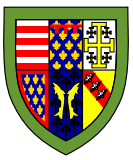
UNIVERSITY OF CAMBRIDGE

Generating Sets of Diverse Models for Ensembling

Vivek Palaniappan



Queens'

Final Report submitted for Part IIB Engineering Project

Abstract

Machine learning is becoming an integral part of everyday life, and with that, the complexity of the models used is increasing exponentially. Ensemble methods have always outperformed individual models by combining a diverse set of models to average out the errors of any individual model.

However, with the increasing complexity of models being used, for example AlexNet and VGGs, constructing ensembles with them as base learners has become a difficult problem. Existing methods are unable to generate enough useful diversity between the base models to make ensemble learning effective in the complex model paradigm. In this project, we explore how we can effectively generate sets of diverse models that allow us to improve ensemble performance.

First, we justify the need for this by theoretically proving that (a) diversity between base models directly contributes to ensemble loss via the Bias-Variance-Covariance (BVC) decomposition, (b) increasing diversity between base models reduces the Rademacher complexity of the ensemble, which gives a tighter bound on the generalisation error of the ensemble.

Motivated by the importance of base model diversity in ensembling, we explore an existing diversity generation method: Local Independence Training (LIT). During experimental evaluation of LIT, we notice issues with scaling between the loss function components, and show that the assumption of flat Euclidean input space breaks down for datasets such as images with intrinsic curvature.

We propose an improvement to LIT, which we call Manifold LIT, that uses the encoder function of a Deep Generative Model (DGM), such as a Variational Autoencoder (VAE), to transform the input space gradients used in LIT to compute the diversity loss, into gradients on the data manifold to compute a better calibrated diversity loss. Using Bagging, AdaBoost and Negative Correlation Learning (NCL) as benchmark ensembling methods, we perform experimental evaluation of Manifold LIT on MNIST and CIFAR-10, which shows that Manifold LIT outperforms both benchmarks and LIT on accuracy, area under ROC curves (AUC) and F1 scores.

Given the dependence of Manifold LIT on the gradients of a DGM that models the data manifold, we investigate the sensitivity of our method to variations in the parameters of the DGM. We derive theoretical bounds on the error that arises from variations in the Jacobians of the encoder function and DGM parameters. Using a Sobolev space analysis, we are able to show that if our DGM is trained appropriately, we can bound the perturbations on the gradients computed with the Sobolev norms of the encoder functions, which are reduced when the DGM learns a smoother encoder function. We empirically check this by adding Gaussian noise of varying variance to DGM parameters and show that the l_2 norm of the difference in manifold gradients remains small for several orders of magnitude of noise variance.

Although Manifold LIT allows us to generate diversity in base models, we are still unable to explore what this diversity means in terms of base model predictions, and probe such areas of diversity. So, we propose a novel framework, GeValDi (Generative Validation

of Discriminative Models), where we use the latent space of a DGM to find maximally diverse samples (MDS) where a set of base models predictions differ the most in terms of some divergence metric. We show that these MDS points represent regions where the ensemble is less confident and hence less accurate in, and by clustering MDS points, we are able to characterise regions in the data manifold where ensemble is least confident in. Therefore, GeValDi allows us to not only evaluate the diversity between the base models in terms of the divergence measure, but also generate regions of uncertainty for ensembles.

Motivated by the ability of MDS points to characterise regions of uncertainty of an ensemble, we propose a novel ensemble training procedure, inspired by the Active Learning literature, called Latent Divergence Training (LDT). Here, we train a set of base models on training data batches to maximise log-loss and Jensen-Shannon (JS) divergence, and use MDS points generated by our GeValDi algorithm to further maximise log-loss. So, MDS becomes active learning samples that are customised to an ensemble to improve its performance. The key to this is being able to either use a conditional-VAE/GAN or an oracle to generate true labels for our MDS points.

From experimental evaluation on CIFAR-10, we show that LDT outperforms Manifold LIT and all benchmarks, and manages to achieve high performance with even smaller ensemble size compared to the benchmarks. Finally, we perform a large-scale experiment on ImageNet, where we use a pre-trained SqueezeNet as the base model, with 58% top 1 accuracy, and show that the benchmark methods fail to introduce enough useful diversity to improve the ensemble performance, whilst LDT is able to improve the performance of the ensemble to 65%. This highlights the power encouraging diversity via JS divergence and using MDS points as additional training points. We conclude by discussing the dependence of LDT on the DGM parameter variations, and show that for noise variance up to the same order of magnitude of the DGM parameters, we have stable latent space paths and log-loss computation.

This project leaves several directions of further work

- Can we relate our JS divergence loss to distance measures between base learners, so that we can use our generalisation bound to get useful results on practical generalisation loss ranges?
- Can we improve the computational speed of LDT by adaptively scheduling the generation of MDS points, and using the clusters generated by MDS points as proxies for the MDS points?
- How does LDT perform when we use more cutting-edge ImageNet models?
- How does LDT perform for regression tasks?
- Can we apply GeValDi to model selection?
  - Can we use Human Subject Experiments (HSE) to choose between models by their MDS points?
  - Can we reconstruct the selection criteria used by humans by constructing a new loss function?
- Can we tune the ensemble base model weights to optimise some secondary objective that enhances the ensemble performance?

Declaration

A part of this project, namely Section 3.2, has been submitted to International Conference of Representation Learning (ICLR) 2023 Workshop in Trustworthy Machine Learning (ML), and under the tiny-paper track, where it has been accepted as a poster paper for the workshop and a presentation paper for the ICLR 2023 Tiny Papers track. The references for both are found in [48, 49]. Furthermore, a version of this project will be submitted to Association for the Advancement of Artificial Intelligence (AAAI) 2024 on August 15 2023.

I hereby declare that except where specific reference is made to the work of others, the contents of this report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This report is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and in Acknowledgements. This report contains fewer than 50 pages including footnotes, figures, tables, equations, appendices, and references.

Acknowledgements

I am truly grateful to my supervisor, Dr Adrian Weller, who has helped me tremendously in this project, always effectively probing my mind to help me generate new directions for the project and to refine existing directions in the project. The meetings I had with him were very helpful in formulating and evaluating this project, and I am very grateful for his insights.

I would also like to thank Umang Bhatt and Matthew Ashman, Dr Weller's PhD students, who have helped me immensely throughout the year. From brainstorming sessions at Bould Brothers, to coding sessions across the continent, their input has been vital in the output of this project.

Lastly, I would like to thank Katherine Collins, Dr Weller's PhD student, who assisted me in the writing of the paper submitted to ICLR 2023, which led to some of the ideas in this project as well.

Contents

1 Introduction

1.1 Motivation

In the rapidly evolving world of machine learning, the complexity of models used for decision-making and prediction tasks has increased significantly. Deep neural networks and other complex models have demonstrated remarkable capabilities across domains like computer vision and natural language processing. However, as model complexity grows, so do the challenges related to performance, robustness, and generalisability.

Ensemble methods have emerged as powerful techniques to address these challenges by harnessing the collective intelligence of multiple base models. By combining predictions from diverse models, ensembles often achieve superior performance, enhance robustness, and mitigate overfitting. However, as base models become more complex, traditional ensemble methods face limitations in generating sufficient model diversity.

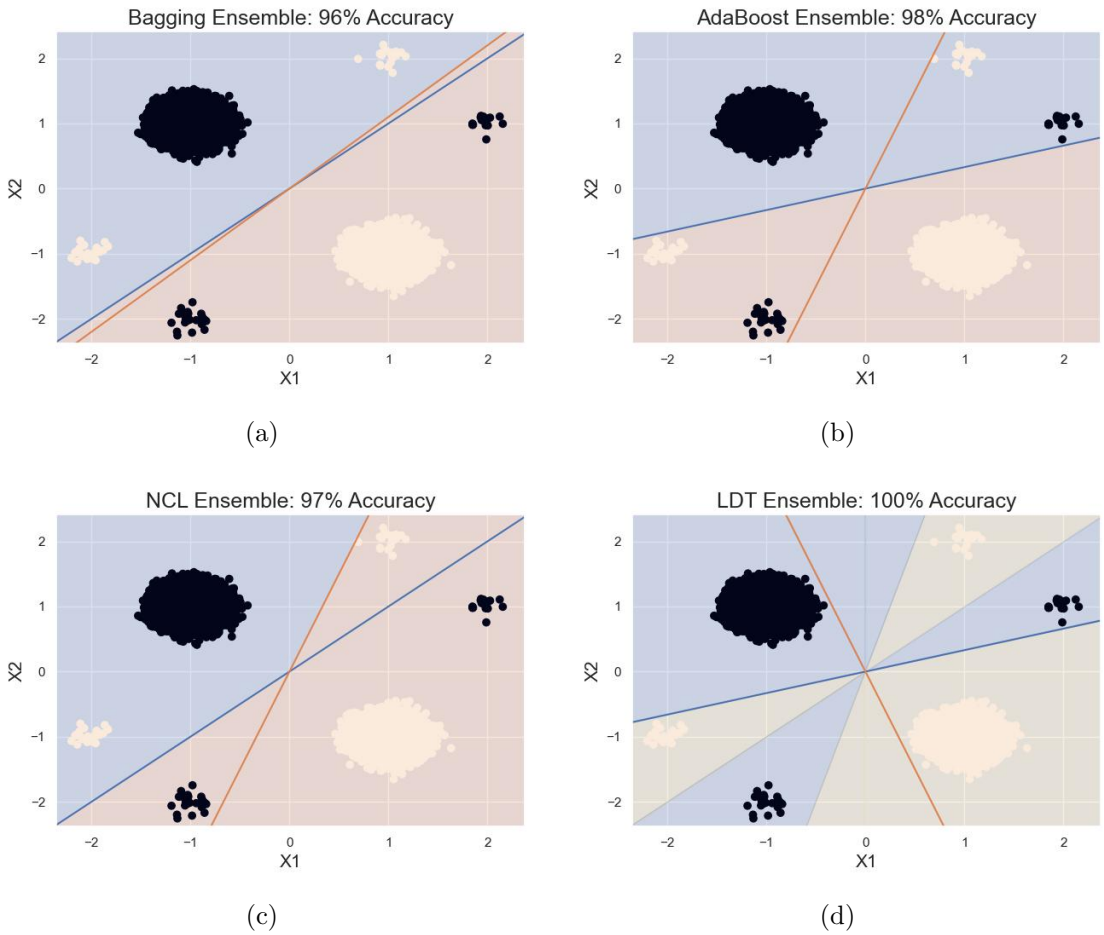
Generating sets of diverse models become increasingly challenging with growing base model complexity. Traditional ensemble techniques like bagging, boosting, and negative correlation learning may not ensure adequate diversity for large-scale models. Additionally, existing methods for measuring diversity in ensembles have limitations in capturing its true extent and impact on performance.

This project addresses these challenges by exploring novel techniques to generate diverse models and evaluate their impact on ensemble performance. We analyze theoretical motivations for diversity in ensembles, limitations of existing diversity generation methods, propose new approaches to enhance model diversity, and experimentally evaluate their effectiveness on real-world datasets.

1.2 Outline of Report

In this report, we develop tools to generate diverse sets of models for ensembling.

- In Chapter 2, we explore the theoretical relationship between ensemble performance and base model diversity and complexity to show why being able to generate diverse sets of models is critical for ensembling.
- In Section 2.3, we highlight the issues with current approaches to diversity measurement and diverse model generation to illustrate that they break down for the large-scale models.
- In Section 3.1, we introduce Local Independence Training (LIT) [55], where diversity is explicitly optimised in training. After illustrating scaling and gradient issues with the original LIT method, in Section 3.1.5 we propose a novel improvement, Manifold LIT, that uses a Deep Generative Model (DGM) to compute the diversity metric in the data manifold to improve diversity optimisation.



(a) (b) (c) (d)

Figure 1.1: (a) Bagging, (b) AdaBoost, (c) Negative Correlation Learning (NCL), (d) Latent Divergence Training classifiers on Toy Dataset

- In Section 3.2, we introduce GeValDi (Generative Validation of Discriminative Models), a novel method to probe the differences between models within an ensemble by exploring the latent space of a DGM that models the data manifold. Using this, we explore the diversity of base models generated by our ensembles, and construct a diversity metric in Section 3.2.4 that correlates with ensemble generalisation performance.
- Motivated by the diversity metric, in Section 3.3, we directly train on this metric on top of log-loss, whilst using maximally diverse samples obtained in GeValDi as additional training samples. We define this method to be Latent Divergence Training (LDT), and we show that the performance exceeds current methods, especially in large-scale datasets. In particular, in ImageNet, we are able to use pre-trained classifiers as base models and improve significantly on their performance. Figure 1.1 gives a summary of the methods on a toy dataset, showing that only the LDT ensemble is able to perfectly classify the dataset. This proves the value of this project.

2 Background: Generation of Diverse Models

Enhancing the robustness and generalisability of ensemble models is contingent upon generating diverse sets of models. For ensembles to be successful, their constituent models need to make distinct errors on unseen data, a concept commonly referred to as ensemble "diversity" [24, 14]. However, the measurement, definition, and potential training of diversity remain open questions subject to ongoing research [39, 8].

Implicit diversity methods introduce stochasticity in the interaction between models and data. For example, random resampling of training examples can induce diversity in bagging [5], as can randomly selecting input feature subsets in random forests [6]. Stochasticity can also be introduced by retraining models with different initialisations [32].

Explicit diversity techniques, such as boosting [58], sequentially modify each model's objective function to specialize in previous models' errors. Other methods explicitly encourage diversity using proxies such as negative correlation learning [43] and amended cross-entropy [62].

2.1 Model Diversity vs Ensemble Error

In this section, we explore the crucial relationship between the diversity of models within an ensemble and the ensemble's overall error. Intuitively, it is expected that diverse models, which make different errors on unseen data, would lead to improved performance when combined, as they complement each other's strengths and compensate for their respective weaknesses. This relationship is fundamental in understanding how to effectively construct ensembles that achieve superior generalization performance compared to single model approaches.

Numerous studies have investigated the relationship between model diversity and ensemble error, providing both theoretical and empirical evidence that supports the importance of diversity in ensemble learning. For instance, [24] demonstrated that an ensemble's error could be reduced when the constituent models are both accurate and diverse, while [14, 8] performed an experimental comparison of different ensemble methods, highlighting the significance of diversity in achieving better performance. Moreover, [39, 54] presented an extensive analysis of diversity measures in classifier ensembles and their connection to ensemble accuracy. We focus on the intuitive relationship of diversity and performance by decomposing the ensemble loss.

Regression losses allow easy decompositions, where we can understand the role of diversity of base learners in the ensemble performance by considering two decompositions of the squared loss: the ambiguity decomposition and the Bias-Variance-Covariance decomposition of the ensemble.

The ambiguity decomposition [35] proves that *at a single data point, the quadratic loss of the ensemble estimator is guaranteed to be less than or equal to the average quadratic*

error of the base models:

$$(f_{\text{ens}} - \mathbf{y})^2 = \sum_{i=1}^{M} w_i (f_i - \mathbf{y})^2 - \sum_{i=1}^{M} w_i (f_i - f_{\text{ens}})^2 \tag{2.1}$$

where $f_{\text{ens}}(\mathbf{x}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x})$ is the ensemble with base estimator $f_i(\mathbf{x})$ and weights w_i, and $\mathbf{y}$ is the true target variable. The first term is the weighted average error of the base models, and the second term is the *ambiguity* term. The ambiguity term measures the amount of variability among the base model predictions. Since this is always positive, the larger the ambiguity term, the more the negative correlation is between base models, and consequently, the more the ensemble error reduction. However, increasing the variability of the base models will also increase the first term of the decomposition, which highlights the trade-off between increasing diversity and reducing base model error to achieve lowest ensemble error.

The bias-variance decomposition for quadratic loss states that the generalisation error of an estimator can be broken down into two components: bias and variance. These two usually work in opposition to one other: attempts to reduce the bias component will cause an increase in variance, and vice-versa. The decomposition is as follows:

$$\mathbb{E}[(f - \mathbb{E}[\mathbf{y}])^2] = \mathbb{E}[(f - \mathbb{E}[f])^2] + (\mathbb{E}[f] - \mathbb{E}[\mathbf{y}])^2 \tag{2.2}$$

$$\text{Loss}(f) = \text{Variance}(f) + \text{Bias}(f)^2 \tag{2.3}$$

where $f(\mathbf{x})$ is our estimator, $\mathbf{y}$ is the target variable, $\mathbb{E}[\mathbf{y}]$ is the expected value of the target variable, given the noise in data. In the case of a convex combined ensemble, $f(\mathbf{x}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x})$ such that $\sum_{i=1}^{M} w_i = 1$, the variance term breaks down even further, leading to the Bias-Variance-Covariance breakdown. Bias_a, Variance_a and Covariance_a is the averaged bias, variance and covariance of the ensemble members, defined as:

$$\text{Bias}_a = \frac{1}{M} \sum_{i=1}^{M} w_i \mathbb{E}[f_i] - \mathbb{E}[\mathbf{y}] \tag{2.4}$$

$$\text{Variance}_a = \frac{1}{M} \sum_{i=1}^{M} w_i^2 \mathbb{E}[(f_i - \mathbb{E}[f_i])^2] \tag{2.5}$$

$$\text{Covariance}_a = \frac{1}{M(M-1)} \sum_{i=1}^{M} \sum_{j \neq i}^{M} w_i w_j \mathbb{E}[(f_i - \mathbb{E}[f_i])(f_j - \mathbb{E}[f_j])] \tag{2.6}$$

Using this, and some elementary mathematics omitted due to space constraints, we obtain the bias-variance-covariance decomposition (BVC) of our squared loss:

$$\mathbb{E}\left[\left(\left(\frac{1}{M} \sum_{i=1}^{M} w_i f_i\right) - \mathbb{E}[\mathbf{y}]\right)^2\right] = \text{Bias}_a^2 + \frac{1}{M}\text{Variance}_a + \left(1 - \frac{1}{M}\right)\text{Covariance}_a \tag{2.7}$$

We can see that the squared loss of an ensemble of estimators depends critically on the amount of error correlation between base estimators, quantified by the covariance term.

We would ideally like to decrease the covariance, without causing any increases in the bias or variance terms. It is also worth noting that while bias and variance are constrained to be positive-valued, the covariance term can be negative.

