Y_{tr}(i))$, by projecting onto spaces of dimensions $10, 8$ and 5 respectively. We use the same random projection $\mathbf{T}$ is used for each class. For each case of d, we compare the performance of models trained in three different ways. Our first model is trained on a data set which contains only the extreme points $(\mathbf{X}_e(i), Y_e(i))$. We train our second model on a data set that consists of the conventional augmented extreme points $(\mathbf{X}_{e,a}(i), Y_{e,a}(i))$. Note that the size of this conventional augmented data set is four times the number of extreme points. Our third model is trained on the union of the original data set and the set of conventional augmented extreme points $(\mathbf{X}_{tr}(i), Y_{tr}(i)) \cup (\mathbf{X}_{e,a}(i), Y_{e,a}(i))$.

For our randomized extreme point algorithm `RandomizedExtremePoints`(S, d) (Algorithm 2) we choose $d = 5, 8, 10$ in our experiments. The output of our algorithm is a set of extreme points. For a visual depiction, we demonstrate the output of our algorithm for $d = 2$, for the digit 4 in Fig. 4.4. Interestingly, the extreme points appear to have an *interpretability* property, as they correspond to our intuitive perception of different writing styles for the same digit.

Also, when projected to the lower dimension $d = 2$, the depiction on the $2 - D$ plane is shown in figure 4.5.

We observe from table 4.1 that, with these settings, the performance of linear models trained on the extreme subsamples of data is very close to the performance of the baseline

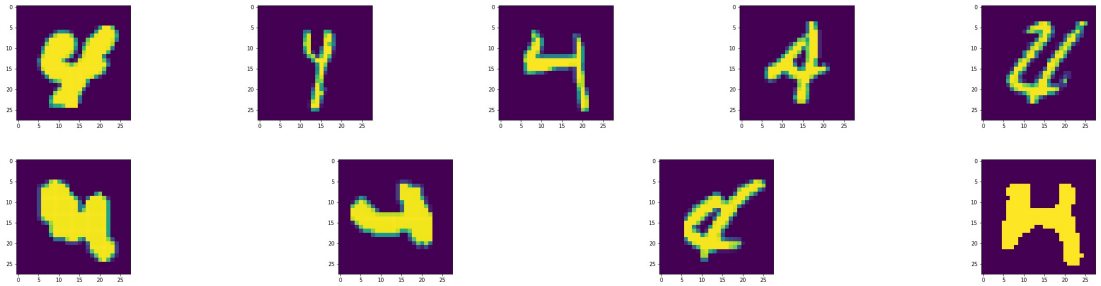

Figure 4.4: Outcomes of our randomized extreme point enumeration heuristic for the images corresponding to the digit 4s in the MNIST data set when d_0 is set to 2. The algorithm picks *subset* of 9 extreme points from a set of around 5000 images of '4'.

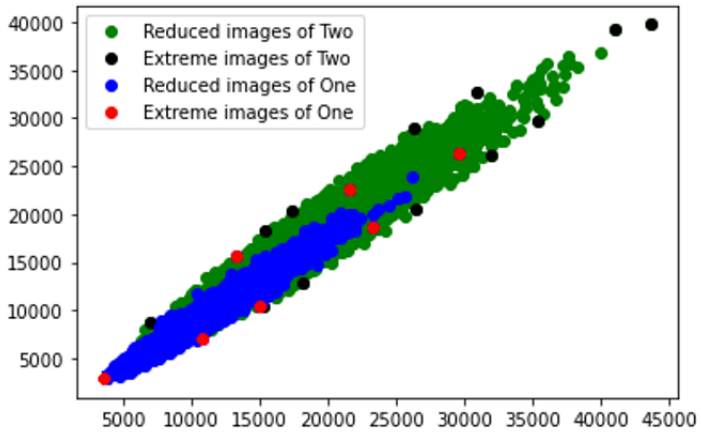

Figure 4.5: 2D Visualization of Extreme points for two classes: images of 1s and 2s

models. Specifically, we observe from table 4.1 that, the linear models trained on the augmented extreme points $(\mathbf{X}_{e,a}(i), Y_{e,a}(i))$, for dimensions $d = 8, 10$ have very close mean test classification accuracy of 90.98% and 92.31%, compared to 92.91% of the Baseline model trained on the Augmented samples $(\mathbf{X}_{tr,a}(i), Y_{tr,a}(i))$. When the model is trained on the union set of $(\mathbf{X}_{tr}(i), Y_{tr}(i)) \cup (\mathbf{X}_{e,a}(i), Y_{e,a}(i))$, we observe that the model's performance improves further. For dimension $d = 8$ we observe the mean test classification accuracy of 92.81% compared to 92.91% of the Baseline model, with a reduction in data size by 52%. We observe that model trained using the union set $(\mathbf{X}_{tr}(i), Y_{tr}(i)) \cup (\mathbf{X}_{e,a}(i), Y_{e,a}(i))$ for reduced dimension of $d = 10$, achieves a mean test classification accuracy of 92.98% and outperforms the baseline model (which has a test classification accuracy of 92.91%), with a reduction in data size by around 30%.

4.4.3 Random Subset

We perform similar experiments on a selection of random subsets of the original data set maintaining the same cardinality as the set of extreme points to analyze the performance of our randomized extreme point subset selection approach. As in the previous case, we develop three models: one over the random subset $(\mathbf{X}_r(i), Y_r(i))$, the second over the conventional augmented subset $(\mathbf{X}_{r,a}(i), Y_{r,a}(i))$, and the third over the union of the original data set and the conventional augmented subset $(\mathbf{X}_{tr}(i), Y_{tr}(i)) \cup (\mathbf{X}_{r,a}(i), Y_{r,a}(i))$.

We observe a general drop in performance with its extreme points (of the same cardinality) counterparts. This emphasizes the importance of the choosing the extreme points over random subsets of the whole data set.