Our regression analysis can be thought of as a generalisation of classification analysis by constraining the number of possible predictions to a finite number (number of classes). Furthermore, we can also note the relationship between the correlation of model hard predictions, $\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$, and the discriminative output of the models, $p(\mathbf{y}|\mathbf{x})$, by considering various cases. This is useful to generalise BVC decomposition to any setup with discriminative models. Let's first consider the two model case, where we denote $\mathbf{y}'_1$ and $\mathbf{y}'_2$ the hard prediction, and $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$ the discriminative model outputs for model 1 and 2 respectively:

- If there is a high correlation between the hard classifications, $\mathbf{y}'_1$ and $\mathbf{y}'_2$, it means that the models tend to agree on their predictions. This can be due to two reasons:
  - Both models have similar discriminative outputs $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$ for all $\mathbf{x}$, meaning they tend to assign similar probabilities to the same classes.
  - Both models have slightly different discriminative outputs $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$, but the maximium probability class is the same, and the minor differences are in the other class probabilities.
- If there is a low correlation between the hard classifications, $\mathbf{y}'_1$ and $\mathbf{y}'_2$, it means that the models tend to disagree on their predictions. This can be due to two reasons:
  - The discriminative outputs $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$ are significantly different for most $\mathbf{x}$, meaning they tend to assign very different probabilities to the classes. So, the maximum probability class is also different.
  - Both models have similar discriminative outputs $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$, but with low confidence in all classes. So, even small perturbations to the probability predictions can change the maximum probability class, making the hard classifications different.

Considering this intuition, we see that in general, when the difference between $p_1(\mathbf{y}|\mathbf{x})$ and $p_2(\mathbf{y}|\mathbf{x})$ is large, in some probability distributional sense, we get lower rates of correlation between their hard predictions. This motivates us to design diversity measures that measure the difference between the discriminative outputs of models so that by training our models to maximise this difference, we minimise correlation between the predictions, which according to our BVC decomposition, should improve the ensemble performance.

2.2 Model Complexity vs Ensemble Error

We have demonstrated that ensembling is most effective when the base learners are as "negatively correlated" from each other as possible, considering both their ambiguity and loss covariance. However, it is also crucial to explore the relationship between ensemble error and model complexity. This is particularly relevant for real-world applications, where

the goal is often to deploy the best-performing set of models, which are typically large-scale deep neural networks [42, 22], and ensembles of such large-scale deep neural networks [23, 36, 67].

We first start with how model complexity relates to its generalisation error. For this, we first consider a discriminative classifier, $f(\mathbf{x}) = p(\mathbf{y} = 1|\mathbf{x})$. We now use the generalisation bound that relates the expected generalisation error of a model, $L_G(f)$, the sample error of a model, $L_S(f)$, and the Rademacher complexity, $\mathcal{R}_n(\mathcal{F})$, of the model from function class $\mathcal{F}$ [60]. Let $\hat{\mathcal{R}}_n(\mathcal{F})$ be the *empirical* Rademacher complexity of function class $\mathcal{F}$, defined as

$$\hat{\mathcal{R}}_n(\mathcal{F}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \sigma_i f(\mathbf{x}_i) \right) \right] \tag{2.8}$$

where $\{\sigma_i\}_{i=1}^n$ are independent random variables uniformly chosen from $\{-1, 1\}$, called the Rademacher variables. Note that we can extend this analysis to multi-class by just allowing Rademacher variables to take c different values, and that we can equivalently use Gaussian complexity, which replaces the Rademacher variables with independent unit Gaussians, which is more appropriate for regression analysis. Gaussian and Rademacher complexities are known to be equivalent up to logarithmic factors. Using the Hoeffding inequality [29], for any $\epsilon > 0$, we get that

$$P(|L_G(f) - L_S(f)| > \epsilon) \leq 2 \exp \left(- \frac{2N\epsilon^2}{\hat{\mathcal{R}}(\mathcal{F})^2} \right) \tag{2.9}$$

We also get a weaker bound on the generalisation error of a model by using Mcdiarmid Inequality [46], such that for any δ with $0 \leq \delta \leq 1$, with probability $1 - \delta$, the following bound holds

$$L_G(f) \leq L_S(f) + \hat{\mathcal{R}}_n(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{n}} \tag{2.10}$$

These two bounds give us some idea on how model complexity affect the model generalisation error. As the model complexity increases, the bound we can enforce on its generalisation error is weaker. In other words, as model complexity increases, we are less certain that the model generalisation error will be close to the sample error. This poses an issue because in reality, we only have some finite samples to estimate the error of the models on, and model complexity therefore becomes an important consideration.

For ensembles, we can make the exact same argument by considering the complexity of an ensemble. It is trivial to show that for a convex sum of base estimators $f(\mathbf{x}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x})$ from the same function class $f_i \in \mathcal{F}$, the Rademacher complexity of the ensemble is equal to the Rademacher complexity of the base estimator $\mathcal{R}_n(\mathcal{F}) = \mathcal{R}_n(\text{conv}(\mathcal{F})) = \mathcal{R}_n(\mathcal{F})$, where $\text{conv}(\mathcal{F})$ denotes the convex hull of the base model function classes. However, we can form a more useful bound if we consider the margin-based generalisation bound for ensembles, derived in [40]. Let margin be defined as $\rho_f(\mathbf{x}, \mathbf{y}) = f(\mathbf{y}|\mathbf{x}) - \max_{\mathbf{y}' \neq \mathbf{y}} f(\mathbf{x}, \mathbf{y}')$, where the classifier correctly classifies if $f(\mathbf{y}|\mathbf{x}) > \rho_f(\mathbf{x}, \mathbf{y})$. In a multi-class classification problem with c labels, consider M classes of functions $\mathcal{F}_1, ..., \mathcal{F}_M$ from which we get our base classifiers $f_1, ..., f_M$, and the ensemble

function class $\mathcal{H} = \text{conv}(\cup_{i=1}^{M} \mathcal{F}_i)$, that is the class of functions formed by the convex sum of base classifiers, $h = \sum_{i=1}^{M} w_i f_i$. Now, for $\rho > 0$, we can define ensemble generalisation error $L_G(h) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[1(\rho_h(\mathbf{x},\mathbf{y}) \leq 0)]$, ρ-margin error $L_{G,\rho}(h) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[1(\rho_h(\mathbf{x},\mathbf{y}) \leq \rho)]$ and its empirical margin error, $L_{S,\rho}(h) = \mathbb{E}_{S\sim p(\mathbf{x},\mathbf{y})}[1(\rho_h(\mathbf{x},\mathbf{y}) \leq \rho)]$, for a sample $S \sim p(\mathbf{x},\mathbf{y})$. Given this, assuming $p > 0$, fixing $\rho > 0$, for every $\delta > 0$, with probability $1 - \delta$ over the choice of sample $S \sim p(\mathbf{x},\mathbf{y})$ of size n, the following quantity holds for all $f_i \in \mathcal{F}_i$ and $h = \sum_{i=1}^{M} w_i f_i \in \mathcal{H}$:

$$L_G(h) \leq L_{S,\rho}(h) + \frac{8c}{\rho} \sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i) + O\left(\sqrt{\frac{\log p}{\rho^2 n} \log\left[\frac{\rho^2 c^2 n}{4 \log p}\right]}\right) \tag{2.11}$$

We extend this further by considering how base model diversity affects the Rademacher complexities of the base model function classes and ensemble function classes. Consider the case where $\mathcal{H}$ is fixed, and we have a constrained optimisation problem as our learning algorithm. We want to minimise our loss, $L(h)$, such that the functional distance between the base models is at least ϵ.

$$h^* = \arg\max_{h\in\mathcal{H}} L_S(h) \quad \text{s.t.} \quad D(\{f_i\}_{i=1}^{M}) \geq \epsilon \tag{2.12}$$

$$D(\{f_i\}_{i=1}^{M}) = \min\{d(f_j, f_k); f_j \in \mathcal{F}_j, f_k \in \mathcal{F}_k, j, k = 1, ..., M\} \tag{2.13}$$

for some distance metric between functions $d(f_j, f_k)$ (for example L^2 norm). This is motivated by the fact that most diversity generation methods can be interpreted as generating base learners such that the pairwise distance (in a functional notion) is increased [39]. Furthermore, we can rewrite the constrained optimisation as unconstrained optimisation using Karush-Kuhn-Tucker (KKT) conditions [37] as

$$h^* = \arg\max_{h\in\mathcal{H}} L_S(h) + \lambda_d(D(\{f_i\}_{i=1}^{M}) - \epsilon) \tag{2.14}$$

where λ_d is essentially the KKT multiplier, which we can interpret as the trade-off between our diversity and loss. This is essentially the form of explicit diversity generation techniques we use in this project. Defining $\{f^{(\epsilon)}\}_{i=1}^{M} \in \{f_i; D(\{f_i\}_{i=1}^{M}) \geq \epsilon, i = 1, ..., M\} = \{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$ as the set of base learners with at least ϵ diversity, we propose the following

Proposition 1. *Suppose we have two sets of base models $\{f^{(\epsilon)}\}_{i=1}^{M}$ and $\{f^{(\epsilon')}\}_{i=1}^{M}$. If $\epsilon' > \epsilon$, then $\mathcal{F}_i^{(\epsilon')}$ is a subset of $\mathcal{F}_i^{(\epsilon)}$ and the cardinality of the set $\mathcal{F}_i^{(\epsilon')}$ is at most as big as the cardinality of the set $\mathcal{F}_i^{(\epsilon)}$. In other words, if $\epsilon' > \epsilon$, then $\mathcal{F}_i^{(\epsilon')} \subseteq \mathcal{F}_i^{(\epsilon)}$ and $|\mathcal{F}_i^{(\epsilon')}| \leq |\mathcal{F}_i^{(\epsilon)}|$ for $i = 1, ..., M$.*

Proof. We prove this by construction. We know that by definition, for any $\{f_i^{(\epsilon')}\}_{i=1}^{M} \in \{\mathcal{F}_i^{(\epsilon')}\}_{i=1}^{M}$, $D(\{f_i^{(\epsilon')}\}_{i=1}^{M}) \geq \epsilon' > \epsilon$. However, the converse, that for any $\{f_i^{(\epsilon)}\}_{i=1}^{M} \in \{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$, $D(\{f_i^{(\epsilon)}\}_{i=1}^{M}) \geq \epsilon < \epsilon'$ shows that there is a subset of functions $\{f_i^{(\epsilon,\epsilon')}\}_{i=1}^{M} \in \{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$, for which $\epsilon' > D(\{f_i^{(\epsilon,\epsilon')}\}_{i=1}^{M}) \geq \epsilon$. So, we have a subset of functions that are in the set $\{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$ but not in $\{\mathcal{F}_i^{(\epsilon')}\}_{i=1}^{M}$, but all functions in $\{\mathcal{F}_i^{(\epsilon')}\}_{i=1}^{M}$ are also in $\{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$, which shows that $\mathcal{F}_i^{(\epsilon')}$ is a subset of $\mathcal{F}_i^{(\epsilon)}$ and the cardinality of sets $\{\mathcal{F}_i^{(\epsilon')}\}_{i=1}^{M}$ is at most as large as the cardinality of sets $\{\mathcal{F}_i^{(\epsilon)}\}_{i=1}^{M}$. □

Since we know that for a subset $\mathcal{F}' \subseteq \mathcal{F}$, $\hat{\mathcal{R}}_n(\mathcal{F}') \leq \hat{\mathcal{R}}_n(\mathcal{F})$, we can apply our constrained diversity optimisation to propose that

Proposition 2. *Consider two ensembles $h^{(\epsilon)}, h^{(\epsilon')} \in \mathcal{H}$, constructed from base models $\{f^{(\epsilon)}\}_{i=1}^M$ and $\{f^{(\epsilon')}\}_{i=1}^M$ respectively. If $\epsilon' > \epsilon$ and $L_{S,\rho}(h^{(\epsilon')}) \leq L_{S,\rho}(h^{(\epsilon)})$, then*

$$L_{S,\rho}(h^{(\epsilon')}) + \frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon')}) \leq L_{S,\rho}(h^{(\epsilon)}) + \frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon)}) \tag{2.15}$$

which means the generalisation bound on $L_G(h^{(\epsilon')})$ is tighter than the generalisation bound on $L_G(h^{(\epsilon)})$. In other words, if we use a more diverse set of base learners, and we get an equal or lower sample margin loss in the more diverse ensemble, we can bound the generalisation loss of the more diverse ensemble tighter than we can bound the generalisation loss of the less diverse ensemble.

Proof. Using Proposition 1, we know that $\mathcal{F}_i^{(\epsilon')} \subseteq \mathcal{F}_i^{(\epsilon)}$ for $i = 1, ..., M$. Furthermore, we know that for any subset for any subset $\mathcal{F}' \subseteq \mathcal{F}$, $\mathcal{R}_n(\mathcal{F}') \leq \mathcal{R}_n(\mathcal{F})$. Combining the two, and using Equation (2.11), since we are performing a convex sum over the base learner Rademacher complexities, we get that

$$\hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon')}) \leq \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon)}) \quad i = 1, ..., M \tag{2.16}$$

$$\frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon')}) \leq \frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon)}) \tag{2.17}$$

$$L_{S,\rho}(h^{(\epsilon')}) \leq L_{S,\rho}(h^{(\epsilon)}) \tag{2.18}$$

$$L_{S,\rho}(h^{(\epsilon')}) + \frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon')}) \leq L_{S,\rho}(h^{(\epsilon)}) + \frac{8c}{\rho}\sum_{i=1}^{M} w_i \hat{\mathcal{R}}_n(\mathcal{F}_i^{(\epsilon)}) \tag{2.19}$$

which concludes the proof. □

Proposition 2 allows us to show that using the abstract notion of diversity, defined by increasing the functional distance between base model function classes, we have a bound that states that the generalisation loss of the more diverse ensemble is more tightly bounded than the generalisation loss of the less diverse ensemble. This illustrates the importance of diversity in ensemble base learners.

2.3 Issues with Diversity Measurement and Generation

While there are several existing methods of measuring model diversity [62, 39, 8] and training for model diversity [24, 14, 5, 6, 58, 43], the current methods suffer from a few issues that we will elaborate in this section.

Measuring diversity suffers from the following issues:

- Dependency on training dataset: Many diversity measures are based on analysing the predictions made by the ensemble models on the training dataset. However,

the diversity observed on the training dataset may not necessarily generalise well to unseen data or reflect the ensemble performance on the test dataset. This can limit the effectiveness of diversity measures in predicting the ensemble performance in real-world datasets [8, 18].

- Lack of interpretability: Some diversity measures used in ensemble learning are not easily interpretable or intuitive. They often rely on specific assumptions about the data and models, which makes it challenging to usefully understand what the diversity measures mean and to effectively apply them in ensemble learning [20, 39].
- Limited scope: Many existing diversity measures focus only on caputring the differences in predictions made by the ensemble models on the training set. These measures may not fully capture the true diversity of the models in terms of their underlying behvaiours, decision boundaries, or uncertainty estimation. As a result, they may not provide a comprehensive understanding of the ensemble model diversity [8, 20].
- Lack of consensus: There is no widely accepted or standardized measure of diversity in the field of ensemble learning. Different practitioners may use different diversity measures based on their specific goals. This lack of consensus makes it challenging to compare and evaluate the diversity of different ensemble models in a consistent manner [20, 18].

Similarly, current methods to train for diversity suffer from the following issues:

- Dependency on base models: Many diversity generation methods rely on the diversity already present in the base models. If the base models themselves lack diversity, the ensemble may not achieve the desired level of diversity [18].
- Sensitivity of base model performance: Diversity generation methods can be sensitive to the performance of base models. For example, boosting methods rely heavily on the misclassified instances by the base models, which can be problematic if the base models have low accuracy or are prone to making similar errors. A similar issue also arises if the base models are overfit to the training data because the boosting will amplify this effect and impact generalisation performance [14].
- Limited exploration of solution space: Some diversity generation methods, such as bagging, focus on introducing diversity through random sampling or perturbations of the training data. While this can lead to some diversity, it may not fully explore the solution space and miss out on alternative solutions that could improve ensemble performance [20].
- Lack of diversity control: Existing methods lack fine-grained control over the level and type of diversity generated. In some cases, the diversity may be insufficient or excessive for the specific problem or dataset, leading to sub-optimal ensemble performance [8].
- Lack of theoretical guarantees: While many diversity generation methods show empirical success, they lack theoretical guarantees regarding the impact of diversity on ensemble performance [39].

3 Generating Diverse Models

3.1 Local Independence Training

In this section we introduce a method for generating diverse sets of models where local independence is optimised as part of the loss function. Local Independence Training (LIT) allows us to generate arbitrary sets of models that are trained to optimise both likelihood and local independence, giving us a useful method to generate diverse models. Note that the beginning of this section implements the method proposed in [56], and Section 3.1.5 improves upon the issues of the original method to implement an extended version of this algorithm.

Before introducing the notion of local independence, we need to introduce some necessary notation. We use $\mathbf{x}$ to denote D-dimensional inputs, which are supported over an input space $\Omega_{\mathbf{x}} \subseteq \mathbb{R}^D$. We use $\mathbf{y}$ to denote prediction targets in an output space $\Omega_{\mathbf{y}}$. We denote a prediction model as a function $f(\cdot;\theta) : \Omega_{\mathbf{x}} \to \Omega_{\mathbf{y}}$, that is parameterised by θ, that estimates $\mathbf{y}$ from $\mathbf{x}$. In addition to this, we suppose a joint distribution over inputs and targets $p(\mathbf{x}, \mathbf{y})$ and a distribution $p(\mathbf{y}|f(\mathbf{x};\theta))$ that quantifies the likelihood of the observed target given the model prediction. This gives us the expected log-likelihood $\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\log p(\mathbf{y}|f(\mathbf{x};\theta))]$ that will be part of our loss function.

3.1.1 Local Independence

We introduce a measure of model diversity that quantifies how differently two models generalize over small patches of the data manifold $\Omega_{\mathbf{x}}$. Let ϵ be a small number, $N_\epsilon(\mathbf{x})$ be the ϵ-neighbourhood of $\mathbf{x}$ on the data manifold, which is the intersection of an ϵ-ball centered at $\mathbf{x}$ in the input space, $\mathcal{B}_\epsilon(\mathbf{x}) \subset \mathbb{R}^D$, and the data manifold: $N_\epsilon(\mathbf{x}) = \mathcal{B}_\epsilon(\mathbf{x}) \cap \Omega_{\mathbf{x}}$. We capture the notion of generalization difference on a small neighborhood of $\mathbf{x}$ through an intuitive geometric condition: we say that two functions f and g generalize maximally differently at $\mathbf{x}$ if f is invariant in the direction of the greatest change in g (or vice versa) within an ϵ-neighborhood around $\mathbf{x}$. That is,

$$f(\mathbf{x}) = f(\mathbf{x}_{g_{\max}}), \text{ for all } \epsilon' < \epsilon \tag{3.1}$$

where $\mathbf{x}_{g_{\max}} = \arg\max_{\mathbf{x}' \in N_{\epsilon'}(\mathbf{x})} g(\mathbf{x}')$. In other words, perturbing $\mathbf{x}$ by small amounts to increase g inside N_ϵ does not change the value of f. If a choice of ϵ exists to satisfy Equation (3.1), we say that f is *locally independent* at $\mathbf{x}$. We call f and g *locally independent* without qualification if for every $\mathbf{x} \in \Omega_{\mathbf{x}}$, the functions f and g are locally independent at $\mathbf{x}$ for some choice of ϵ. Note that to define the right-hand side expression of Equation (3.1), we assume that the gradient of g is not zero at $\mathbf{x}$ and that ϵ is chosen to be small enough that g is convex or concave over $N_\epsilon(\mathbf{x})$.

In the case that f and g are classifiers, local independence implies a kind of dissimilarity between their decision boundaries. For example, if f and g are linear and the data

manifold is Euclidean, then f and g are locally independent if and only if their decision boundaries are orthogonal. This definition motivates the formulation of a diversity measure, $\text{IndepErr}(f, g)$, quantifying how far f and g are from being locally independent:

$$\text{IndepErr}(f, g) = \mathbb{E}[(f(\mathbf{x}_{g_{\max}}) - f(\mathbf{x}))^2] \tag{3.2}$$

Computing IndepErr exactly, however, is challenging, because it requires an inner optimisation of g. We can, however, approximate it for fixed small ϵ with projected gradient descent as in adversarial training, though that is computationally intensive. Alternatively, if we let $\epsilon \to 0$, we can approximate $\mathbf{x}_{g_{\max}}$ by a fairly simple equation that only needs to compute ∇g once per $\mathbf{x}$. In particular, we observe that if we assume g is at least once differentiable, and that $N_\epsilon(\mathbf{x}) \approx \mathcal{B}_\epsilon(\mathbf{x})$ (i.e. that the data manifold in the neighbourhood of $\mathbf{x}$ is Euclidean), we can make the following approximation as $\epsilon \to 0$

$$\mathbf{x}_{g_{\max}} \approx \mathbf{x} + \epsilon \nabla g(\mathbf{x}) \tag{3.3}$$

With similar smoothness assumptions on f, we can use the Taylor expansion to get

$$\begin{aligned} f(\mathbf{x}_{g_{\max}}) - f(\mathbf{x}) &\approx f(\mathbf{x} + \epsilon \nabla g(\mathbf{x})) - f(\mathbf{x}) && (3.4)\\ &= [f(\mathbf{x}) - \epsilon \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) + \mathcal{O}(\epsilon^2)] - f(\mathbf{x}) && (3.5)\\ &\approx \epsilon \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) && (3.6) \end{aligned}$$

In other words, the independence error between f and g is approximately equal to the dot product of their gradients $\nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x})$. Empirically, it is useful to normalise the dot product and work in terms of the cosine similarity of the gradients

$$\cos(\nabla f(\mathbf{x}), \nabla g(\mathbf{x})) = \frac{\nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x})}{||\nabla f(\mathbf{x})||_2 ||\nabla g(\mathbf{x})||_2} \in [-1, 1] \tag{3.7}$$

Note that for numerical stability, we add a small constant to the denominator of cosine similarity. This gives us the final form of the IndepErr, which we now call LIT Loss.

$$\text{LIT Loss} = \frac{\nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x})}{||\nabla f(\mathbf{x})||_2 ||\nabla g(\mathbf{x})||_2} \tag{3.8}$$

It is worth considering this LIT Loss from a statistical perspective. Consider sampling small perturbations $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I})$ and evaluating $\Delta f(\mathbf{x}) = f(\mathbf{x} + \epsilon) - f(\mathbf{x})$ and $\Delta g(\mathbf{x}) = g(\mathbf{x} + \epsilon) - g(\mathbf{x})$. As we let $\sigma \to 0$, these differences will approach $\epsilon \cdot \nabla f(\mathbf{x})$ and $\epsilon \cdot \nabla g(\mathbf{x})$, which are essentially 1D Gaussian random variables. We can compute the covariance and correlation between these random variables as follows:

$$\Delta f(\mathbf{x}) = f(\mathbf{x}+\epsilon) - f(\mathbf{x}) \approx \nabla f(\mathbf{x}) \cdot \epsilon \tag{3.9}$$
$$\Delta g(\mathbf{x}) = g(\mathbf{x}+\epsilon) - g(\mathbf{x}) \approx \nabla g(\mathbf{x}) \cdot \epsilon \tag{3.10}$$
$$\begin{aligned}\mathrm{Cov}(\Delta f(\mathbf{x}), \Delta g(\mathbf{x})) &\approx \mathbb{E}[(\nabla f(\mathbf{x}) \cdot \epsilon - \mathbb{E}[\nabla f(\mathbf{x}) \cdot \epsilon])(\nabla g(\mathbf{x}) \cdot \epsilon - \mathbb{E}[\nabla g(\mathbf{x}) \cdot \epsilon])] && (3.11)\\ &= \mathbb{E}[(\nabla f(\mathbf{x}) \cdot \epsilon)(\nabla g(\mathbf{x}) \cdot \epsilon)] && (3.12)\\ &= \mathbb{E}[\epsilon^2] \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) && (3.13)\\ &= \sigma^2 \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) && (3.14)\end{aligned}$$

Using the fact that $\mathrm{Var}(\epsilon) = \sigma^2$, we directly get that the correlation between $\Delta f(\mathbf{x})$ and $\Delta g(\mathbf{x})$ is

$$\begin{aligned}\mathrm{Corr}(\Delta f(\mathbf{x}), \Delta g(\mathbf{x})) &= \frac{\nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x})}{||\nabla f(\mathbf{x})||_2 ||\nabla g(\mathbf{x})||_2} && (3.15)\\ &= \cos(\nabla f(\mathbf{x}), \nabla g(\mathbf{x})) && (3.16)\end{aligned}$$

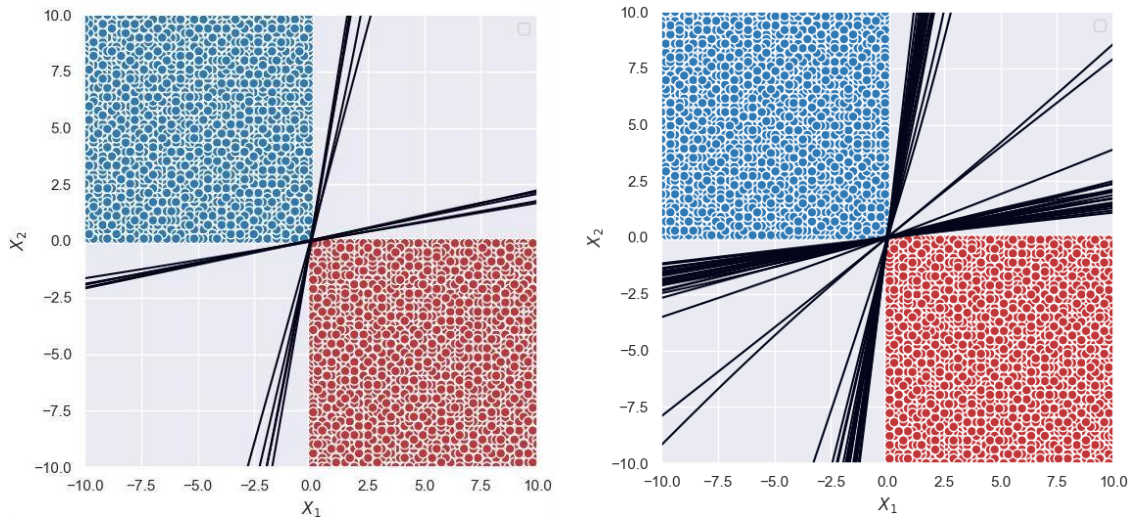
Given that our LIT Loss naturally drops out of the correlation between the gradients with random perturbations, it seems sensible to explore the distribution of this gradient perturbation itself, i.e. the distribution of $\Delta f(\mathbf{x})$ and $\Delta g(\mathbf{x})$. Since $\Delta f(\mathbf{x})$ is the dot product between a deterministic vector $\nabla f(\mathbf{x})$ and a Gaussian random variable ϵ, we get that $\Delta f(\mathbf{x})$ and $\Delta g(\mathbf{x})$ are both Gaussian random variables. Since we know that the entropy of Gaussian random variables is $H(X) = \frac{1}{2}\ln(2\pi e|\Sigma|)$, and $|\Sigma| = \mathrm{Var}(X)\mathrm{Var}(Y) - \mathrm{Cov}(X,Y)^2$ [11], we get that the mutual information between our gradient perturbations is

$$\begin{aligned}I(\Delta f(\mathbf{x}), \Delta g(\mathbf{x})) &= H(\Delta f(\mathbf{x})) + H(\Delta g(\mathbf{x})) - H(\Delta f(\mathbf{x}), \Delta g(\mathbf{x})) && (3.17)\\ &= \frac{1}{2}\ln\left(\frac{Var(\Delta f(\mathbf{x}))\mathrm{Var}(\Delta g(\mathbf{x}))}{\mathrm{Var}(\Delta f(\mathbf{x}))\mathrm{Var}(\Delta g(\mathbf{x})) - \mathrm{Cov}(\Delta f(\mathbf{x}), \Delta g(\mathbf{x}))^2}\right) && (3.18)\\ &= \frac{1}{2}\ln(1 - \mathrm{Corr}(\Delta f(\mathbf{x}), \Delta g(\mathbf{x}))^2) && (3.19)\end{aligned}$$

This shows us that when we have zero correlation between the local gradients of the models, it is equivalent to having statistical Independence between those model gradients for local perturbations under $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbb{I})$, giving a strong argument for LIT Loss as a diversity measure. Therefore, the final loss function for Local Independence Training is

$$\begin{aligned}\mathcal{L}(\{\theta_m\}) &= \sum_{m=1}^{M} \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[-\log p(\mathbf{y}|f(\mathbf{x};\theta_m))] && (3.20)\\ &\quad + \lambda_{LIT} \sum_{l\neq m}^{M} \mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla f(\mathbf{x};\theta_m), \nabla f(\mathbf{x};\theta_l))] && (3.21)\end{aligned}$$

where λ_{LIT} quantifies our trade-off between accuracy of the models and the diversity between the models.



(a) (b)

Figure 3.1: (a) 10, (b) 50, classifiers trained using the LIT Loss

3.1.2 Toy Example

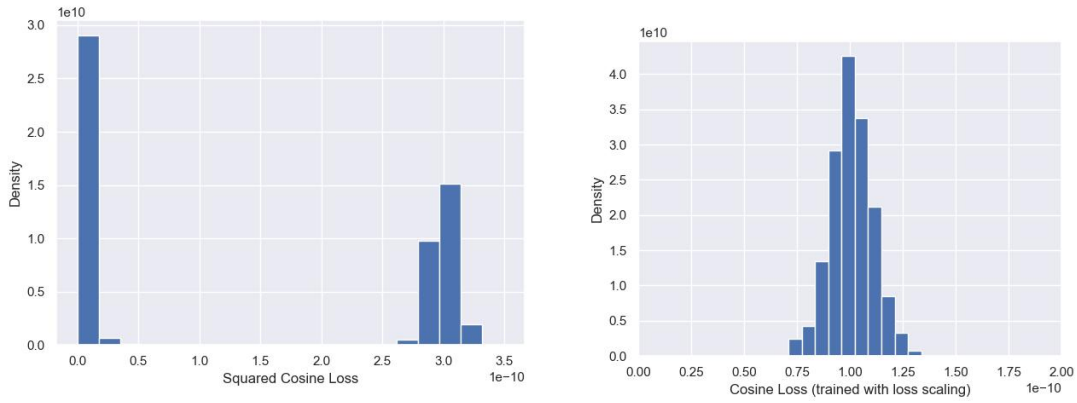
In order to understand the performance of the LIT Loss function, we run a toy experiment on a simple dataset: we let the top left quadrant be class 0, and bottom right quadrant be class 1. We fit (a) 10 and (b) 50 linear classifiers to this dataset, with a $\lambda_{LIT} = 10^{-4}$. The resulting classifiers are plotted in Figure 3.1.

Looking at the classifiers, we notice a few interesting properties. Firstly, we are able to generate an arbitrary number of classifiers with the same accuracy (logloss), and we observe visible differences between the decision boundaries of the classifiers. However, we notice that the degree of diversity (in terms of the angles between the decision boundaries) is variable across the set of classifiers. In particular, we observe that there is some clustering of the classifier boundaries to the functions $f(y|x) = \infty$ and $f(y|x) = 0$.

Given the clustering of classifiers and the fact that the models do not span the entire range of classifiers with 100% accuracy, which in this case is the set of models $f(y|x) = \{mx : m \in [0, \infty]\}$, we suspect that there is an issue with the distribution of diversity between the classifiers. In other words, the distribution of pairwise cosine loss terms between classifiers is not unimodally distributed like we would want it to be. We validate this by considering the distribution of cosine loss between all the pairs of classifiers.

3.1.3 Issues with Loss Scaling

By plotting a histogram of cosine loss in Figure 3.2, we notice a bimodal distribution. This is inline with what we observe in Figure 3.1, where we have two clusters of classifiers. Considering how the cosine loss of those classifier will be distributed, we see that the bi-modality is a natural product of having two clusters. Since cosine loss is computed pair-wise, we see that for pairs in the same cluster, we get small values of cosine loss, and for pairs in different clusters, we get large values of cosine loss. Having approximately the same number of classifiers in each cluster, we see that this will result in half of the pairs



(a) (b)

Figure 3.2: (a) Histogram of squared cosine loss for original LIT, (b) Histogram of squared cosine loss after loss scaling

having small cosine losses, and half having large cosine losses. This is exactly what we observe in our histogram of cosine loss Figure 3.2(a). A simple solution for this scaling issue is to notice that we are summing over all pairs of models in our cosine loss, whereas for log-loss, we are only summing over our models. In other words, cosine loss is scaled $O(n^2)$ whereas log-loss is scaled $O(n)$, where n is the number of models in the ensemble. Therefore, we can fix this by taking the square root of our cosine loss in the overall loss function. When we use this as the loss function to train, and plot the histogram of cosine loss (noting that the individual pairwise cosine loss are still in the same scale as the original cosine loss), we get Figure 3.2(b). Noting that the issue of multi-modality is resolved, we propose the LIT loss to become

$$\text{LIT Loss} = \sqrt{\sum_{i=1}^{M}\sum_{j\neq i}^{M}\mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla f(\mathbf{x};\theta_i),\nabla f(\mathbf{x};\theta_j))]} \tag{3.22}$$

3.1.4 Issues with Input Gradients

One of the main assumptions we made in forming a closed form expression for LIT Loss in Equation (3.8) was that locally, the data manifold is Euclidean. In other words, in the neighbourhood of data points $\mathbf{x}$, we can say that $N_\epsilon(\mathbf{x}) \approx \mathcal{B}_\epsilon(\mathbf{x})$. This allowed us to approximate the directions along which f and g are opposed via the dot product of the gradients, instead of looking at the local tangent bundle of the manifold. A critical issue with this assumption is that in cases where it does not hold, we can obtain independence between two models by minimising the dot product between the gradients along certain dimensions but by keeping the dot product equal in the important dimensions that are most in-line with the manifold dimensions.

In many real-world datasets, the data manifold can be very different from Euclidean space [57, 65]. This difference arises when the data points lie on a lower-dimensional structure (the data manifold) in a higher-dimensional space. In such cases, the intrinsic geometry of the data manifold deviates from the flat Euclidean geometry, leading to

non-linear relationships between data points. This is especially true in image datasets, such as MNIST [41] or ImageNet [12] because the space of all possible images lies on a lower-dimensional manifold within the high-dimensional pixel space. Additionally, the local relationships between pixels and global relationships between image parts create complex, non-linear structures within the data manifold [47].

We illustrate this issue by showing the difference between the LIT loss computed using input space gradients (as in the original method) and using manifold gradients for synthetic manifolds. Using basic methods in differential geometry [15], we study how the gradients diverge when considering manifolds with differing intrinsic curvature. Our procedure will be as follows:

1. Choose a set of manifolds with different curvatures.
2. Compute the Jacobian matrix $\mathbf{J}_M : \mathbb{R}^D \to \Omega_{\mathbf{x}}$ for each manifold, which allows us to compute the gradient in the manifold, $\mathbf{g}_M = \nabla_M f(\mathbf{x})$, from our Euclidean gradient $\mathbf{g}_E = \nabla f(\mathbf{x})$, via $\mathbf{g}_M = \mathbf{J}^T \mathbf{g}_E$.
3. For two classifiers, $f(\mathbf{x})$ and $g(\mathbf{x})$, we compute their gradients in both Euclidean space and on the manifold.
4. We compute the LIT loss, as defined in Equation (3.8) for a set of random points on the manifold, using both the Euclidean gradients, $\mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla f(\mathbf{x}), \nabla g(\mathbf{x}))]$, and manifold gradients, $\mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla_M f(\mathbf{x}), \nabla_M g(\mathbf{x}))]$.
5. We gather the mean absolute difference between the LIT loss when we use Euclidean gradients vs manifold gradients.