Table 4.1: Results for a Linear Model (Trainable parameters: 7,850)

Experiments with 5-fold cross validation			Number of samples in Training Data	Training Accuracy (mean ± std-dev) (%)	Validation Accuracy (mean ± std-dev) (%)	Test Accuracy (mean ± std-dev) (%)
Baseline	All the data		48,000	90.97 ± 0.65	92.18 ± 0.36	92.02 ± 0.16
	Augmented samples		192,000	88.76 ± 0.42	92.41 ± 0.30	**92.91 ± 0.46**
Extreme point based subset selection and classification	Only Extreme Samples	Reduction to lower dimension				
		5	3,007	83.37 ± 0.36	87.34 ± 0.35	86.94 ± 0.19
		8	14,995	86.16 ± 0.25	90.53 ± 0.31	90.12 ± 0.23
		10	28,504	85.91 ± 0.32	91.63 ± 0.49	91.92 ± 0.18
	Augmented Extreme Samples	5	12,028	82.17 ± 0.28	88.15 ± 0.18	88.01 ± 0.11
		8	59,980	86.31 ± 0.19	89.97 ± 0.65	90.98 ± 0.53
		10	114,016	86.87 ± 0.41	91.93 ± 0.35	92.31 ± 0.22
	Training set + Augmented Extreme Samples	5	57,021	88.52 ± 0.39	91.98 ± 0.27	91.83 ± 0.13
		8	92,985	87.41 ± 0.68	92.85 ± 0.30	92.81 ± 0.21
		10	133,512	90.11 ± 0.54	92.95 ± 0.21	**92.98 ± 0.18**
Random samples based subset selection and classification	Only Random Samples	Samples chosen randomly to match the total number of Extreme samples	3,007	84.19 ± 0.25	86.11 ± 0.13	85.05 ± 0.29
			14,995	85.34 ± 0.52	87.18 ± 0.26	86.79 ± 0.33
			28,504	84.89 ± 0.28	88.25 ± 0.25	88.32 ± 0.31
	Augmented Random Samples		12,028	85.23 ± 0.33	87.13 ± 0.51	86.56 ± 0.11
			59,980	85.91 ± 0.25	87.93 ± 0.33	87.39 ± 0.14
			114,016	84.81 ± 0.47	90.65 ± 0.25	90.59 ± 0.27
	Training set + Augmented Random Samples		57,021	86.31 ± 0.35	88.99 ± 0.57	89.72 ± 0.26
			92,985	88.35 ± 0.57	90.84 ± 0.23	90.64 ± 0.35
			133,512	89.71 ± 0.24	92.32 ± 0.11	92.25 ± 0.17

The mean test classification accuracy is depicted on a line chart (refer figure 4.7) and bar chart (refer figure 4.6) for easy readability.

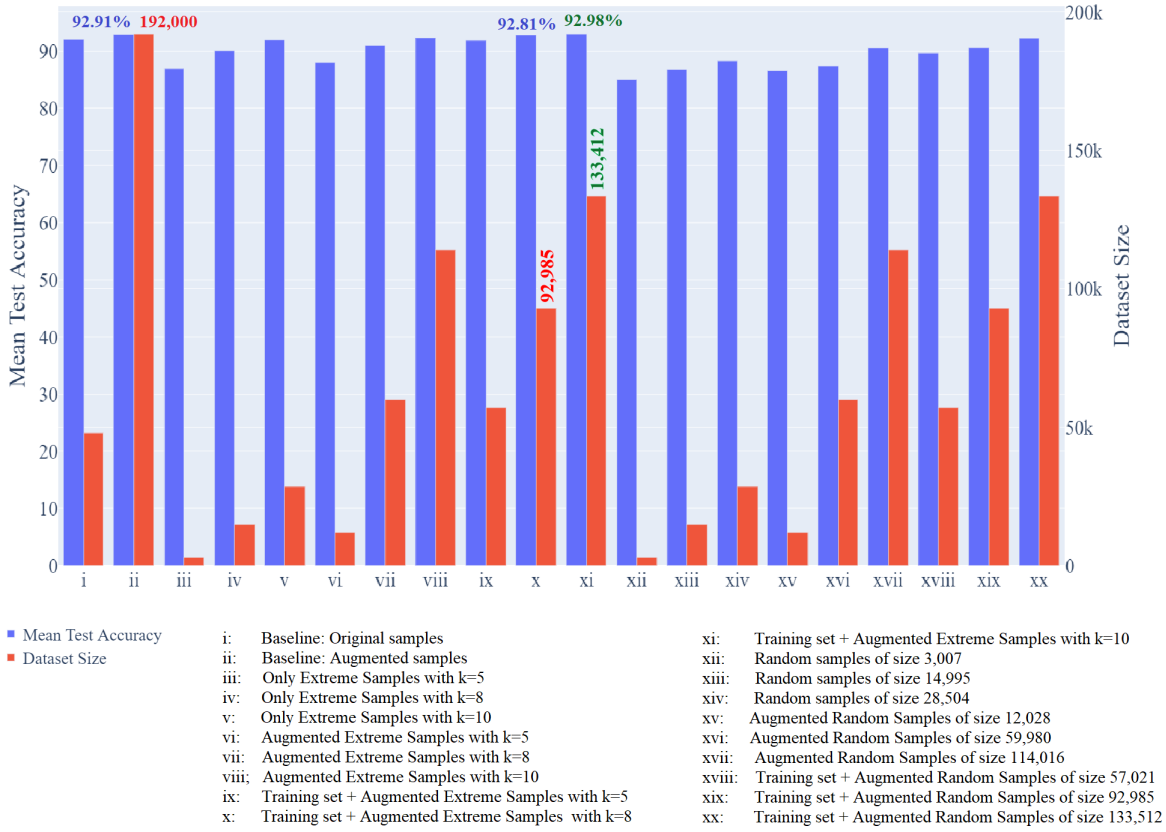

Figure 4.6: A summary of performance of all our experiments using Bar grapg representation

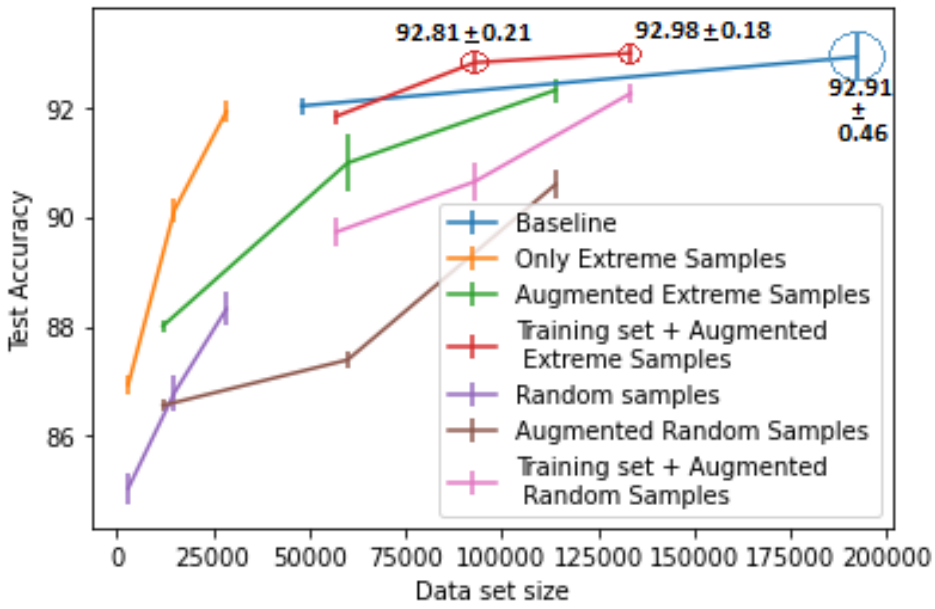

Figure 4.7: A summary of performance of all our experiments using line chart representation