For our set of manifolds, in order to keep the mathematics tractable, we use 2-spheres of different radii, which are 2 dimensional spheres embedded in 3 dimensional space. So, $\dim(\Omega_{\mathbf{x}}) = 2$ and $\mathbb{R}^D = \mathbb{R}^3$. This works because we can explicitly compute the scalar curvature of 2-spheres as $S(r) = \frac{2}{r^2}$. [15] In order to compute the Jacobian of this manifold, we need to parameterise the manifold, for which we use spherical coordinates

$$x = r\sin(\theta)\cos(\phi) \tag{3.23}$$
$$y = r\sin(\theta)\sin(\phi) \tag{3.24}$$
$$z = r\cos(\theta) \tag{3.25}$$

For which we get the following Jacobian, $\mathbf{J}_M$

$$\mathbf{J}_M = \begin{bmatrix} r\cos(\theta)\cos(\phi) & -r\sin(\theta)\sin(\phi) \\ r\cos(\theta)\sin(\phi) & r\sin(\theta)\cos(\phi) \\ -r\sin(\theta) & 0 \end{bmatrix} \tag{3.26}$$

For our classifiers, we choose two linear (in weights) classifiers with sigmoid activation,

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{1}^T\mathbf{x})} \tag{3.27}$$
$$g(\mathbf{x}) = \frac{1}{1 + \exp(-x - y^2 - z)} \tag{3.28}$$

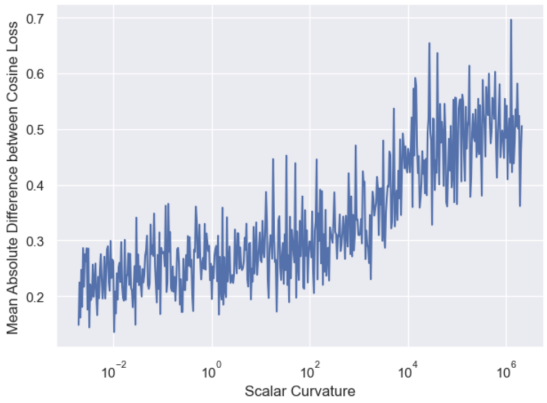


Figure 3.3: Mean Absolute Error between Euclidean LIT Loss and Manifold LIT Loss as a function of manifold curvature

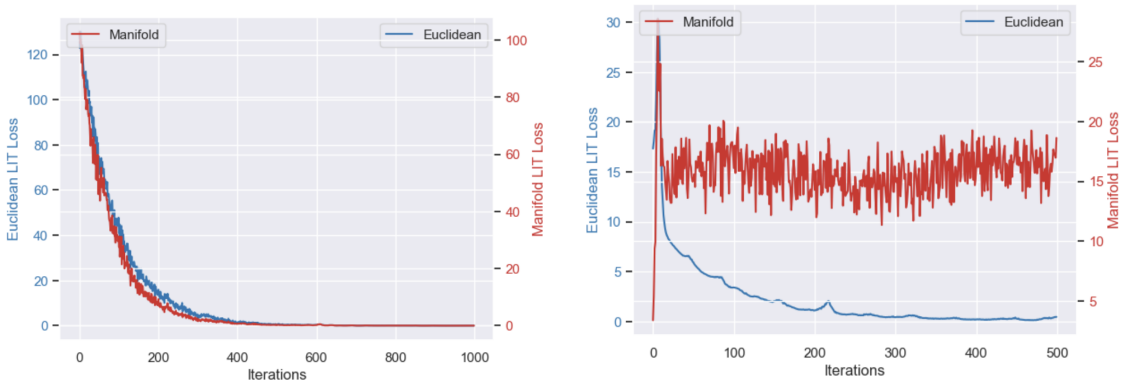
using these classifiers, we can compute our gradients and calculate the relationship between the scalar curvature and mean absolute error between the LIT loss. Before that, it is useful to compute the analytic relationship between the LIT losses

$$\begin{aligned}
\nabla_M f(\mathbf{x}) \cdot \nabla_M g(\mathbf{x}) &= (\mathbf{J}^T \nabla f(\mathbf{x})) \cdot (\mathbf{J}^T \nabla g(\mathbf{x})) && (3.29)\\
&= \nabla f(\mathbf{x})^T \mathbf{J}\mathbf{J}^T \nabla g(\mathbf{x}) && (3.30)\\
&\propto r^2 \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) && (3.31)\\
&\propto \frac{1}{S} \nabla f(\mathbf{x}) \cdot \nabla g(\mathbf{x}) && (3.32)
\end{aligned}$$

where ∇_M denotes the gradient along the manifold, and ∇ indicates the gradient in Euclidean space. From this, we see that the ratio between the Euclidean LIT loss and manifold LIT loss increases with the scalar curvature of the manifold. This analytic expression is also experimentally validated, as shown in Figure 3.3, where the mean absolute error between the Euclidean LIT Loss and the Manifold LIT loss increases with increasing scalar curvature of the manifold.

This illustrates the importance of using manifold gradients instead of Euclidean gradients when computing the LIT Loss. We confirm this further by comparing the evolution of LIT Loss across training for image datasets, when computing with the Euclidean and manifold gradients. This is difficult, however, because we are not able to obtain the exact structure and parameterisation of the manifold. In particular, we cannot directly infer the Jacobian that allows us to convert the Euclidean gradient into the manifold gradient.

Our proposed workaround for this issue is to use a generative model to learn the data manifold structure [61]. Latent Variable Generative models, such as Variational Autoencoders (VAEs) typically have a latent space produced by the encoder part of the generative model, that is supposed to be a lower dimensional representation of the dataset it generates. This is analogous to the concept of a manifold, a lower dimensional space (latent space) embedded in a higher dimensional space (input space). Therefore, we can consider the encoder as the mapping from the input space to the data manifold, and correspondingly, the derivative of that encoder with respect to the input space to be



(a) (b)

Figure 3.4: Evolution of LIT Loss computed with Euclidean vs Manifold gradients across iterations for (a) Toy 2D Classification, (b) MNIST Digit Classification

the Jacobian matrix that maps input space gradients to manifold gradients. Using this technique, we are able to investigate the impact of using Euclidean gradients instead of manifold gradients on the models trained using the LIT Loss.

Two experiments are performed, one on a 2D dataset made up of a mixture of two Gaussians, and another on MNIST. By training a set of LIT models, and computing the evolution of cosine loss computed with euclidean and manifold gradients, we are able to observe if there are any differences in convergence between the losses. Looking at Figure 3.4, we see that in our toy example, the choice of whether we use euclidean or manifold gradient does not matter as both cosine losses convergence together. However, in MNIST, we see that after a certain point in training, cosine loss computed using manifold gradients plateau, while the cosine loss computed using euclidean gradients continue to decrease. Intuitively, this means that we are developing local independence in the directions that are not represented in the manifold, whilst staying constant along the manifold. By doing so, we are not fully optimising local independence in the way we defined it in Equation (3.2).

3.1.5 Manifold Gradient LIT

Given the issues we discussed in Sections 3.1.3 and 3.1.4, we now develop an extended LIT Loss that accounts for both the scaling and gradient issues. This gives us the extended LIT Loss

$$\text{LIT Loss} = \sqrt{\sum_{i=1}^{M}\sum_{j\neq i}^{M} \mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla_M f(\mathbf{x};\theta_i), \nabla_M f(\mathbf{x};\theta_j))]} \tag{3.33}$$

$$\mathcal{L}(\{\theta_m\}) = \sum_{m=1}^{M} \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[-\log p(\mathbf{y}|f(\mathbf{x};\theta_m))] \tag{3.34}$$

$$+ \lambda_{LIT} \sqrt{\sum_{l\neq m} \mathbb{E}_{p(\mathbf{x})}[\cos^2(\nabla_M f(\mathbf{x};\theta_m), \nabla_M f(\mathbf{x};\theta_l))]} \tag{3.35}$$

where ∇_M denotes taking the gradient of the model on the data manifold. This gives us the following model training algorithm

Algorithm 1 Manifold LIT Algorithm

Input: Set of Models $\{f(\mathbf{x};\theta_m)\}_{m=1}^M$, Generative Model Encoder $q(\mathbf{z}|\mathbf{x})$, Training Data $\mathcal{D}$
Parameters: Number of Epochs N, Learning Rate γ, LIT Lambda λ_{LIT}
Initialise : $\{\theta_m\}_{m=1}^M$

for $i \in \{1, ..., N\}$ **do**
  for $\mathcal{B} \in \mathcal{D}$ **do**
    $\mathcal{L}_{logloss} = \sum_{m=1}^M \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{B}}[-\log p(\mathbf{y}|f(\mathbf{x};\theta_m))]$
    $\{\nabla f(\mathbf{x}|\theta_m)\}_{m=1}^M = \{\partial f(\mathbf{x}|\theta_m)/\partial \mathbf{x}\}_{m=1,\mathbf{x}\in\mathcal{B}}^M$
    $\mathbf{J} = \partial q(\mathbf{z}|\mathbf{x})/\partial \mathbf{x}$
    $\mathcal{L}_{LIT} = \sqrt{\sum_{m=1}^M \sum_{l\neq m}^M \mathbb{E}_{\mathbf{x}\in\mathcal{B}}[\cos^2(\mathbf{J}^T\nabla f(\mathbf{x};\theta_m), \mathbf{J}^T\nabla f(\mathbf{x}|\theta_l))]}$
    $\mathcal{L} = \mathcal{L}_{logloss} + \lambda_{LIT}\mathcal{L}_{LIT}$
    $\{\theta_m \ = \theta_m + \gamma\nabla_{\theta_m}\mathcal{L}\}_{m=1}^M$
  end
end
return $\{f(\mathbf{x};\theta_m)\}_{m=1}^M$;

Using this updated LIT Loss, we again investigate the performance on simple datasets. Firstly, we consider a 2D classification task with a mixture of Gaussians

$$p(\mathbf{x}|\mathbf{y}=0) = \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \tag{3.36}$$

$$p(\mathbf{x}|\mathbf{y}=1) = \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \tag{3.37}$$

$$p(\mathbf{y}=0) = p(\mathbf{y}=1) = \frac{1}{2} \tag{3.38}$$

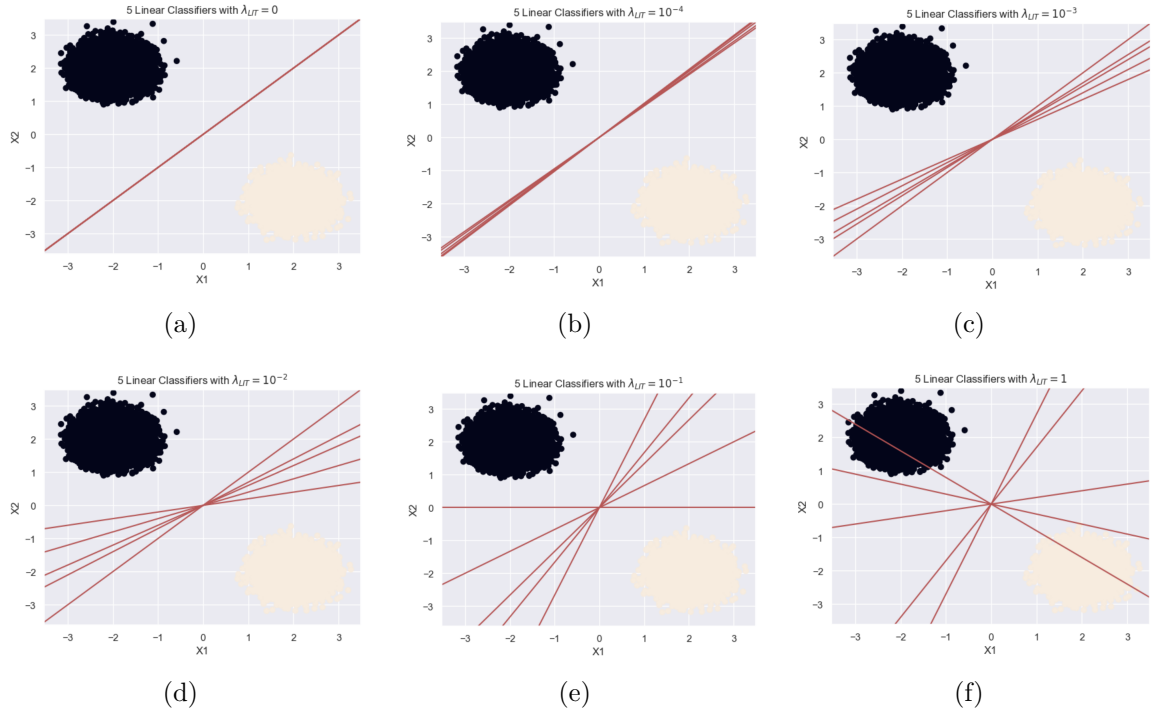
$$\boldsymbol{\mu}_1 = [2, -2] \quad \boldsymbol{\mu}_2 = [-2, 2] \tag{3.39}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{3.40}$$

On this dataset, we train five linear classifiers for varying values of λ_{LIT}, in the set $\lambda_{LIT} = \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$, and the classifiers are plotted in Figure 3.5. In this example, we observe that our results are as expected, and we are able to generate models of arbitrary diversity by controlling the tradeoff between cosine loss and log-loss. However, in order to test how using manifold gradients improves the performance of LIT, we need to test the method on datasets with intrinsic curvature.

3.1.6 Experimental Evaluation

In this section, we evaluate Manifold LIT against the Original LIT, Bagging [5], AdaBoost [19], and Negative Correlation Learning (NCL) [43] for the following datasets: MNIST [13], CIFAR-10 [33] and ImageNet [12]. All experiments here and in future sections are implemented in PyTorch [51], with Adam optimisation and learning rate scheduling [22]



(a) (b) (c)

(d) (e) (f)

Figure 3.5: $n = 5$ Classifiers trained using Manifold LIT Loss with varying λ_{LIT} = (a) 0, (b) 10^{-4}, (c) 10^{-3}, (d) 10^{-2}, (e) 10^{-1}, (f) 1

and AutoGrad [50] to compute various input gradients, used in Sections 3.1 to 3.3 and Jacobians, used in Section 3.1.5.

We choose MNIST because although it is a toy dataset, it has a more complex data manifold structure in comparison to our mixture of Gaussian dataset. This will allow us to explore how much value of Manifold LIT adds over the Original LIT and other ensemble methods that do not account for the data manifold structure. Classification on CIFAR10 is a more realistic example of real-world datasets and testing our methods on this will illustrate their value. Finally, classification on ImageNet is a challenging problem and current state-of-the-art classifiers with high accuracy are typically large scale neural networks, such as AlexNet [34], ResNet [25] and VGG [63]. By testing out our ensemble method in such a challenging task will allow us to explore how useful this method will be in realistic machine learning applications.

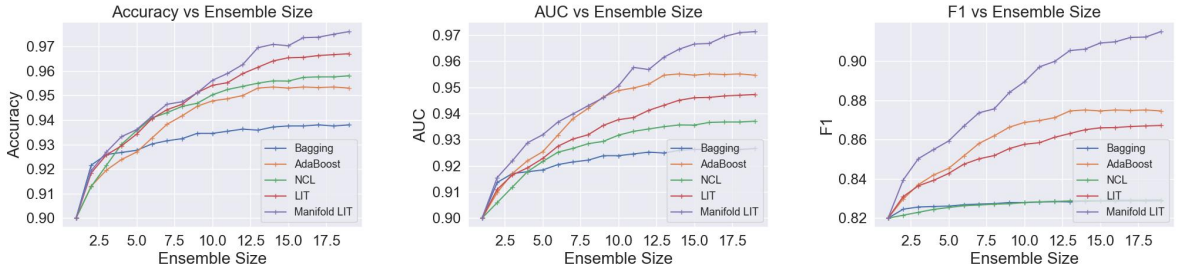
For MNIST, we conduct two experiments:

1. In experiment 1, we train our ensemble with a Convolutional Neural Network (CNN) as the base model. In particular, we use a CNN architecture that, when trained alone, gives us an accuracy of $\sim 90\%$. Using this CNN as our base classifier, we train our ensembles to convergence (by early stopping at loss function plateaus).

2. In experiment 2, we train our ensemble with a very simple base CNN that is especially weak at classification, with an accuracy of $\sim 50\%$ (again trained to convergence). Using this, we will be able to fully consider the utility of the ensemble learning methods even for a simple dataset like MNIST.

	Experiment 1			Experiment 2		
	Accuracy	**AUC**	**F1 Score**	**Accuracy**	**AUC**	**F1 Score**
Base Model	90.0	0.93	0.82	50.0	0.73	0.54
Bagging	93.8	0.95	0.83	55.1	0.77	0.57
AdaBoost	95.3	0.96	0.88	80.8	0.89	0.80
NCL	95.8	0.96	0.83	78.5	0.85	0.78
LIT	96.7	0.98	0.87	68.1	0.82	0.71
Manifold LIT	**97.6**	**0.98**	**0.92**	**82.8**	**0.90**	**0.83**

Table 3.1: Comparison of benchmark methods and LIT, Manifold LIT for MNIST digit classification in two scenarios, with best performance in bold.



(a) (b) (c)

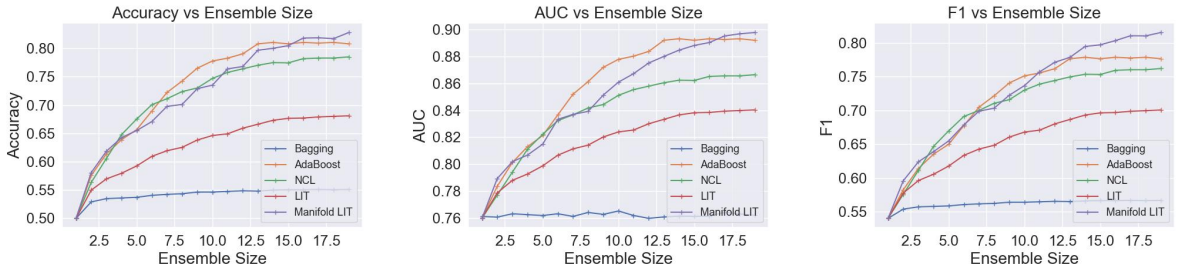
Figure 3.6: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for MNIST Experiment 1

For each of the experiments, and for all experiments in the future, we compute the following metrics to quantify the performance of the ensembles:

- Accuracy: the percentage of instances where the maximum probability prediction, $\arg\max P(\mathbf{y}|\mathbf{x}^*)$, is equal to the correct class.
- Area under ROC curve (AUC): we measure the area under the Receiver Operating Characteristics (ROC) curve. Since we have a multi-class prediction problem, we compute the one-vs-rest ROC curve and AUC for each class, and average it. Note that the range of values of AUC is $[0.5, 1]$, where 0.5 corresponds to random guessing, and 1 corresponds to perfect classification. A conversion from accuracy to AUC, in the case that our true positives and false positives and our class proportions are balanced will be scaling by half (i.e. an accuracy of 50% will corresponds to an AUC of ~ 0.75.)
- F1 Score: defined as $F1 = TP/(TP + 0.5(FP + FN))$, where TP is true positives, FP is false positives, FN is false negatives. As with AUC, we compute this as one-vs-rest for each class and average it.

Looking at Table 3.1, we note a few points:

- Ensembling improves the performance in both experiments and for all of the tested methods.



(a) (b) (c)

Figure 3.7: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for MNIST Experiment 2

- The increase in performance is particularly obvious in the case that the base model is a weak learner.
- For experiment 1, we see that Manifold LIT and LIT perform marginally better than the benchmark models, and Manifold LIT performs marginally better than LIT.
- For experiment 2, we see that bagging performs the worst, which makes sense given that the only diversity induction is via random sampling of the dataset. This means that we are very dependent on the base model performing adequately well to utilise the diversity generated by random sampling.
- Boosting on the other hand, vastly improves the performance of the weak learner. This is in fact exactly what AdaBoost is meant for: combining weak learners into a strong learner.
- NCL, LIT and Manifold LIT show performance improvement, but are not much better than boosting. This is likely due to the fact that in order to encourage diversity in NCL, we are promoting negative correlation between base model predictions. However, if the base model predictions are inaccurate/uncertain, we will not gain much as the bias term in the BVC decomposition will be very large.

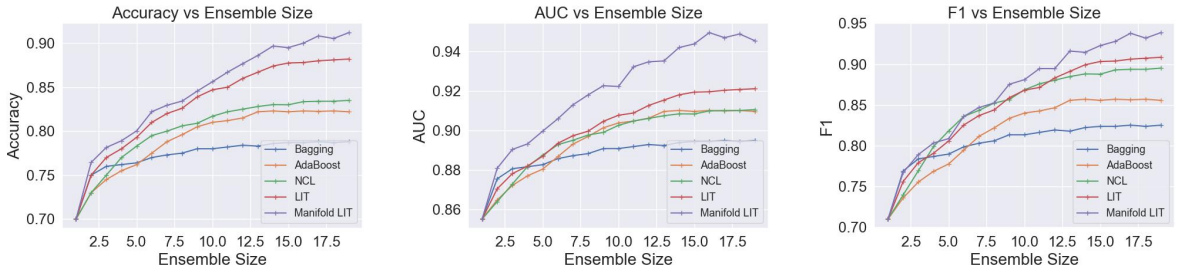
For CIFAR-10, we conduct the following experiments:

1. In experiment 1, we train our ensembles using a base CNN that has an accuracy of $\sim 70\%$.
2. In experiment 2, we train our ensembles using a more sophisticated ResNet as the base model, with accuracy $\sim 85\%$.

Before looking at the results of the experiments, a brief note on training the DGMs for CIFAR-10 is important. Unlike MNIST, training a VAE for CIFAR-10 required more nuance. After attempting a simple deep CNN as the encoder and decoder modules, which gives a very high Frechet Inception Distance (FID) score [26], we opted for a more sophisticated VAE architecture with Residual Network blocks [25]. Furthermore, during training, we used a learning rate scheduling that reduces the learning rate when the objective function stops improving [22]. Combining these, we get a FID score of 25.23,

	Experiment 1			Experiment 2		
	Accuracy	**AUC**	**F1 Score**	**Accuracy**	**AUC**	**F1 Score**
Base Model	70.0	0.84	0.66	85.0	0.92	0.82
Bagging	78.8	0.89	0.79	88.1	0.94	0.89
AdaBoost	82.3	0.91	0.82	89.2	0.84	0.90
NCL	83.5	0.92	0.86	93.4	0.96	0.90
LIT	88.2	0.94	0.87	94.2	0.96	0.91
Manifold LIT	**91.2**	**0.96**	**0.89**	**96.1**	**0.97**	**0.94**

Table 3.2: Comparison of benchmark methods and LIT, Manifold LIT for CIFAR-10 image classification in two scenarios, with best performance in bold.



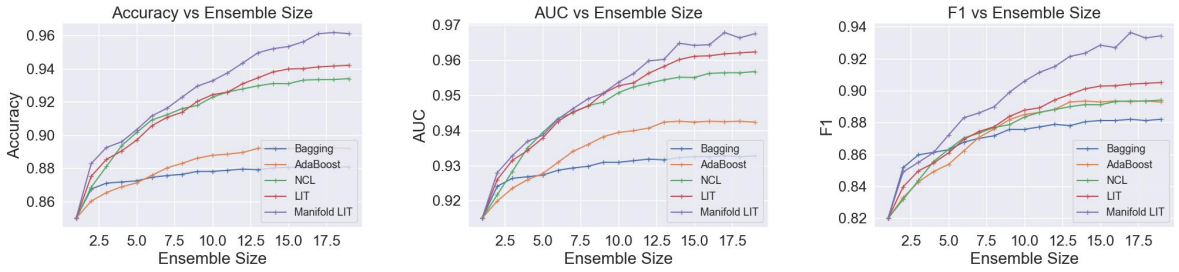
(a) (b) (c)

Figure 3.8: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for CIFAR-10 Experiment 1

which is much higher than the state-of-the-art models that achieve a FID score of ~ 2 [1]. This becomes a key limiting factor in the application of Manifold LIT.

Furthermore, we see that Manifold LIT requires a smaller ensemble size to achieve the same accuracy as our benchmark models. This points to Manifold LIT being a more efficient ensemble method. Moreover, we notice that LIT and Manifold LIT is less sensitive to perturbations in the λ_{LIT} compared to the corresponding hyperparameters in our benchmark models (λ_{NCL} etc.) The performance of LIT and Manifold LIT remain stable across several orders of magnitude, $[10^{-4}, 10^{-1}]$, for both MNIST and CIFAR-10 datasets, which makes hyperparameter tuning much more manageable.

In order to perform the experiments on ImageNet, for Manifold LIT in particular, we need to be able to compute the Jacobian via a VAE on ImageNet. Unable to train a good VAE on ImageNet, we looked for pre-trained VAEs, which were not readily accessible. Furthermore, for complex datasets like ImageNet, Generative Adversarial Networks (GANs) [7] and its variants are more frequently used. Unfortunately, since GANs have a generator-discriminator structure instead of an encoder-decoder structure, we cannot compute the required Jacobian from it. This is because even though we can invert the Jacobian, $\frac{\partial q(\mathbf{z}|\mathbf{x})}{\partial \mathbf{x}}|_{\mathbf{x}_i} = \mathbf{J}_{\mathbf{x}\to\mathbf{z}}(\mathbf{x}_i) = \mathbf{J}^{-1}_{\mathbf{z}\to\mathbf{x}}(\mathbf{z}_i)$, we still need to know $\mathbf{z}_i = g^{-1}(\mathbf{x}_i)$ where $g(\mathbf{z})$ is the generator of the GAN, i.e. in order to compute the Jacobian at a particular input space point, we need to know which latent space point corresponds to that input space point. One workaround would be if we could invert the generator. However, this is not a feasible



(a) (b) (c)

Figure 3.9: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for CIFAR-10 Experiment 2

task and the computational complexity would render Manifold LIT useless.

Overall, from our experimental evaluation, we show that Manifold LIT outperforms both the benchmarks and LIT, whilst remaining stable for several orders of magnitude of λ_{LIT}. However, a major limiting factor becomes the availablity and trainability of a VAE to model the data manifold in order to compute the required Jacobian for Manifold LIT.

3.1.7 Limitations of Manifold LIT

As we discussed before, the main limitation of Manifold LIT is the requirement of a Latent Variable Generative Model of the dataset, which is not always available, and usually difficult to train. This motivates Latent Divergence Training (LDT), which is introduced in Section 3.3, where this issue is circumvented.

3.1.7.1 Sensitivity to Generative Model Perturbations

A limitation of Manifold LIT we alluded to earlier is our reliance on a deep generative model (DGM) to compute gradients in the data manifold. For simple cases, such as our toy example and MNIST, this is easily done by training a DGM on the training dataset (or even a held-out dataset). However, in more complex experiments, and in reality, we have to be wary about relying on a DGM, assuming that it is going to be equivalent to probing the data manifold. Therefore, it is important to evaluate how much our method will be affected when perturbing the quality of the DGM.

For the case of Manifold LIT, we can start by analysing how the gradients change when we perturb the Jacobian matrix. Using the sub-multiplicative and sub-additive properties of matrix norms [30], we get the following proposition about the asymptotic behaviour of the error of manifold gradients as we perturb the Jacobian with Gaussian noise:

Proposition 3. *For* $\mathbf{g}_M = \mathbf{J}^T\mathbf{g}_E$, $\mathbf{g}'_M = \mathbf{J'}^T\mathbf{g}_E = (\mathbf{J} + \mathbf{\Delta})^T\mathbf{g}_E$, *where* $\mathbf{J} \in \mathbb{R}^{N\times M}$ *and* $\Delta_{ij} \sim N(0, \sigma^2)$, *as* $N, M \to \infty$, *with* $r = \frac{N}{M}$, *we say that, for some* $\epsilon > 0$, *with probability* $1 - \epsilon$,

$$||\mathbf{g}_M - \mathbf{g}'_M||_2 \leq ||\mathbf{g}_E||_2\lambda^* \tag{3.41}$$

where $F(\lambda^*) = 1-\epsilon$, $F(\lambda)$ is the cumulative distribution function (CDF) of the Marchenko-Pastur distribution [45], for $\lambda \in [-\lambda_-, \lambda_+]$ given as

$$F(\lambda) = \int_{\lambda_-}^{\lambda} p(\lambda')d\lambda' \tag{3.42}$$

$$p(\lambda) = \frac{1}{2\pi\sigma^2}\sqrt{\frac{(\lambda_+ - \lambda)(\lambda - \lambda_-)}{\lambda}} \quad \lambda_\pm = \sigma^2(1 \pm \sqrt{r})^2 \tag{3.43}$$

Furthermore, we can bound, with probability 1*, that*

$$||\mathbf{g}_M - \mathbf{g}'_M||_2 \leq ||\mathbf{g}_E||_2\lambda_+ \tag{3.44}$$

Proof. We prove this as follows:

$$\begin{aligned} ||\mathbf{g}_M - \mathbf{g}'_M||_2 &= ||\mathbf{J}^T\mathbf{g}_E - \mathbf{J}'^T\mathbf{g}_E||_2 && (3.45)\\ &= ||\mathbf{J}^T\mathbf{g}_E - (\mathbf{J}^T + \mathbf{\Delta}^T)\mathbf{g}_E||_2 && (3.46)\\ &= ||\mathbf{\Delta}^T\mathbf{g}_E||_2 && (3.47)\\ &\leq ||\mathbf{g}_E||_2\rho(\mathbf{\Delta}) && (3.48) \end{aligned}$$

where we used the fact that the l_2 matrix norm is equal to the maximum singular value of the matrix, also known as the spectral radius, $\rho(\Delta)$. Now, we can directly say that, for some $\epsilon > 0$, with probability $1 - \epsilon = F(\lambda^*)$, $\rho(\Delta) \leq \lambda^*$ which gives Equation (3.41). Further, by setting $\lambda^* = \lambda_+$, we can say with probability 1 that Equation (3.44) holds and this concludes the proof. □

This proposition tells us that we can bound the error in manifold gradients for Gaussian perturbations of the Jacobian with the spectral radius of the perturbation matrix, $\mathbf{\Delta}$, which can subsequently be bounded, with some probability, according to a eigenvalue distribution defined in terms of the ratio of dimensions, $r = N/M$ of the Jacobian. However, note that this is an asymptotic analysis and for small matrices, we are only able to use the spectral radius bound given in Equation (3.48). We can develop more intuition on when our Jacobians are well-behaved for our method.

- Smoothness of the manifold: If the data manifold modelled by the VAE is smooth, it can contribute to a smaller spectral radius. Smoothness in this context refers to the absence of abrupt changes or discontinuities in the data manifold. A smooth manifold allows for locally well-behaved mappings and is characterised by gradual variations of data points within its neighbourhood. When the data manifold is smooth, the Jacobian captures the local linearisation of the mapping and exhibits stable behaviour. This can lead to a smaller spectral radius, indicating that the manifold tangent space undergoes less expansion/contraction. [4, 3, 21] show that the data manifolds of datasets such as ImageNet and CIFAR-10 are indeed smooth under this definition.

- Low-Dimensional Struture: If the data manifold has an intrinsic low-dimensional structure, where the data lies in a lower-dimensional subspace embedded in a higher-dimensional space, the Jacobian spectral radius can be small. This is because the

low-dimensional structure implies that the manifold variations occur primarily in a smaller set of dimensions, resulting in a more contrained behaviour captured by the Jacobian. Furthermore, when the dimensionality reduction is large, the ratio $r = N/M$ is also small, which gives a tighter bound in Equation (3.44). [57, 27] show that data manifolds of real-world datasets indeed form a much lower-dimensional manifold embedded in a higher-dimensional input space.

To make this analysis more rigorous, we consider how perturbing the weights in our DGM will affect the encoder function and its gradients. Let our encoder function be $f(\mathbf{x}; \mathbf{w}) : \mathcal{X} \to \Omega_{\mathbf{x}}$, parameterised by weights $\mathbf{w}$, where $\mathcal{X}$ is the input space and $\Omega_{\mathbf{x}}$ is the data manifold as before. Without loss of generality, we assume that we are in a Hilbert space that permits a Fourier-like basis $\phi_k(\mathbf{x})$, and that the s-th derivative of $\phi_k(\mathbf{x})$ is proportional to k^s (which is the case for the typical Fourier basis). So, we can express our encoder function as $f(\mathbf{x}; \mathbf{w}) = \sum_{k=1}^{\infty} c_k(\mathbf{w})\phi_k(\mathbf{x})$. Using this, we can bound the L^2 norm of $||f(\mathbf{x}; \mathbf{w}') - f(\mathbf{x}; \mathbf{w})||_2^2$, where $\mathbf{w}' = \mathbf{w} + \delta\mathbf{w}$. To do so, we make the assumption that our encoder function belongs to the Sobolev space of functions $W_2^s(\mathcal{X}) = \{f \in L^2(\mathcal{X}); D^s f \in L^2(\mathcal{X})\}$, which is also a Hilbert space, where $L^2(\mathcal{X})$ is the space of square-integrable functions on $\mathcal{X}$. Our Sobolev space has norm [17] defined as

$$||f||_{W_2^s}^2 = \left(\int_{\mathcal{X}} |D^s f(\mathbf{x})|^2 d\mathbf{x} \right)^{1/2} \tag{3.49}$$

$$= ||D^s f||_{L^2}^2 \tag{3.50}$$

$$= \sum_{k=1}^{\infty} k^{2s} c_k^2 \tag{3.51}$$

where we used Parseval's identity [52]. Putting these together, we get that

$$||f(\mathbf{x}; \mathbf{w}') - f(\mathbf{x}; \mathbf{w})||_{L^2}^2 = \sum_{k=1}^{\infty} (c_k(\mathbf{w}') - c_k(\mathbf{w}))^2 \tag{3.52}$$

$$= \sum_{k=1}^{\infty} (c_k' - c_k)^2 \tag{3.53}$$

$$\leq \sum_{k=1}^{\infty} 2(c_k'^2 + c_k^2) \frac{1}{k^{2s}} k^{2s} \tag{3.54}$$

$$= \sum_{k=1}^{\infty} \frac{2}{k^{2s}} k^{2s} (c_k'^2 + c_k^2) \tag{3.55}$$

$$\leq \frac{2}{n^{2s}} \sum_{k=n+1}^{\infty} k^{2s} (c_k'^2 + c_k^2) \tag{3.56}$$

$$\leq \frac{2}{n^{2s}} (||f(\mathbf{x}; \mathbf{w}')||_{W_2^s}^2 + ||f(\mathbf{x}; \mathbf{w})||_{W_2^s}^2) \tag{3.57}$$

where we used the Cauchy-Schwartz inequality [64] to bound the L^2 norm, and brought out a factor of n to get the Sobolev norm. To get Equation (3.56), we need to assume that the effect of perturbing the weights is to change higher frequency terms in the encoder function. In other words, when we perturb the weights, we make small changes in the encoder function, which is usually the case for a well trained VAE. To understand why,

consider the training procedure of a VAE. Over the course of training, the encoder function learns to map the input space to the latent space. At the start of training, the encoder will be mapping the larger scale global structures of the input space, and subsequently after some epochs, the encoder will focus on the fine grained details of the data manifold [66]. Therefore, for an underfit VAE, perturbing the weights will result in a large scale (low spatial frequency) change in the encoder function outputs, and for an well trained VAE, perturbing the weights will result in smaller scale (high spatial frequency) change in the encoder function outputs. This gives us a few note-worthy intuitions:

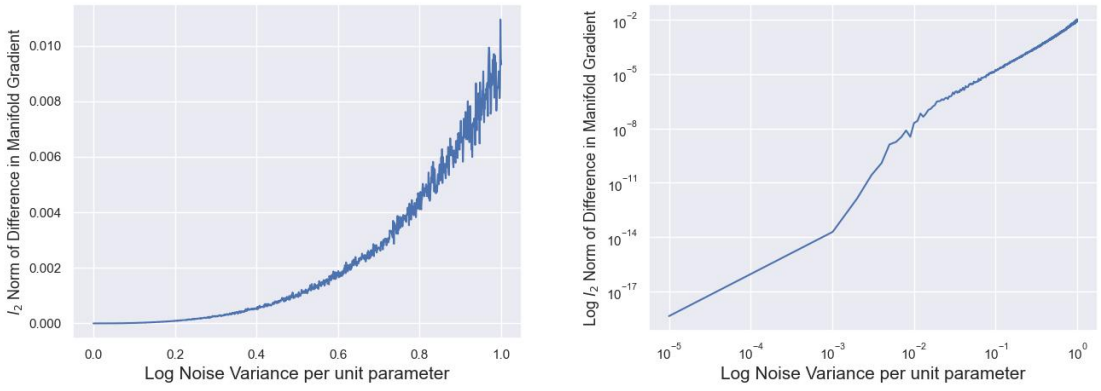
- The larger n is, the smaller the variations of the encoder is when we perturb the weights. The value of n depends on how our weight perturbations affect the encoder function in terms of the Fourier basis. The higher the frequency of the components our weight perturbations affects, the larger the value of n and hence the smaller the variation. Intuitively, this means that if our VAE captures global structure using low-frequency components and the fine-grained structure using high-frequency components, as [44] suggests, then the value of n will be large.

- The larger s is, the smaller the variations of the encoder is when we perturb the weights. The value of s depends on the smoothness of the encoder function in our VAE. This directly ties to our previous analysis, where the smoothness of the data manifold is suggested in literature, and that would apply here as a smoother data manifold, combined with an encoder that is not overfit to the data (overfit models tends to have large variations in their values, which decrease smoothness) will result in a larger value of s.

In order to experimentally evaluate the effect of perturbing the DGM parameters, we perform an experiment on MNIST, where we compute the l_2 norm of the difference of manifold gradients when Gaussian noise is added to each of the VAE parameters.

- Experiment 1: we first compute the 'original' manifold gradient at a set of points for a VAE with no added noise. For a range of noise variance values, defined as a proportion of the scale of the parameter, ranging from 10^{-5} to 1, we add this noise to each of the VAE parameters, and compute the Jacobian and manifold gradient with this noisy Jacobian. Now, we compute the difference between our noisy manifold gradient and the 'original' manifold gradient, and take the l_2 norm of this vector. This norm will give us an idea of how much the noise varies the manifold gradient vector.

Looking at Figure 3.10, we see that the l_2 norm is initially at very low levels and it rises exponentially as noise variance approaches the magnitude of the parameters. We get a clearer picture by looking at the log-scale plots, where we see that there is almost a transition in the 10^{-3} to 10^{-2} range, where the curve becomes steeper. If we think of this as a phase transition, we can say that as long as our noise variance is less that 1% of the parameter value, our manifold gradients are essentially the same. Furthermore, even when the noise variance is equal to the parameter scale, we see that the l_2 norm is only 0.01, which is almost insignificant for a 16 dimensional vector. This allows us to conclude that even experimentally, our Manifold LIT method will not be overly sensitive to the parameter variations of the DGM.