Table 2. Results for a Linear Model (Trainable parameters: 7,850)

Experiments with 5-fold cross validation			Number of samples in Training Data	Training Accuracy (mean ± std-dev) (%)	Validation Accuracy (mean ± std-dev) (%)	Test Accuracy (mean ± std-dev) (%)
Baseline	**All the data**		48,000	90.97 ± 0.65	92.18 ± 0.36	92.02 ± 0.16
Extreme point based	**Augmented Extreme Samples**	$d_0 = 10$	114,016	86.87 ± 0.41	91.93 ± 0.35	92.31 ± 0.22
subset selection & classification	**Training set + Augmented Extreme Samples**	$d_0 = 10$	133,512	90.11 ± 0.54	92.95 ± 0.21	**92.98 ± 0.18**
Random samples based	**Augmented Random Samples**		114,016	84.81 ± 0.47	90.65 ± 0.25	90.59 ± 0.27
subset selection & classification	**Training set + Augmented Random Samples**		133,512	89.71 ± 0.24	92.32 ± 0.11	92.25 ± 0.17
Extreme point based	**Augmented Extreme Samples**	$d_0 = 10$	29,798	87.56	92.39	**92.54**
subset selection & classification	**Training set + Augmented Extreme Samples**	$d_0 = 10$	49,294	89.63	92.71	**92.88**
Random samples based	**Augmented Random Samples**		29,798	86.89	91.02	90.75
subset selection & classification	**Training set + Augmented Random Samples**		49,294	88.96	92.06	92.31

Figure 4.8: Performance of our novel direction-based data augmentation process

4.5 Direction-based data augmentation

We perform similar experiments as Extreme point subset selection and random subset selection, with a replacement of the conventional augmentation process with our novel data augmentation process. We perform the experiments for reduced dimension of $d = 10$ and repeat the random subset selection experiments to match the cardinality of the extreme points subset selection experiments. The results of these experiments are tabulated in Figure 4.8. A comparison study is performed, where we observe that with our augmentation techniques, we observe a further drop in the size of the training data set, with the maintenance of the mean test classification accuracy. Specifically, we observe that there is a further reduction of 26.13% for the augmented extreme samples case with the mean test accuracy of 92.54% (an improvement in the performance). However, with the case of the union set of original Training samples union with the augmented extreme samples, we observe that the mean test classification accuracy is very close to that of the counterpart of 92.92% with a further reduction in size of 36.92%. This emphasizes the fact that the performance can be maintained with a further reduction in size with a large reduction in size. The key point to keep in mind in all the experiments performed are that the subset selection is performed as a pre-processing stage for the linearly separable data set.

Chapter 5
Conclusions

The popularity of modern machine learning techniques across various disciplines is increasing in recent times. This has given rise to a great inflow of data. However, the increase in the data size poses several challenges, namely the cost of resources to manage and store data and also resulting in the large time taken to process and learn from the data. The data set size is bound to increase in the coming years. The existing machine learning algorithms become obsolete as the size of the data grows exponentially in the coming years. Evaluating the current scenario, there is definitely a need to develop algorithms that focuses on understanding the data and effectively finding a representative subset that reduces the training data set size as well as preserves the effective information contained in them. In this research work, we primarily focus and build our algorithms for linearly separable data set. We successfully develop novel data representation algorithm, which is based on representing the whole training data set using the set of extreme points. We show that the complexity of our novel approach is $O(MN^2)$ independent of the dimension d.

In this research work, we show that the chosen subset of points from our algorithm are effectively a set of $\epsilon-$ approximate extreme points and also show that, these $\epsilon-$ approximate extreme points are returned with with high probability. These constitute some of the key contributions of our work. Towards this, we have provided with detailed proofs and necessary experiments on standard MNIST data set (almost linearly separable data set) that support our algorithm. We have shown that, for the standard MNIST data set, the model trained on the subset returned by our algorithm coupled with data augmentation, achieves over 50% **reduction in size**, while maintaining the classification accuracy of the model trained on the whole data coupled with augmentation. Another key contribution to note is that our algorithm is effectively a pre-processing step for linearly separable data set, and hence, the overall running time of our algorithm is very

faster compared to the baseline and other existing works, which require a certain amount of training to find the core sets.

Along with this primary contribution, it is important to note that, the conventional data augmentation techniques we employed to obtain the above results, is again adding on to the growing size of the data sets. The conventional data augmentation techniques naively increase the size of the overall training set by *k-fold*, when $k-1$ different augmentation techniques are employed. Although the conventional data augmentation techniques improve the performance and increase the robustness of the learning models, it certainly has a drawback of increasing the data set size (which is the motivation for our research work). Hence, we emphasize the need to develop new data augmentation techniques which address these issues.

To tackle the problems with the largely increasing data size after conventional augmentations, we introduce a novel data augmentation technique in this research work, which focuses on augmenting the data points only in specific directions. With this technique, we successfully show that, only certain subset of points are chosen for a particular direction of noise which are augmented, hence, the overall size of the data set after augmentation does not increase drastically. One of the key contributions of this research work is to demonstrate and prove that the convex hull of the augmented points contains the convex hull of the original training data, hence, showing that the augmentations performed are along specific directions.

Along with the theoretical results, we also show necessary experimental results that support our novel approach. We show that the model trained on the augmented points returned by our algorithm coupled with all the original training data achieves a **further reduction** in data size by **over** 25% against the model trained on the augmented points (using conventional data augmentation techniques) coupled with all the original training data, while maintaining a very close classification accuracy.

In this research work, we successfully present novel approaches that tackle the problem of increasing data size, while effectively preserving the information contained in the original training data, by demonstrating both analytically and experimentally that the model designed on our representative data set performs equally with the model designed on the whole data set with an effective reduction in size. We further extend our ideas of representative data set and develop a data augmentation technique which again promotes scalable training by improving the performance and making the model more robust, with a further reduction in the data size. We effectively show this using theoretical and experimental results.