(a) (b)

Figure 3.10: Effect of parameter noise on manifold gradients, plotted in (a) linear axes, (b) log-log axes

3.1.8 Evaluating Model Diversity

We see that using our training procedure, we are able to obtain better than benchmark performance in our validation data sets. However, apart from the path of our LIT Loss, and our distribution of pairwise cosine loss between the models, we are unable to interpret the diversity of our models. In particular, we are not able to tell how our model predictions differ from each other, and whether the models predictions are negatively correlated, which as we saw in Section 2.1, was critical in understanding the bounds on the generalisation error of an ensemble. This is a common issue in machine learning, especially when we are dealing with large-scale models that provide very little interpretability. Although there have been work in this field on using local gradients to explain the model predictions [2], there is very little work in finding points where a set of large scale models differ in predictions.

3.2 GeValDi: Generative Validation of Disrciminative Models

In this section, we introduce a novel method to evaluate differences between discriminative models using generative models. As explained earlier, this is important when we are using large scale discriminative models that are complex to visualise and understand. In particular, in order to understand the diversity between models, and where they differ, we need some method of visualising and evaluating points of difference between model predictions. *All the work presented in this section henceforth has been accepted at ICLR 2023 Workshop on Trustworthy ML, and ICLR 2023 Tiny Papers Track (with presentation), both of which can be found at [48, 49].*

We *syntheise* data for which the predictions of a set of models differ maximally by optimising in the latent space of a generative model. We dub our approach GeValDi: Generative Validation of Discriminative models. In the following sections, we formally introduce the method, investigate various examples and apply it to our ensembles.

3.2.1 Formalism

GeValDi is a framework for generative validation of discriminative models, and in this section, we describe its working in detail. Given a set of M discriminative models $\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M$, GeValDi generates data points that maximise the divergence between the models. Formally, GeValDi finds a point $\tilde{\mathbf{x}} \in \mathcal{X}$ on the data manifold where the set of classifiers differ maximally, called a maximally diverse sample (MDS), which is defined as:

$$\tilde{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{X}} D\left[\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M\right] \tag{3.58}$$

Here, $\mathcal{X}$ denotes the input space, and D denotes some divergence measure, such as the KL-divergence [38] or Jensen-Shannon divergence [11]. However, without restricting the search space, this objective may lead to samples that are not representative of the true data distribution, and such models may not encounter such samples in practice. Therefore, it is necessary to constrain the optimisation to the data manifold.

GeValDi addresses this issue by introducing a latent variable model, where we can optimise in the latent space, instead of the input space. Specifically, consider the latent variable model:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{3.59}$$

Here, $p(\mathbf{z})$ denotes the prior over the latent variable $\mathbf{z}$, and $p(\mathbf{x}|\mathbf{z})$ is some conditional distribution with mean function $E[p(\mathbf{x}|\mathbf{z})] = g(\mathbf{z})$. By assuming that $p(\mathbf{x})$ accurately models the data distribution, GeValDi can formalise the problem of generating an MDS with high-probability under the true data distribution as follows:

$$g^{-1}(\tilde{\mathbf{x}}) = \arg\max_{\mathbf{z}\in\mathcal{Z}} D\left[\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M\right] + \lambda_{prior}\log p(\mathbf{z}) \tag{3.60}$$

Here, λ_{prior} is a regularisation constant that balances the flexibility of the optimisation problem with the likelihood of the generated sample under $\tilde{p}(\mathbf{x})$. The additional regularisation term encourages the generated samples to be representative of the true data distribution.

Algorithm 2 MDS Algorithm

Input: Set of Models $\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M$, Generative Model $p_\theta(\mathbf{x}|\mathbf{z})$ Divergence Metric $D\left[\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M\right]$

Parameters: Number of Epochs N, Learning Rate γ, Prior Lambda λ_{prior}

Initialise : $\tilde{\mathbf{z}}$

for $i \in \{1, ..., N\}$ **do**

  $\tilde{\mathbf{x}} \sim p_\theta(\mathbf{x}|\mathbf{z} = \tilde{\mathbf{z}})$

  Classifier Predictions $= \{p_m(\mathbf{y}|\mathbf{x} = \tilde{\mathbf{x}})\}_{m=1}^M$

  $L = D\left[\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M\right] + \lambda_{\text{prior}} p_\theta(\tilde{\mathbf{z}})$

  $\tilde{\mathbf{z}} = \tilde{\mathbf{z}} + \gamma\nabla_{\mathbf{z}} L(\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^M, p_\theta(\mathbf{x}|\mathbf{z}))$

end

return $\tilde{\mathbf{z}}$;

By optimising in the latent space, GeValDi ensures that the generated samples are from

the true data manifold. As latent spaces are typically low-dimensional, this optimisation provides a computationally inexpensive way to generate MDS. The overall GeValDi framework involves finding MDS that maximise the divergence between a set of discriminative models using a generative model, by optimising in the latent space. This "MDS Algorithm" is described in Algorithm 2.

It is worth noting the types of divergence metrics we can use here, and when they are appropriate. Two notable divergence metrics are the KL divergence and Jensen-Shannon (JS) divergence. KL divergence is only defined for pairs of distributions, and hence we can only appropriately use it when we are attempting to generate MDS on pairs of models. We do this in our toy example due to computational efficiency, and intuitive ease (KL divergence behaviours are well-known). However, JS divergence is more appropriate when we are generating MDS on a set of models. Even in this case, there is a slight ambiguity on how to evaluate JS divergence across a set of models. We can either (i) evaluate the JS divergence pairwise and average across all pairs, or, (ii) we can evaluate the JS divergence across all models at once with equation Equation (3.61)

$$D_{JS}(\{p_m(\mathbf{y}|\mathbf{x})\}_{m=1}^{M}) = \frac{1}{M}\sum_{m=1}^{M} D_{KL}(p_m(\mathbf{y}|\mathbf{x})||\bar{p}(\mathbf{y}|\mathbf{x})) \tag{3.61}$$

$$\bar{p}(\mathbf{y}|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_m(\mathbf{y}|\mathbf{x}) \tag{3.62}$$

The choice on which divergence metric to use, and whether to compute it pairwise or on a set becomes a design choice depending on the context.

3.2.2 Toy Example

In this section, we investigate the validity of our MDS algorithm by applying it to the toy example of 2D classification with a mixture of Gaussians, as in Equation (3.40). As a simple case, we use KL divergence as our divergence measure in our MDS algorithm. By fitting two simple linear classifiers, and generating MDS points for various values of λ_{prior}, we able to get an understanding of how the algorithm works in practice. The set of plots that show the i) MDS points overlaid on the dataset and classifier decision boundaries, ii) the sample space path of the MDS points over the latent space optimisation procedure, iii) the evolution of the objective function, along with the decomposition into KL divergence and Log Prior distribution, is shown in Figure 3.11, for $\lambda_{prior} = \{0, 10^{-4}, 10^{-3}\}$.

We notice a few interesting points in the plots. When $\lambda_{prior} = 0$, we see that our KL divergence term dominates and the points we get as maximally diverse are on extreme ends of the second and fourth quadrants. In order to understand why this occurs, we need evaluate what the theoretically expected KL divergence will be for our toy example. However, this is not an easy task for a general pair of classifiers, and would not yield any interpretable results. So, with some intuition, we can break down the various scenarios that can occur.

Given that both classifiers have perfect accuracy in our toy example, we can decompose the classifier predictions into two cases: Case 1 will be when the both classifiers are confident in their predictions, and their probabilities are perturbations away from each other. In other words, $p_1(\mathbf{y}|\mathbf{x}) = [p, 1-p]$ and $p_2(\mathbf{y}|\mathbf{x}) = [p+\epsilon, 1-p-\epsilon]$. In this case, we

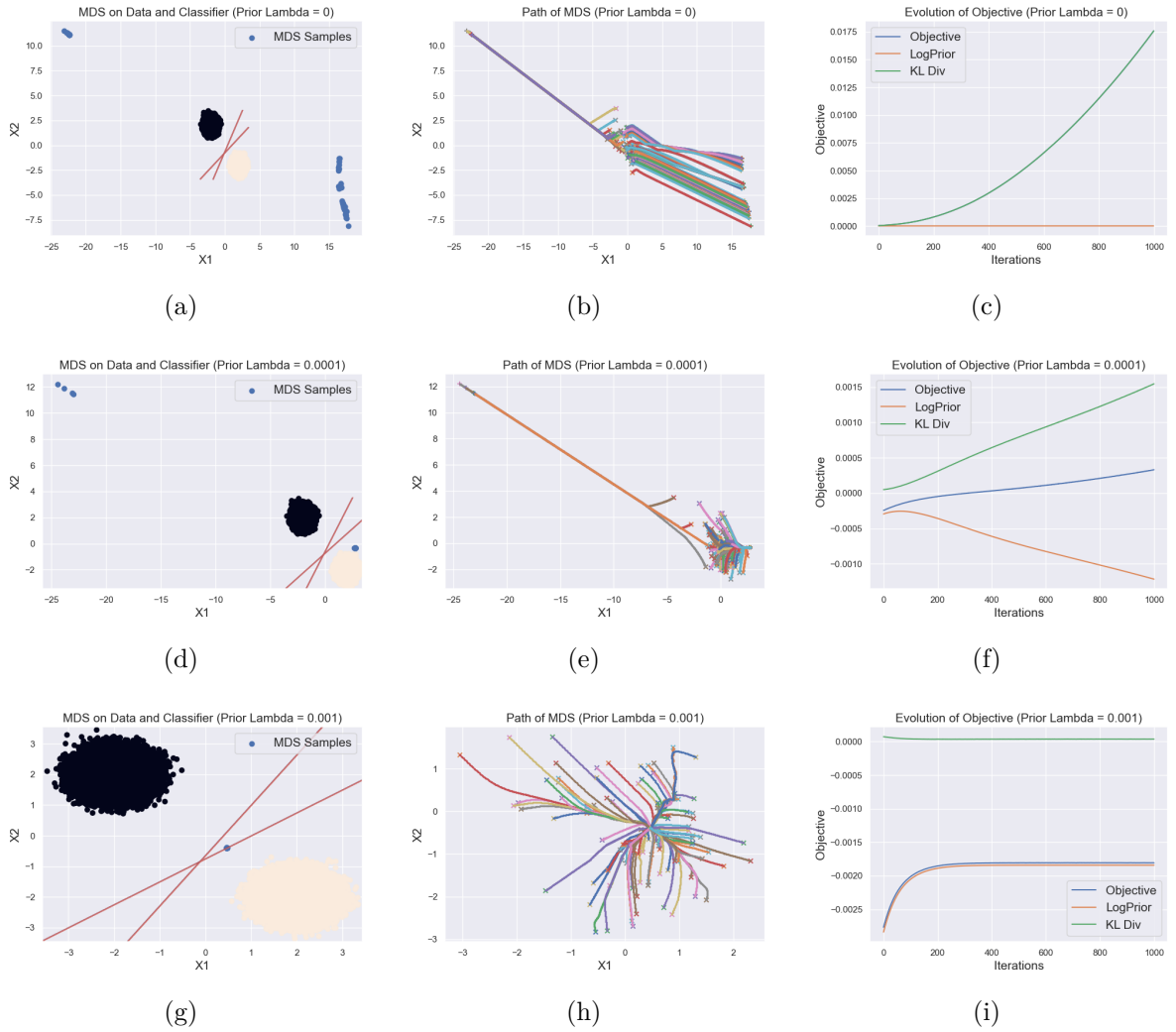


Figure 3.11: Sets of 3 plots: (a, d, g) MDS Points overlaid on data and decision boundary. (b, e, h) Path of MDS points where $\times$ denotes the initial point and $+$ denotes the end point. (c, f, i) Evolution of objectives, broken down into KL divergence and Log Prior terms. Plotted for $\lambda_{prior} = \{0, 10^{-4}, 10^{-3}\}$ as $\{(a, b, c), (d, e, f), (g, h, i)\}$ respectively.

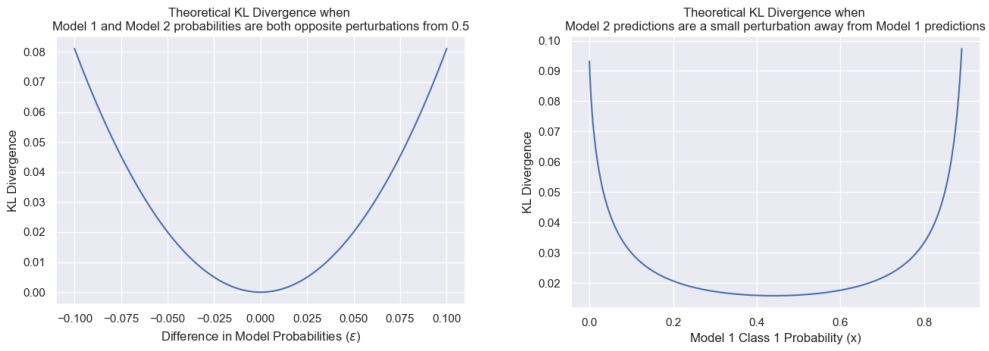


Figure 3.12: Theoretical Values of KL Divergence for 2 Model Scenarios

can write the KL divergence as

$$D_{KL}(p_1, p_2) = p \log\left(\frac{p}{p+\epsilon}\right) + (1-p)\log\left(\frac{1-p}{1-p-\epsilon}\right) \tag{3.63}$$

We plot the KL divergence as a function of p for a fixed ϵ in Figure 3.12(b). We see that KL divergence is maximised when the model confidence is high (i.e. when p is either close to one or zero).

Case 2 is when our models are not confident in their predictions, but are predicting opposite hard classifications. In other words, $p_1(\mathbf{y}|\mathbf{x}) = [0.5+\epsilon, 0.5-\epsilon]$ and $p_2(\mathbf{y}|\mathbf{x}) = [0.5-\epsilon, 0.5+\epsilon]$. In this case, we can write the KL divergence as

$$D_{KL}(p_1, p_2) = 2\epsilon \log\left(\frac{0.5+\epsilon}{0.5-\epsilon}\right) \tag{3.64}$$

We plot this KL divergence as a function of $\epsilon \in [-0.1, 0.1]$ in Figure 3.12(a). The choice of the range of ϵ comes from two factors. Intuitively, our models should not be diverging too much in their uncertain predictions (i.e. in the region between the decision boundaries). Empirically, if we compute the model probabilities of Model 1 at the sample values of the decision boundary of Model 2, and vice-versa, we get $\epsilon = \pm 0.08$, which we round up to $\epsilon = \pm 0.1$.

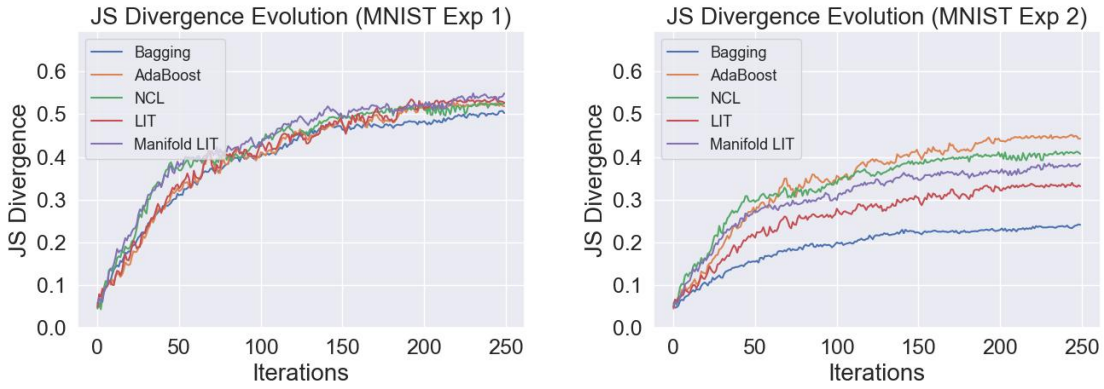
From the two theoretical KL divergence plots, we see that maximum values of Case 2 are typically smaller than maximum values of Case 1. So, we can see that the points that maximise the KL divergence are the points that our models are most confident about, and the model probabilities are small perturbations away from each other. This is exactly what we observe in our empirical evaluation with $\lambda_{prior} = 0$.

The occurrence of such a phenomenon is why it is important to include the log-prior term as a regulariser as it balances how much we value divergence vs sample likelihood in the data manifold. However, as we will discuss in future sections, the impact of λ_{prior} diminishes in datasets with a complex data manifold, because of the highly non-linear nature of the loss landscape, our discriminative models are not able to perfectly separate the classes, which results in regions within the high-probability region of the data manifold having large divergences.

As we increase λ_{prior}, we notice that firstly the term that dominates the objective function drastically changes, and at the same time, the points that are generated become Case 2 in our previous analysis, as these points are more likely under the data manifold than the points that maximise Case 1. In particular, we see that regardless of where the starting point is, we converge to very similar points in the end, as our λ_{prior} increases.

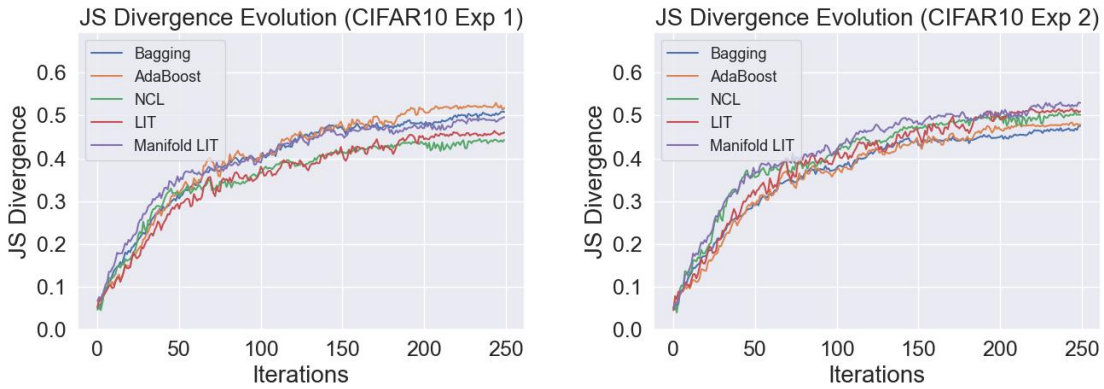
3.2.3 Experimental Evaluation

In line with the purpose of GeValDi, we apply our MDS algorithm to the benchmark and Manifold LIT classifiers we trained in Section 3.1.6 to investigate how the predictions of those classifiers actually differ, and where in the input space it differs. Given that we observed some clustering in the MDS points in Figure 3.11, we suspect that the clustering behaviour of MDS points can give valuable insight into how the ensemble performs.



(a) (b)

Figure 3.13: MNIST (a) Experiment 1 (b) Experiment 2 Evolution of JS Divergence



(a) (b)

Figure 3.14: CIFAR (a) Experiment 1 (b) Experiment 2 Evolution of JS Divergence

(a) (b) (c) (d) (e) (f)

Figure 3.15: MDS Samples from various clusters for (a-c) CIFAR-10 from clusters 1,2,3 respectively, (d-f) MNIST from clusters 1,2,3 respectively

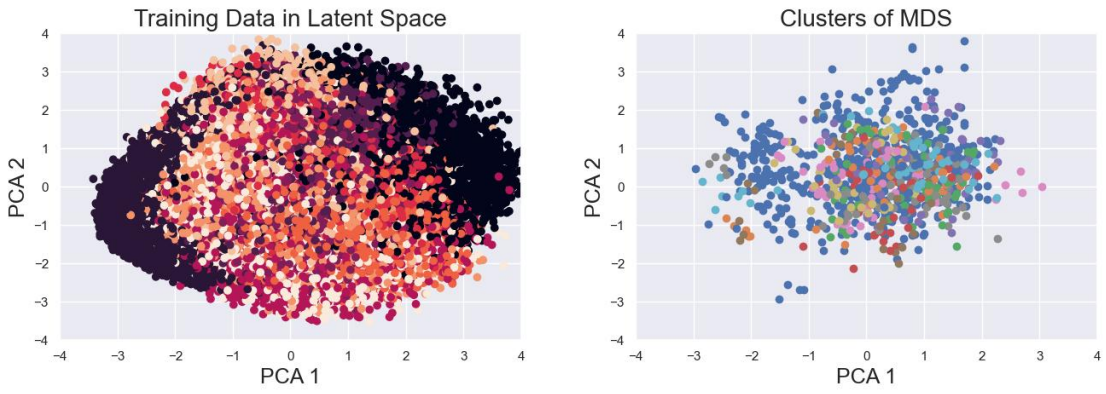
	Test Dataset			MDS Dataset		
	Accuracy	**AUC**	**F1 Score**	**Accuracy**	**AUC**	**F1 Score**
Bagging	93.8	0.95	0.83	65.3	0.80	0.66
AdaBoost	95.3	0.96	0.88	68.2	0.82	0.68
NCL	95.8	0.96	0.83	69.3	0.85	0.65
LIT	96.7	**0.98**	0.87	72.1	0.87	0.71
Manifold LIT	**97.6**	**0.98**	**0.92**	**75.2**	**0.89**	**0.73**

Table 3.3: Comparison of accuracies in the test dataset and MDS dataset, for Experiment 1 of MNIST, with best performance in bold.

In order to do so, we first run our MDS algorithm on our ensembles from MNIST and CIFAR-10 datasets, with JS divergence computed on the set of the base models as the divergence metric, according to Equation (3.62). The optimisation curves for JS divergence is Figures 3.13 and 3.14. The JS divergence curves we obtained are by computing the evolution of JS divergence across optimisation when we for 1000 different initial latent space points. This allows us to find the 'average' evolution of JS divergence, and the final convergent values of JS divergence give us an indication as to the average value of maximum JS divergence we can expect for a given set of classifiers.

A natural question that arises is what we can do with such MDS points, other than extract the convergent JS divergence value? Intuitively, the points where base models differ the most characterise regions of largest uncertainty for the ensemble. We understand this by considering the cases when JS divergence is maximised. If we have a point where base models all agree, we can either have that all base models predict correctly or incorrectly. This case is not good for ensemble performance because we want to encourage diversity. On the other hand, if we have a point with very large JS divergence (say, close to the theoretical bound of $\ln(2)$), this indicates that the base models are themselves very confident in their predictions, but each of the base models are confident in different predictions. This can be understood by considering the functional form of JS divergence, from which we know that JS divergence is maximised by having one-hot probability distributions (i.e. one on a particular class and zeros on all other classes, but that particular class is different for different base classifiers). This case is also detrimental to our ensemble performance because the divergence between base models inhibits the accuracy. So, there is a sweet spot between having no divergence and too much divergence between base models. Furthermore, MDS points form regions of too much disagreement between base models. We show this by computing the accuracy of the ensemble on the MDS points (generated by a conditional generative model, so we have the true label) and a test dataset of same size, which is depicted in Table 3.3, where we see a big decrease in accuracy between the test dataset and MDS points. This is why MDS points characterise regions of uncertainty in ensembles.

This points us to find if we can fully characterise all the uncertain regions of an ensemble by using the MDS points. One way would be to cluster the points and make the clusters represent the regions of uncertainty of the ensemble. For this, we use a Density-based spatial clustering of applications with noise (DBSCAN) algorithm [16]. The choice of this clustering algorithm comes from the fact that we do not need to choose the number of



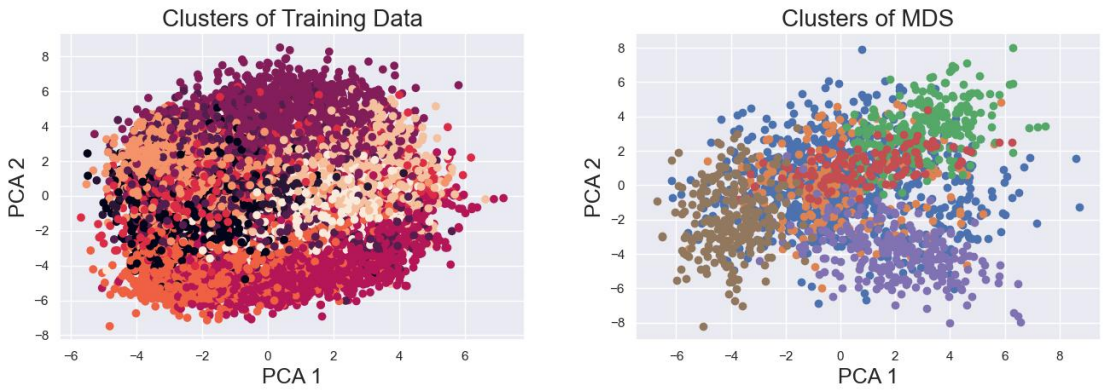
(a) (b)

Figure 3.16: (a) MNIST training data clustered by true labels, (b) MNIST MDS data clusters in latent space on top 2 principal components

clusters apriori. This allows us to investigate the number of clusters present in the data in a more general way. However, we still have to tune the radius of search around each point as a hyperparameter, for which we can perform a simple grid search. Since our goal with clustering is to understand how big and far apart the regions of uncertainties are, we can compute a Cluster Spread Index (CSI), defined as the ratio between the average intra-cluster distance and the average inter-cluster distance. CSI ranges between $(0, \infty)$, and a large CSI means that we have large clusters that are far apart from each other. Given our understanding that MDS points to regions of uncertainty in ensemble predictions, if MDS clusters have a large CSI, it would mean that we have several large regions where the ensemble is uncertain. For a $10,000$ sample MDS dataset for MNIST with LIT classifiers, and a $1,000$ sample MDS dataset for CIFAR-10 with LIT classifiers, the clusters in the latent space is illustrated in Figures 3.16 and 3.17, where we plot the clusters on the top two principal component axes for effective visualisation. Comparing the clusters of true labels in latent space and the MDS clusters, we see that MDS clusters form at regions where the label clusters overlap, which agrees with our intuition that these MDS points form regions of uncertainty in our ensemble estimates which naturally arise when the data can be equally from two or more labels.

Examples of MDS points from various clusters is depicted in Figure 3.15. We note a few interesting aspects here For MNIST, cluster 1, i.e. Figure 3.15(d), consists of images where the digit can either be a $9, 5, 3$. Similarly, cluster 2, Figure 3.15(e), consists of digits similar to $1, 7, 2$, and finally, cluster 3, Figure 3.15(f), consists of digits similar to $8, 3$. By sampling points from the clusters, we can understand specific types of data that the ensemble struggles with, and we can subsequently use these points to further improve the ensemble. We can confirm this by considering the maximum probability classifications of the base model, which are consistent with the possible labels of the images.

For CIFAR, a similar analysis can be made, so that cluster 1, Figure 3.15(a), consists of images of either 'airplane' or 'ship'. Similarly, cluster 2, Figure 3.15(b), consists of images of either 'cat' or 'dog' and finally, cluster 3, Figure 3.15(c), consists of images of either 'truck' or 'automobile'.



(a) (b)

Figure 3.17: (a) CIFAR10 training data clustered by true labels, (b) CIFAR10 MDS data clusters in latent space on top 2 principal components

Overall, from our experimental evaluation, we show that GeValDi serves as a method to identify and characterise regions of uncertainty in ensembles, and using latent space optimisation, we are able to find MDS points that characterise these regions. These points give useful qualitative evaluations of the failure modes of the set of models.

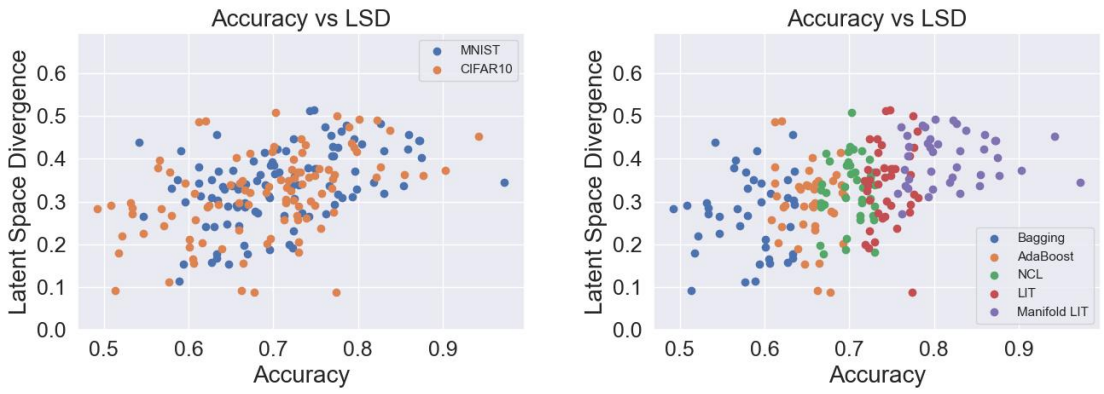
This gives us an idea on how to use GeValDi in practice. We get two important pieces of information from our MDS algorithm:

- The value of JS divergence our MDS points converge to after optimisation. Especially when we sample over a large number of points, we will get an accurate estimate of how different the predictions between classifiers are.
- By clustering the MDS points, we are able to visualise the types of data points our classifiers are uncertain about, and maybe use it as additional training data points. Also, we can compute the CSI of the clusters and that will allow us to understand how far apart the clusters are.

3.2.4 Latent Space Divergence

Noticing the relationship between the convergent value of JS divergence and the performance of ensembles in Figures 3.13 and 3.14, we suspect that convergent JS divergence value will correlate with the performance of ensembles. We call this Latent Space Divergence (LSD).

To test for any correlation between the ensemble performance and LSD, we train a large number of ensembles, using Bagging, AdaBoost, NCL, LIT and Manifold LIT, with different base models, on MNIST and CIFAR10, but with subsampling, such that the ensembles are each trained on a different (possibly overlapping) subset of the data. The main purpose of this experiment is to generate pairs of points (ensemble accuracy, LSD) on as a diverse set of situations as possible. Ideally, this would be done on a wide variety of datasets, ensemble methods and base models. However, given the restrictions on time and procuring vast varieties of datasets, we restrict ourselves sampling CIFAR10 and MNIST, and sampling model architectures for the base models.



(a) (b)

Figure 3.18: Scatter Plot of LSD against Accuracy, labelled by (a) Dataset, (b) Ensemble Method

Even though we see a visual correlation between the two, we can make our analysis rigorous by performing the Pearson Correlation Test [9]. With our sample correlation coefficient, r

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{(\sum_{i=1}^{N} x_i - \bar{x})(\sum_{i=1}^{N} y_i - \bar{y})} \tag{3.65}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ are the empirical averages, we test the null hypothesis, H_0, where

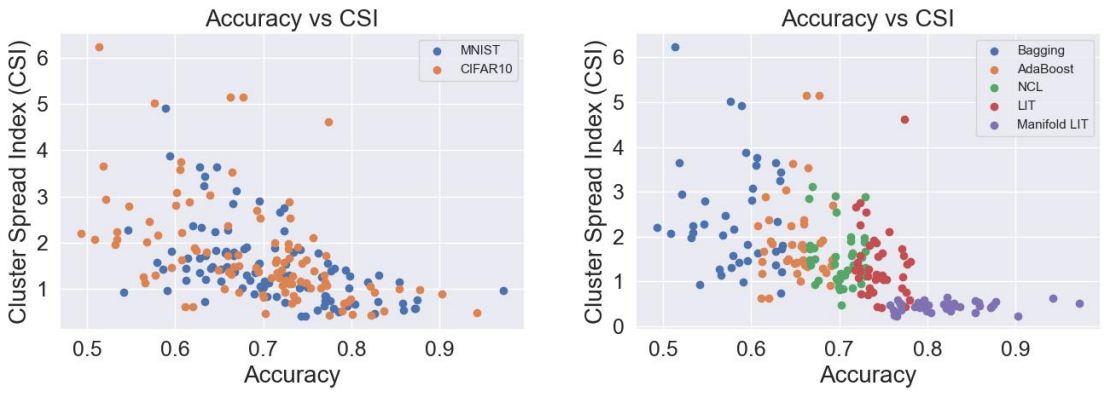
$$H_0 : r = 0 \tag{3.66}$$
$$H_1 : r \neq 0 \tag{3.67}$$

The test statistic, t, is then computed as

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{3.68}$$

This test statistic follows a t-distribution with $n-2$ degrees of freedom if the true correlation between data points $\{x_i, y_i\}_{i=1}^{N}$ is 0. Using this, we compute the sample correlation coefficient between accuracy and LSD to be 0.504, with a p-value of 7.56×10^{-15}, which allows us to reject the null-hypothesis, H_0, to conclude that we have statistically significant correlation between LSD and accuracy. However, it is important to note that this does not mean we have causation between LSD and accuracy. In other words, we cannot use this test to prove that having a higher LSD will cause the ensemble accuracy to be higher.

We can establish another relationship between ensemble accuracy and a metric we can calculate from our MDS points. Considering Figure 3.19, we observe that, with some noise, the higher the value of CSI, the worse the performance of the ensemble. This makes intuitive sense, because, when CSI is large, the regions of high uncertainty are larger and far apart, and this would negatively affect the ensemble performance.



(a) (b)

Figure 3.19: Scatter Plot of CSI against Accuracy, labelled by (a) Dataset, (b) Ensemble Method

Using the observed relationships between CSI, LSD and accuracy, we can construct a training protocol that will achieve better performance in our ensembles, as we will discuss in Section 3.3.

3.3 Latent Divergence Training

Given the correlation we observed between ensemble accuracy and Latent Space Divergence, we propose that we use Latent Space Divergence as part of our loss function in training. This is further motivated in a few ways: in Manifold LIT, we struggled to train VAEs or find pre-trained VAEs that were robust for complex datasets. This is due to GANs being more widely adopted, due to their superior performance. However, in Latent Divergence Training (LDT), we only sample points from a DGM and compute the divergence of the base models on those points. Therefore, we only need access to the model architecture, pre-trained weights and the gradients in the latent space. This can be done on GANs, VAEs and even Diffusion models [28].

Furthermore, given that MDS points served as indicators of regions in the data manifold where the ensembles are uncertain about/worse at predicting, we can add a loss on the MDS points to our training procedure to improve the ensemble performance. This resembles Active Learning [59], where the training procedure and the model being trained actively finds datapoints to query labels for, and to learn from. In LDT, we are allowing our ensemble to use MDS as the query points to train on, in addition to our training data, to further improve the performance. In order to be able to do this, we need to be able to find the true labels of those MDS points. Our solution for this is to use a conditional VAE or a conditional GAN, where the data we generate is from a particular label, and we can choose this label a-priori. Note that we can equivalently use an oracle/expert model to do so. This allows us to find MDS points and know the labels of those points. This gives us Latent Divergence Training (LDT), which is implemented in Algorithm 3.