Chapter 6
Future Works

More often we find that the real world data sets are non-linear and we need deep and non-linear classifiers to achieve a good performance on the data set. However, in our research, our focus is on finding a solution to linearly separable data sets and built around linear classifiers. To address this, we can extend our research to non-linear classifiers by a small modification on our approach. The penultimate layer of a non-linear classifier effectively contains a single set of weight parameters w that needs to be optimized on the highly level data representation of the non linear layer output prior to the penultimate layer. When we single out the penultimate layer, this is effectively a linear classifier, which takes in the output of the non-linear classifier (without the penultimate layer). This is again mapped to C classes (refer Figure 6.1). Hence, we can apply our approach to find the extreme samples and also the novel data augmentation techniques to improve the model performance.

Toward this, we present a few preliminary results in Figure 6.2. Further, we aim to perform similar set of experiments on other standard data sets for a thorough experimental backup, namely Fashion-MNIST [21], CIFAR-10 [22] and CIFAR-100 [23].

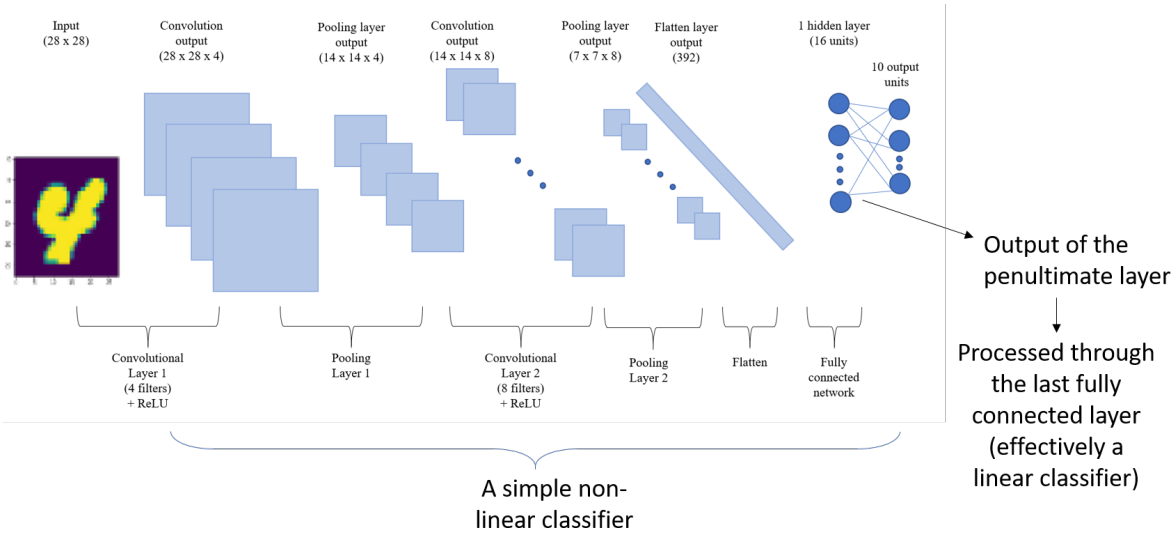

Figure 6.1: Extenson to Non-linear classifiers

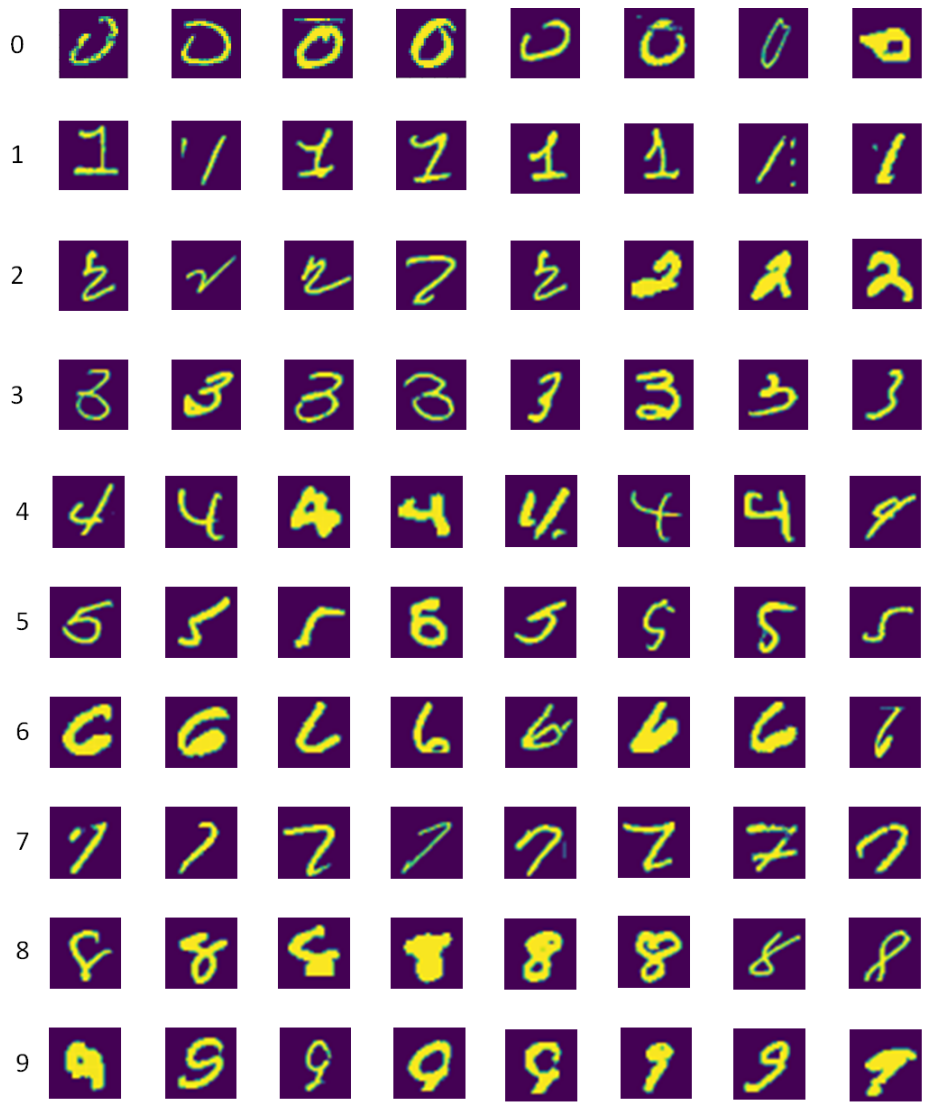

Figure 6.2: MNIST Extreme sample examples from penultimate layer output

Appendix A
Proofs

We begin with the following key lemma. Lemma A.0.1 is about the properties of inner products.

Lemma A.0.1. *Let $\mathbf{T}_{d\times D}$ be an ϵ-isometric transformation on a set $S \subseteq \mathbb{R}^D$. Then, $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in S$, we have:*

$$|\langle \mathbf{x}-\mathbf{z}, \mathbf{y}-\mathbf{z}\rangle - \langle \mathbf{Tx}-\mathbf{Tz}, \mathbf{Ty}-\mathbf{Tz}\rangle| \leq \epsilon \texttt{diam}^2(S)$$

In other words, the inner products after transformation is preserved with a factor of ϵ.

Proof. Let $\mathbf{T}$ be an ϵ-isometric transformation and $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in S$, we observe that:

$$\langle \mathbf{x}-\mathbf{z}, \mathbf{y}-\mathbf{z}\rangle = \frac{||\mathbf{x}-\mathbf{z}||^2 + ||\mathbf{y}-\mathbf{z}||^2 - ||\mathbf{x}-\mathbf{y}||^2}{2}$$

$$\langle \mathbf{Tx}-\mathbf{Tz}, \mathbf{Ty}-\mathbf{Tz}\rangle = \frac{||\mathbf{Tx}-\mathbf{Tz}||^2 + ||\mathbf{Ty}-\mathbf{Tz}||^2 - ||\mathbf{Tx}-\mathbf{Ty}||^2}{2}$$

$$\texttt{diam}^2(S) = \frac{||\mathbf{x}-\mathbf{z}||^2 + ||\mathbf{y}-\mathbf{z}||^2 + ||\mathbf{x}-\mathbf{y}||^2}{2}$$

Also, from the definitions of ϵ-isometric transformation, we have

$$(1-\epsilon)||\mathbf{x}-\mathbf{y}||^2 \leq ||\mathbf{Tx}-\mathbf{Ty}||^2 \leq (1+\epsilon)||\mathbf{x}-\mathbf{y}||^2 \tag{A.1}$$

Or (A.1) can be rewritten as:

$$-(1+\epsilon)||\mathbf{x}-\mathbf{y}||^2 \leq ||\mathbf{Tx}-\mathbf{Ty}||^2 \leq -(1-\epsilon)||\mathbf{x}-\mathbf{y}||^2 \tag{A.2}$$

Similarly, we have,

$$(1-\epsilon)||\mathbf{x}-\mathbf{z}||^2 \leq ||\mathbf{Tx}-\mathbf{Tz}||^2 \leq (1+\epsilon)||\mathbf{x}-\mathbf{z}||^2 \tag{A.3}$$

and

$$(1-\epsilon)||\mathbf{y}-\mathbf{z}||^2 \leq ||\mathbf{Ty}-\mathbf{Tz}||^2 \leq (1+\epsilon)||\mathbf{y}-\mathbf{z}||^2 \tag{A.4}$$

$\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in S$.

Combining, (A.2), (A.3) and (A.4), we get:

$$\begin{aligned}\frac{(1-\epsilon)(||\mathbf{x}-\mathbf{z}||^2+||\mathbf{y}-\mathbf{z}||^2)-(1+\epsilon)||\mathbf{x}-\mathbf{y}||^2}{2} &\leq \langle \mathbf{Tx}-\mathbf{Tz}, \mathbf{Ty}-\mathbf{Tz}\rangle \\ &\leq \frac{(1+\epsilon)(||\mathbf{x}-\mathbf{z}||^2+||\mathbf{y}-\mathbf{z}||^2)-(1-\epsilon)||\mathbf{x}-\mathbf{y}||^2}{2}\end{aligned}$$

$$\begin{aligned}\langle \mathbf{x}-\mathbf{z}, \mathbf{y}-\mathbf{z}\rangle - \epsilon \texttt{diam}^2(S) \leq \langle \mathbf{Tx}-\mathbf{Tz}, \mathbf{Ty}-\mathbf{Tz}\rangle \\ \leq \langle \mathbf{x}-\mathbf{z}, \mathbf{y}-\mathbf{z}\rangle + \epsilon \texttt{diam}^2(S)\end{aligned}$$

Hence we get,

$$\boxed{|\langle \mathbf{x}-\mathbf{z}, \mathbf{y}-\mathbf{z}\rangle - \langle \mathbf{Tx}-\mathbf{Tz}, \mathbf{Ty}-\mathbf{Tz}\rangle| \leq \epsilon \texttt{diam}^2(S)}$$

as desired.

□

A.0.1 Proof of Theorem 3.2.1

We begin with the following lemma.

Lemma A.0.2. *Consider a finite set $S \subseteq \mathbb{R}^D$ and $\mathbf{T}$ is an ϵ-isometric transformation of S. Let $\overline{E}$ be the set of extreme points of S, and $\overline{R} = \{\mathbf{Tx} : \mathbf{x} \in \overline{E}\}$, then,*

(A) There exists a set $E \subseteq \overline{E}$ such that $\{\mathbf{Tx} : \mathbf{x} \in E\}$ is the set of extreme points of $\overline{R}$.

(B) The set E satisfies: $\texttt{conv}(R) = \texttt{conv}(\{\mathbf{Tx} : \mathbf{x} \in E\})$.

(C) $\mathbf{x} \in \overline{E} \backslash E \Rightarrow d^2(\mathbf{x}, E) \leq \epsilon \ \texttt{diam}^2(\overline{E})$

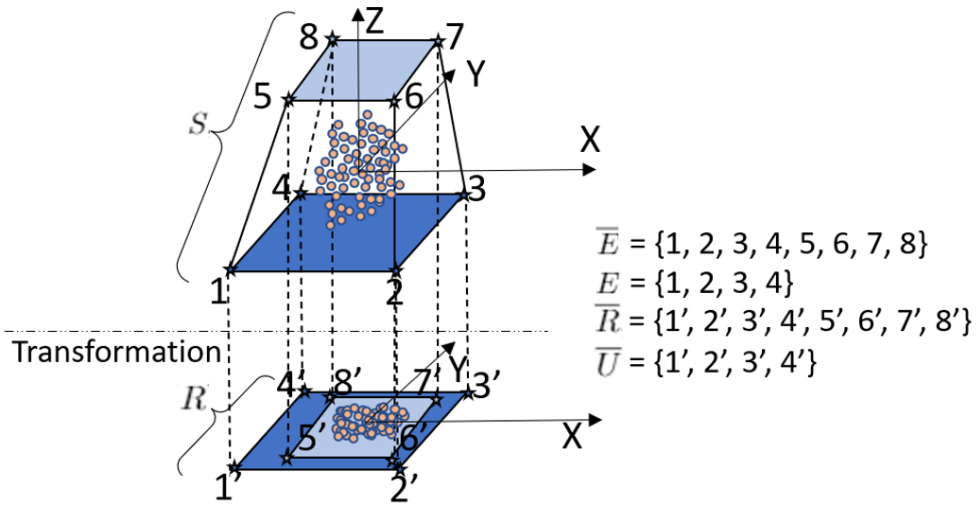

Figure A.1: Visual representation of lemma A.0.2: depiction of projection from 3-D to 2-D space

Proof. Proof of (A):

Let $\overline{U}$ be a set of extreme points of $\overline{R}$. So, for an extreme point $\mathbf{u} \in \overline{U}$, there is an element $\mathbf{x} \in \overline{E}$ such that $\mathbf{Tx} = \mathbf{u}$. Choose E to be the collection of such points in $\overline{E}$, that is, $E = \{\mathbf{x} \in \overline{E} : \mathbf{Tx} \in \overline{U}\}$. Hence, there exists a set $E \subseteq \overline{E}$ such that $\overline{U} = \{\mathbf{Tx} : \mathbf{x} \in E\}$ is the set of extreme points of $\overline{R}$.

Proof of (B):

Consider an element $\mathbf{y} \in R$, where $R = \{\mathbf{Tx} : \mathbf{x} \in S\}$. There is a corresponding element $\mathbf{u} \in S$, such that $\mathbf{y} = \mathbf{Tu}$. Since $\overline{E}$ is the set of extreme points of S, we have

$$\mathbf{u} = \sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} \mathbf{z}$$

where $\lambda_{\mathbf{z}} \geq 0$ and $\sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} = 1$. Therefore, we have

$$\mathbf{y} = \mathbf{Tu} = \sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{x},\mathbf{z}} \mathbf{Tz}$$

That is, $\mathbf{y}$ is within the convex hull of $\overline{R}$. Hence, $\mathbf{y}$ can be written as a convex combination of the extreme points of $\overline{R}$. Recalling that $\overline{U} = \{\mathbf{Tx} : \mathbf{x} \in E\}$ is the set of extreme points of $\overline{R}$, we have:

$$\texttt{conv}(R) \subseteq \texttt{conv}(\{\mathbf{Tx} : \mathbf{x} \in E\}).$$

Since $E \subseteq S$, it is trivial that $\texttt{conv}(\{\mathbf{Tx} : \mathbf{x} \in E\}) \subseteq \texttt{conv}(R)$. Hence, we get:

$$\texttt{conv}(R) = \texttt{conv}\left(\{\mathbf{Tx} : \mathbf{x} \in E\}\right).$$

Proof of (C):

Consider an element $\mathbf{x} \in \overline{E} - E$. As $\mathbf{x}$ lies within $\texttt{conv}\left(\{\mathbf{Tx} : \mathbf{x} \in E\}\right)$ in the transformed space, we have:

$$\mathbf{Tx} = \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{Tz}$$

where $\lambda_{\mathbf{z}} \geq 0$ and $\sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} = 1$.

Now, we examine the distance:

$$||\mathbf{x} - \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{z}||^2 \tag{A.5}$$

$$= ||\mathbf{x} - \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{z}||^2 - ||\mathbf{Tx} - \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{Tz}||^2 \tag{A.6}$$

$$= || \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{x} - \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{z}||^2 - || \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{Tx} - \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{Tz}||^2 \tag{A.7}$$

$$= || \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} (\mathbf{x} - \mathbf{z})||^2 - || \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} (\mathbf{Tx} - \mathbf{Tz})||^2 \tag{A.8}$$

$$= \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}}^2 \left(||\mathbf{x} - \mathbf{z})||^2 - ||\mathbf{Tx} - \mathbf{Tz})||^2 \right) - 2 \sum_{\mathbf{z}, \mathbf{z}' \in E, \mathbf{z} \neq \mathbf{z}'} \lambda_{\mathbf{z}} \lambda_{\mathbf{z}'} \Big(\langle \mathbf{x} - \mathbf{z}, \mathbf{x} - \mathbf{z}' \rangle - \langle \mathbf{T}(\mathbf{x} - \mathbf{z}), \mathbf{T}(\mathbf{x} - \mathbf{z}') \rangle \Big) \tag{A.9}$$

$$\leq \sum_{\mathbf{z} \in E} \left(\lambda_{\mathbf{z}}^2 ||\mathbf{x} - \mathbf{z})||^2 - ||\mathbf{Tx} - \mathbf{Tz})||^2 \right) + 2 \sum_{\mathbf{z}, \mathbf{z}' \in E, \mathbf{z} \neq \mathbf{z}'} \lambda_{\mathbf{z}} \lambda_{\mathbf{z}'} \left(|\langle \mathbf{x} - \mathbf{z}, \mathbf{x} - \mathbf{z}' \rangle - \langle \mathbf{T}(\mathbf{x} - \mathbf{z}), \mathbf{T}(\mathbf{x} - \mathbf{z}') \rangle| \right) \tag{A.10}$$

$$\leq \epsilon \texttt{diam}^2(S) \left(\sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}}^2 + \sum_{\mathbf{z}, \mathbf{z}' \in E, \mathbf{z} \neq \mathbf{z}'} 2 \lambda_{\mathbf{z}} \lambda_{\mathbf{z}'} \right) \tag{A.11}$$

$$\leq \epsilon \texttt{diam}^2(S) \tag{A.12}$$

In the above relations, (a) holds (because $\mathbf{Tx} = \sum_{\mathbf{z} \in E} \lambda_{\mathbf{z}} \mathbf{Tz}$) (b) uses the fact that $\mathbf{T}$ is an ϵ-isometric linear transformation. Specifically, because for all $\mathbf{x}, \mathbf{z}, \mathbf{z}' \in E$, we have that

$$||\mathbf{x} - \mathbf{z}||^2 - ||\mathbf{Tx} - \mathbf{Tz}||^2 \leq \epsilon ||\mathbf{x} - \mathbf{z}||^2 \leq \epsilon \texttt{diam}^2(S)$$

and, from Lemma A.0.1,

$$|\langle \mathbf{x} - \mathbf{z}', \mathbf{x} - \mathbf{z} \rangle - \langle \mathbf{Tx} - \mathbf{Tz}', \mathbf{Tx} - \mathbf{Tz} \rangle| \leq \epsilon \texttt{diam}^2(S),$$

and using Cauchy-Schwarz inequality in the penultimate relation above.

Finally, (c) holds (because $\sum_{\mathbf{z}\in E}\lambda_{\mathbf{z}} = 1$)

□

Proof of Theorem 3.2.1. Note that $\texttt{diam}(S) = \texttt{diam}(\overline{E})$ and $\mathbf{T}$ is an ϵ-isometric transformation on S as well as on $\overline{E}$. We also have,

$$d^2(\mathbf{x}, E) \leq \epsilon\texttt{diam}^2(\overline{E}) = \epsilon\texttt{diam}^2(S)$$

for any $\mathbf{x} \in E$ from Lemma A.0.2. As $\overline{E}$ is a set of extreme points of S, an arbitrary $\mathbf{y} \in S$ can be represented as:

$$\mathbf{y} = \sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}}\mathbf{z}$$

where $\lambda_z \geq 0 and \sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}} = 1$. Let $f(\mathbf{z})$ denote the closest point to $\mathbf{z} \in \overline{E}$, in $\texttt{conv}(E)$. Let us examine:

$$\begin{aligned}
&d^2(\mathbf{y}, \texttt{conv(E)}) \\
&\leq ||\mathbf{y} - \sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}} f(\mathbf{z})||^2 \\
&= ||\sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}}\mathbf{z} - \lambda_{\mathbf{z}} f(\mathbf{z})||^2 \\
&= ||\sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}}(\mathbf{z} - f(\mathbf{z}))||^2 \\
&\leq \sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}}^2||(\mathbf{z} - f(\mathbf{z}))||^2 + 2\sum_{\mathbf{z},\mathbf{z}'\in\overline{E}} \lambda_{\mathbf{z}}\lambda_{\mathbf{z}'}||(\mathbf{z} - f(\mathbf{z}))||||(\mathbf{z}' - f(\mathbf{z}'))|| \\
&\leq \epsilon\texttt{diam}^2(S)\left(\sum_{\mathbf{z}\in\overline{E}} \lambda_{\mathbf{z}}^2 + 2\sum_{\mathbf{z},\mathbf{z}'\in\overline{E}} \lambda_{\mathbf{z}}\lambda_{\mathbf{z}'}\right) \\
&= \epsilon\texttt{diam}^2(S)
\end{aligned}$$

$\implies$ E is an ϵ-approximate set of extreme points of S. (Recalling the definitions of δ-approximate extreme points)

□

To prove Corollary 3.2.1.1, we use the JL Lemma stated below.

Lemma A.0.3 (Johnson-Lindenstrauss, [19])**.** *states that, for any finite set $U \subset \mathbb{R}^D$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$, and $\mathbf{T}$ be a random $d \times D$ matrix drawn from a Gaussian distribution*

$\mathbf{T} \sim \mathcal{N}(0, \mathbf{I}_{dD \times dD})$, *we have*

$$\mathbb{P}\bigg((1-\epsilon)||\mathbf{x}-\mathbf{y}||^2 \leq ||\mathbf{T}\mathbf{x}-\mathbf{T}\mathbf{y}|| \leq (1+\epsilon)||\mathbf{x}-\mathbf{y}||^2\bigg) \leq 2|U|^2 e^{-d\epsilon^2/8}$$

Proof of Corollary 3.2.1.1. Let $\mathbf{T}$ be a random $d \times D$ matrix drawn from a Gaussian distribution $\mathbf{T} \sim \mathcal{N}(0, \mathbf{I}_{dD \times dD})$.

Let $\overline{E}$ be the set of extreme points of S, where $M = |\overline{E}|$. Consider $\mathbf{x} \in S$. We have $\mathbf{x} = \sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} \mathbf{z}$ where $\lambda \geq 0$ and $\sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} = 1$. Therefore,

$$\begin{aligned}
&d(\mathbf{x}, \texttt{conv}(E)) \\
&= d(\sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} \mathbf{z}, \texttt{conv}(E)) \\
&\leq \sum_{\mathbf{z} \in \overline{E}} \lambda_{\mathbf{z}} d(\mathbf{z}, \texttt{conv}(E)) \\
&\leq \sup_{\mathbf{z} \in \overline{E}} (d(\mathbf{z}, \texttt{conv}(E))
\end{aligned}$$

where (a) holds (because $d(., A)$ is a convex function for any convex set A). Also note that because $\overline{E} \subset S$, for any point $\mathbf{v} \in \overline{E}$, we have

$$d(\mathbf{v}, \texttt{conv}(E)) \leq \sup_{\mathbf{z} \in S} (d(\mathbf{z}, \texttt{conv}(E)$$

Inequalities $(b), (c)$ readily imply that

$\implies$ E is an ϵ-approximate set of extreme points of $\overline{E}$ if and only if it is ϵ-approximate set of extreme points of S. Because of Theorem 3.2.1, it suffices to lower bound the probability that $\mathbf{T}$ is an ϵ-isometric transformation of $\overline{E}$.

From JL Lemma applied to $\overline{E}$, that with probability $2M^2 e^{-d\epsilon^2/(8)}$, $\mathbf{T}$ is an ϵ-isometry on $\overline{E}$. The corollary naturally follows on applying Theorem 3.2.1.

A.0.2 Proof of Theorem 3.3.1

We begin with the following lemma:

Lemma A.0.4. *In the* `DirectedDataAug`$(S, \mathbf{T}, N_{aug}, \alpha)$, *if the reduced lower dimension* $d = D$, *and* $\mathbf{T} = \mathbf{I}_{DXD}$, *then,* `conv`$(S_i) \subset$ `conv`$(S_{aug,i})$ *for all non-negative* α *almost surely for every class* i.

Proof. We assume noise $\mathbf{n}_{i,j} \neq 0$ for all the iterations in the for loop. For $d = D$, and $\mathbf{T} = \mathbf{I}_{D\mathtt{x}D}$, we have our linear program as follows:

$$\mathbf{v} = g_{\mathbf{n_{i,j}}}(\mathbf{I}, S) = \arg\max_{\mathbf{y}\in\mathbb{R}^D} \mathbf{n_{i,j}}\mathbf{T}\mathbf{y}^T$$

$$s.t., \mathbf{z}\mathbf{I}\mathbf{y}^T = \mathbf{z}\mathbf{y}^T \leq 1, \forall \mathbf{z} \in S$$

Note that $\forall \mathbf{z} \in S$,

$$\mathbf{v}^T\mathbf{z} \leq 1.$$

Hence, this implies that, for any point $\mathbf{z}^{'} \in \mathtt{conv}(S)$,

$$\mathbf{v}^T\mathbf{z}^{'} \leq 1.$$

We observe that, the augmented sample $\mathbf{w} + \alpha * \mathbf{n}_{i,j}$, (where $\mathbf{w}$ is chosen such that $\mathbf{w} \in S$ and $v^T\mathbf{w} = 1$) has its inner product with v as follows:

$$\mathbf{v}^T(\mathbf{w} + \alpha * \mathbf{n}_{i,j}) = \mathbf{v}^T\mathbf{w} + \alpha * \mathbf{v}^T\mathbf{n}_{i,j} = 1 + \alpha * \mathbf{v}^T\mathbf{n}_{i,j}.$$

For non-zero $\mathbf{n}_{i,j}$ and non-negative $\alpha, \mathbf{v}^T\mathbf{n}_{i,j}$ is always greater than 0, as long as $''0''$ is strictly in S and S is in general position.

Hence, we have our augmented sample $\{\mathbf{w} + \alpha * \mathbf{n}_{i,j}\} \notin \mathtt{conv}(S)$.

In other words, $\mathtt{conv}(S)$ is strictly contained in $S \cup \{\mathbf{w} + \alpha * \mathbf{n}_{i,j}\}$

We can extend the same analogy to the case when $d \neq D$ and the result $\{\mathbf{w}+\alpha*\mathbf{n}_{i,j}\} \notin \mathtt{conv}(S)$ still holds.

Hence, we have $\{\mathbf{w} + \alpha * \mathbf{n}_{i,j}\} \in \mathtt{conv}(S_{aug,i})$, and

$$\boxed{\mathtt{conv}(S_i) \subset \mathtt{conv}(S_{aug,i})}$$

as desired.

□

□

Appendix B
Algorithm Outcomes

B.0.1 Outcomes of our randomized extreme point enumeration heuristic when d is set to 2

We see that the algorithm picks a subset of 103 extreme points when the projection dimension $d = 2$. We can observe that these extreme points represent different ways of writing each digit. It is interesting to note that the algorithm has interpretability property, which can be observed from table B.1

Table B.1: Outcomes of our randomized extreme point enumeration heuristic for the images corresponding to the all the digits in the MNIST data set when d is set to 2. The algorithm picks *subset* of 103 extreme points from a set of around 50000 images.

Bibliography

[1] Schroff, F., D. Kalenichenko, and J. Philbin (2015) "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

[2] Rajput, S., Z. Feng, Z. Charles, P.-L. Loh, and D. Papailiopoulos (2019) "Does Data Augmentation Lead to Positive Margin?" in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 5321–5330. URL `http://proceedings.mlr.press/v97/rajput19a.html`

[3] Charles, Z., S. Rajput, S. Wright, and D. Papailiopoulos (2019) "Convergence and margin of adversarial training on separable data," *arXiv preprint arXiv:1905.09209*.

[4] Yang, Y., R. Khanna, Y. Yu, A. Gholami, K. Keutzer, J. E. Gonzalez, K. Ramchandran, and M. W. Mahoney (2020) "Boundary thickness and robustness in learning models," *arXiv preprint arXiv:2007.05086*.

[5] Ottmann, T., S. Schuierer, and S. Soundaralakshmi (1995) "Enumerating extreme points in higher dimensions," in *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, pp. 562–570.

[6] Bachem, O., M. Lucic, and A. Krause (2017) "Practical coreset constructions for machine learning," *arXiv preprint arXiv:1703.06476*.

[7] Jubran, I., A. Maalouf, and D. Feldman (2019) "Introduction to coresets: accurate coresets," *arXiv preprint arXiv:1910.08707*.

[8] Har-Peled, S. and S. Mazumdar (2004) "On coresets for k-means and k-median clustering," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 291–300.

[9] Huang, L. and N. K. Vishnoi (2020) "Coresets for clustering in euclidean spaces: Importance sampling is nearly optimal," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1416–1429.

[10] Munteanu, A., C. Schwiegelshohn, C. Sohler, and D. P. Woodruff (2018) "On coresets for logistic regression," *arXiv preprint arXiv:1805.08571*.

[11] Raj, A., C. Musco, and L. Mackey (2020) "Importance sampling via local sensitivity," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 3099–3109.

[12] Mirzasoleiman, B., J. Bilmes, and J. Leskovec (2020) "Coresets for data-efficient training of machine learning models," in *International Conference on Machine Learning*, PMLR, pp. 6950–6960.

[13] Dao, T., A. Gu, A. Ratner, V. Smith, C. De Sa, and C. Ré (2019) "A kernel theory of modern data augmentation," in *International Conference on Machine Learning*, PMLR, pp. 1528–1537.

[14] Kuchnik, M. and V. Smith (2018) "Efficient augmentation via data subsampling," *arXiv preprint arXiv:1810.05222*.

[15] Chan, T. M. (1996) "Output-sensitive results on convex hulls, extreme points, and related problems," *Discrete & Computational Geometry*, **16**(4), pp. 369–387.

[16] Clarkson, K. L. (1994) "More output-sensitive geometric algorithms," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, IEEE, pp. 695–702.

[17] Matousek, J. (2013) *Lectures on discrete geometry*, vol. 212, Springer Science & Business Media.

[18] Kalai, G. (1992) "A subexponential randomized simplex algorithm," in *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pp. 475–482.

[19] Wainwright, M. J. (2019) *High-dimensional statistics: A non-asymptotic viewpoint*, vol. 48, Cambridge University Press.

[20] LeCun, Y. and C. Cortes (2010) "MNIST handwritten digit database," .
URL `http://yann.lecun.com/exdb/mnist/`

[21] Xiao, H., K. Rasul, and R. Vollgraf (2017), "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," `1708.07747`.

[22] Krizhevsky, A., V. Nair, and G. Hinton "CIFAR-10 (Canadian Institute for Advanced Research)," .
URL `http://www.cs.toronto.edu/~kriz/cifar.html`

[23] ——— "CIFAR-100 (Canadian Institute for Advanced Research)," .
URL `http://www.cs.toronto.edu/~kriz/cifar.html`