Algorithm 3 Latent Divergence Training (LDT) Algorithm

Input: Set of Classifiers $\{f(\mathbf{x};\theta_m)\}_{m=1}^M$, Generative Model $p(\mathbf{x}|\mathbf{z})$, Training Data $\mathcal{D}$, Training Batches $\mathcal{B}$, Divergence Metric $D[p_1(\mathbf{y}|\mathbf{x})||p_2(\mathbf{y}|\mathbf{x})]$

Parameters: Number of Epochs N, Learning Rate γ, LDT Lambda λ_{LDT}, Training Batch Size B_t, Latent Batch Size B_l

Initialise : $\{\theta_m\}_{m=1}^M$

for $i \in \{1, ..., N\}$ **do**
  for $\mathcal{B} \in \mathcal{D}$ **do**
    $\mathcal{L}_{train} = \sum_{m=1}^M \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{B}}[\log p(\mathbf{y}|f(\mathbf{x};\theta_m)) + \lambda_{LDT} D[p_1(\mathbf{y}|\mathbf{x}||p_2(\mathbf{y}|\mathbf{x})]]$
    $\{\tilde{\mathbf{z}}\}_{i=1}^{B_l} \sim \text{MDS}(\{p(\mathbf{y}|\mathbf{x};\theta_m)\}_{m=1}^M)$
    $\{\tilde{\mathbf{x}}\}_{i=1}^{B_l} \sim p(\mathbf{x}|\tilde{\mathbf{z}})$
    $\mathcal{L}_{MDS} = \sum_{i=1}^{B_l} \log p(\mathbf{y}|f(\tilde{\mathbf{x}}_i;\theta_m))$
    $\mathcal{L} = \mathcal{L}_{train} + \mathcal{L}_{MDS}$
    $\{\theta_m \ = \theta_m + \gamma\nabla_{\theta_m}\mathcal{L}\}_{m=1}^M$
  end
end
return $\{f(\mathbf{x};\theta_m)\}_{m=1}^M$;

3.3.1 Toy Example

In order to investigate the usefulness of LDT, we test it on a toy example of synthetic data. In particular, we generate synthetic data that we think is going to be a hard problem for our benchmark models. We propose a dataset defined as per Equation (3.77). Simply described, the dataset is mostly similar to our previous mixture of Gaussians dataset, except now, 10% of the time, the data will be from a Gaussian centered near the opposite class. Combining all the Gaussians, we end up with a dataset that is not linearly separable, unlike our previous toy example.

$$p(\mathbf{x}|\mathbf{y}=0) = 0.9\mathcal{N}(\boldsymbol{\mu}_{1,1}, \boldsymbol{\Sigma}_{1,1}) + 0.05\mathcal{N}(\boldsymbol{\mu}_{1,2}, \boldsymbol{\Sigma}_{1,2}) \tag{3.69}$$

$$+ 0.05\mathcal{N}(\boldsymbol{\mu}_{1,3}, \boldsymbol{\Sigma}_{1,3}) \tag{3.70}$$

$$p(\mathbf{x}|\mathbf{y}=1) = 0.9\mathcal{N}(\boldsymbol{\mu}_{2,1}, \boldsymbol{\Sigma}_{2,1}) + 0.05\mathcal{N}(\boldsymbol{\mu}_{2,2}, \boldsymbol{\Sigma}_{2,2}) \tag{3.71}$$

$$+ 0.05\mathcal{N}(\boldsymbol{\mu}_{2,3}, \boldsymbol{\Sigma}_{2,3}) \tag{3.72}$$

$$p(\mathbf{y}=0) = p(\mathbf{y}=1) = \frac{1}{2} \tag{3.73}$$

$$\boldsymbol{\mu}_{1,1} = [-1, 1] \quad \boldsymbol{\mu}_{1,2} = [-1, -2] \quad \boldsymbol{\mu}_{1,3} = [2, 1] \tag{3.74}$$

$$\boldsymbol{\mu}_{2,1} = [1, -1] \quad \boldsymbol{\mu}_{2,2} = [1, 2] \quad \boldsymbol{\mu}_{2,3} = [-2, -1] \tag{3.75}$$

$$\boldsymbol{\Sigma}_{1,1} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad \boldsymbol{\Sigma}_{1,2} = \boldsymbol{\Sigma}_{1,3} = \frac{1}{4}\boldsymbol{\Sigma}_{1,1} \tag{3.76}$$

$$\boldsymbol{\Sigma}_{2,1} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad \boldsymbol{\Sigma}_{2,2} = \boldsymbol{\Sigma}_{2,3} = \frac{1}{4}\boldsymbol{\Sigma}_{2,1} \tag{3.77}$$

Training our benchmark models and LDT will elucidate how well our method works in encouraging diversity in the base models to improve ensemble performance. In particular,

we will be able to see the expected shortcomings of the benchmark models, and illustrate how LDT overcomes such shortcomings. In Figure 3.20, we show the base classifiers generated by an ensemble of two sigmoidal linear classifiers, with Bagging, AdaBoost, NCL and LDT. A few remarks on the results: Only LDT is able to fully separate the classes, whilst AdaBoost performs the best amongst the benchmark ensembles. We can understand the poor performance of bagging by considering how the base models are generated. To generate the different base models, we merely randomly sample from the training dataset. In doing so, we are preserving the ratios of our Gaussians, and therefore our base model is not able to focus on the small scale Gaussians that make this dataset not linearly separable. This results in the base models having minimal variation.

We see that NCL manages to improve upon bagging because we are actively encouraging the predictions of the base models to be negatively correlated, and this is achieved by 'increasing the angle' between the decision boundaries. However, this is still inadequate in generating useful diversity in base models.

AdaBoost performs the best amongst the benchmarks because it is designed to focus on the previous models' errors. In this case, the first base model will make errors on the small scale Gaussians, and the next model will preferentially train on those data points to get a better overall performance. Although this works in theory, we see that in practice, we need to somehow encourage even more diversity between the models to fully classify the data.

LDT works because of the fact that at each iteration, we are generating MDS points, which in this case are the misclassified points in the small scale Gaussians (because the base models agree on all other points), and the base models are trained on them. Furthermore, as part of training, we are also maximising the JS divergence between the base models, which results in increasing the 'angle' between the decision boundaries. Combining the two factors, we get that the base models end up focusing on both the main clusters and the small scale Gaussians.

3.3.2 Experimental Evaluation

In order to fully evaluate how effective LDT is in comparison to the benchmark methods and to Manifold LIT, we test it on CIFAR-10, and in particular, on ImageNet [12]. For CIFAR-10, we extend the two experiments performed in Section 3.1.6, and the results for LDT ensembles are depicted in Table 3.4. Furthermore, the evolution of ensemble performance across ensemble size is plotted in Figures 3.21 and 3.22. We draw a few observations based on these results. In both experiments, LDT performs better than all our benchmark models and it beats the performance of Manifold LIT. In particular, LDT requires a smaller sized ensemble to achieve comparable performance to benchmark methods, and this is due to the useful diversity we can generate in LDT base models via our JS divergence. Especially notable is the jump in accuracy in experiment 1 when we use just two base models in the ensemble. This illustrates the extent to which explicitly encouraging diversity, whilst additionally focusing on uncertain regions aids in ensemble performance.

In order to fully test LDT, we perform an experiment on ImageNet. Given that training classifiers on ImageNet from scratch is itself a challenging task, we attempt a different approach where we test the ability of our ensembling method to enourage diversity in

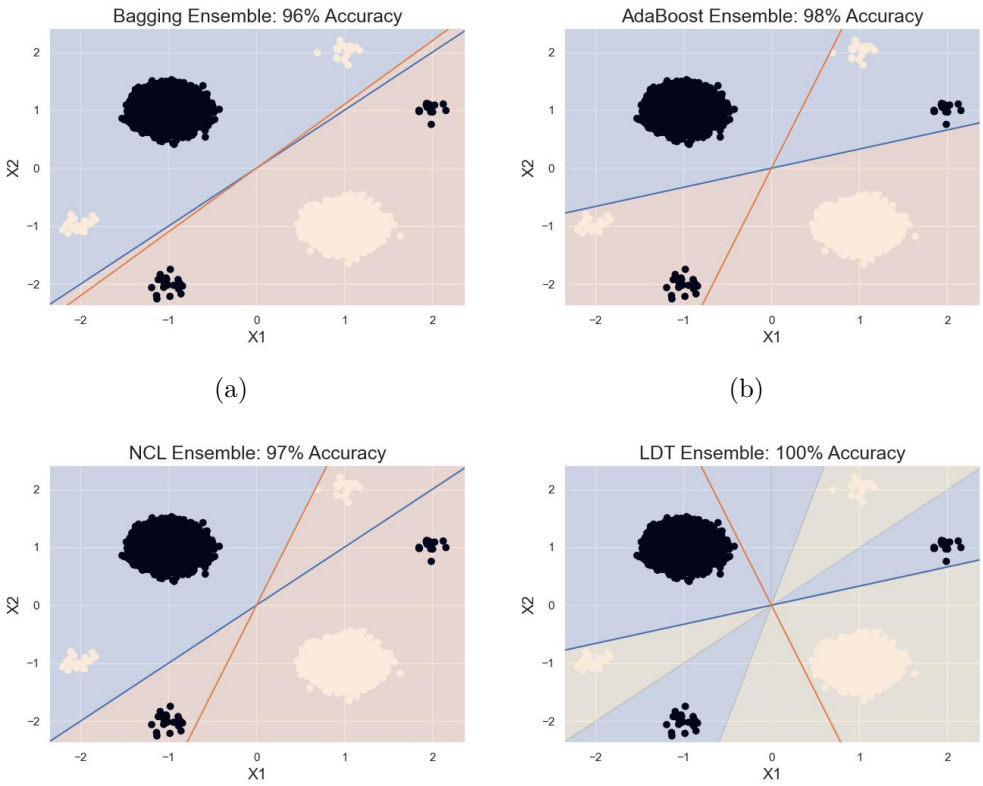
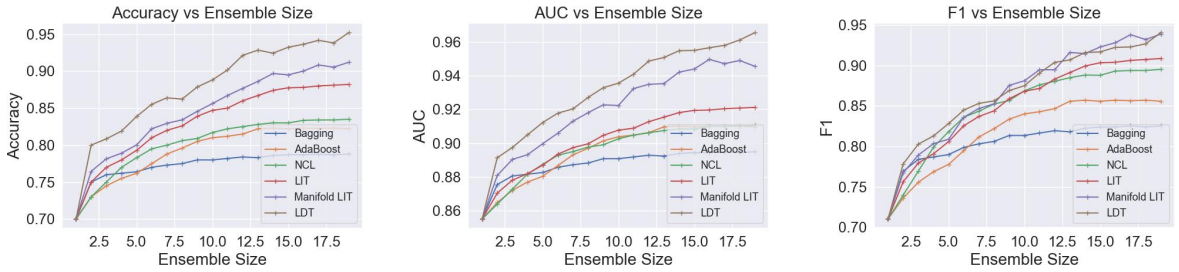


Figure 3.20: (a) Bagging, (b) AdaBoost, (c) Negative Correlation Learning (NCL), (d) Latent Divergence Training classifiers on Toy Dataset

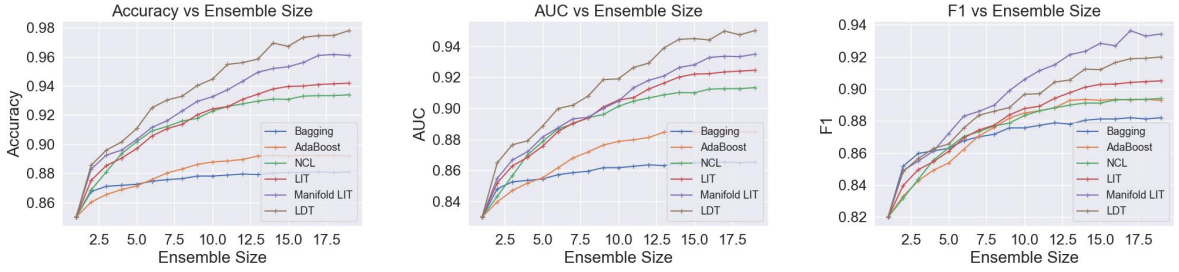
	Experiment 1			Experiment 2		
	Accuracy	**AUC**	**F1 Score**	**Accuracy**	**AUC**	**F1 Score**
Base Model	70.0	0.84	0.66	85.0	0.92	0.82
Bagging	78.8	0.89	0.79	88.1	0.94	0.89
AdaBoost	82.3	0.91	0.82	89.2	0.84	0.90
NCL	83.5	0.92	0.86	93.4	0.96	0.90
LIT	88.2	0.94	0.87	94.2	0.96	0.91
Manifold LIT	91.2	0.96	0.89	96.1	0.97	0.94
LDT	**95.2**	**0.98**	**0.95**	**97.8**	**0.98**	**0.95**

Table 3.4: Comparison of benchmark models and LIT, Manifold LIT, and LDT for CIFAR-10 classification in two scenarios, with best performance in bold.



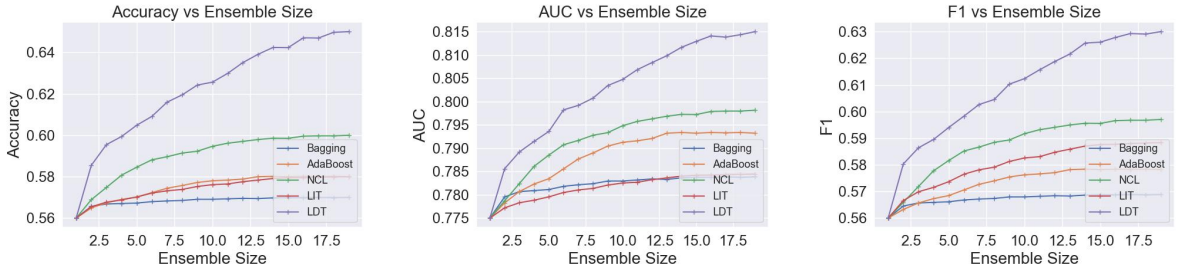
(a) (b) (c)

Figure 3.21: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for CIFAR10 Experiment 1



(a) (b) (c)

Figure 3.22: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for CIFAR10 Experiment 2



(a) (b) (c)

Figure 3.23: (a) Accuracy (b) AUC (c) F1 Score against Ensemble Size for ImageNet Experiment

pre-trained models. We train our ensemble with a pre-trained SqueezeNet [31] with an accuracy of $\sim 58\%$ (top 1 accuracy) with only 1.2million parameters, as the base model, with BigGAN [7], as our generative model to obtain our MDS. The results of ensemble performance metrics against the ensemble size is illustrated in Figure 3.23. A few remarks on the results: We notice that the benchmarks struggled to improve the performance of the base model via ensembling in this experiment. This is the result of the pre-trained nature of the base model, which makes introducing diversity between base models harder than base models trained from scratch. We can intuitively understand this by considering that over the course of training, the model gradients become smaller (as they tend to convergence) and using those gradients and training data metrics to introduce diversity will not be effective. Notably, bagging barely improved the performance, and in some runs, it actually decreased the performance of the base model. I think this can be attributed to overfitting, because by merely subsampling the dataset and allowing pre-trained models train on them, we are encouraging overfitting in the base models and that negatively affects the ensemble performance. NCL performs the best of the benchmarks because it explicitly encourages diversity in terms of predictive correlation and that allows meaningful diversity between the base models to be generated.

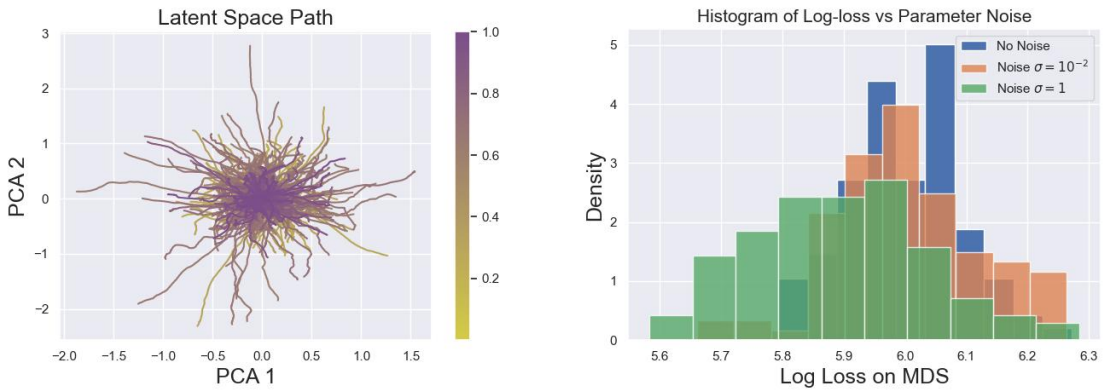
LDT performs vastly better than the benchmarks, and we are able to surpass the top performance of any of the benchmarks with just five base learners. This is evidence of the power of training on MDS points, because regardless of base learners being pre-trained, MDS points serve as novel, useful training data that allow the base learners to improve their predictions. Combined with explicitly encouraging diversity by maximising JS-divergence of the base model predictions, we are able to steer the base models to become usefully diverse. The result of this is that we can achieve close to $\sim 65\%$ top 1 accuracy with only $\sim$ 20million parameters, with a rather simple training procedure, contrasted to complex neural networks such as VGGs [63] requiring close to $\sim$ 100million parameters to achieve $\sim 70\%$ top 1 accuracy. Although $\sim 65\%$ top 1 accuracy is nothing remarkable given that we have cutting-edge models such as Regularised Neural Networks (RegNet), achieving $\sim 88\%$ top 1 accuracy [53], we think that LDT illustrates the use in using generative models to train and evaluate discriminative models.

Overall, from our experimental evaluations, we show that LDT outperforms all our benchmarks and Manifold LIT in both CIFAR-10 and ImageNet. Furthermore, LDT requires far fewer ensemble sizes to achieve equivalent performance in benchmark models. In particular, we are able to further improve the performance of pre-trained models by using them as base learners in LDT, whilst all benchmarks struggled to improve the pre-trained model performance.

3.3.3 Limitations of LDT

In this section, we discuss some of the limitations of our LDT training procedure for ensemble training. As discussed in Section 3.1.7, the reliance on a DGM is a critical factor we need to carefully consider. Although we can make similar theoretical argument on the sensitivity of LDT to the DGM parameters, we perform an experiment instead to illustrate the extent of this sensitivity.

First, it is useful to understand what parts of the LDT algorithm will be affected by changes to the DGM parameters. LDT includes two main components, computing a loss



(a) (b)

Figure 3.24: Effect of parameter noise on (a) latent space path during MDS optimisation with color bar indicating the noise variance as a proportion of parameter scale, (b) histogram of log-loss computed on the MDS points generated for a fixed pair of classifiers, snapshots at three noise levels.

(log-loss and JS-divergence) on training data batches, and computing a log-loss on MDS samples. Clearly, only the MDS samples, and the loss we compute on them will be affected by the parameter variations of DGM. One way we can test this is by performing a similar experiment as before, where we perturb the parameters with Gaussian noise with variance gradually increasing, as a proportion of the parameter scale. The results of this experiment is illustrated in Figure 3.24. A few comments on the results:

- Looking at the latent space path of the MDS algorithm across the range of noise variances, we see that surprisingly, the range of paths and the 'radius' of convergence is not largely affected by the noise variance. In particular, we see some instances where the latent space paths of low parameter noise experiments actually end up converging at a further distance than the experiments with large noise variance. However, we can understand this by considering the added parameter noise as reducing the information content of the gradients. Hence, when we perform gradient descent using noisy parameter values, we end up not traversing in an useful direction, but essentially being stuck near the initial value. Furthermore, the effect of our prior regulariser, λ_{prior}, is also a factor here. By having a regularisation term on the latent space prior density, we are making sure that, in the case that our gradients do not provide useful directions, we are at least still in large prior probability values, which are near the origin. This is exactly what we observe here.

- We get a better understanding of how the parameter noise affects LDT by considering the distribution of log-loss computations from the MDS points at various levels of noise. Looking at Figure 3.24(b), we see that two effects occur: the mean log-loss value drifts from the no-noise log-loss values, and the variance of the log-loss estimates increases with noise variance. Both effects are intuitive as we expect increasing the noise variance to perturb the log-loss values computed. However, one interesting observation is that the actual magnitude of these variations are relatively small in

comparison to the log-loss itself. Even when our noise variance is of the same order of magnitude as our parameters, we see that the drift in mean is only around ~ 0.2, which is negligible in the context of training.

One simple workaround to reduce the sensitivity of our LDT algorithm to the DGM parameter varaitions would be increase the MDS batch size, so that we are averaging the log-loss over a larger sample. However, this will increase the computation time, and with LDT, computation time is another critical issue. Especially when we were running experiments on ImageNet, the computation time and memory needs were very large, with some experiments taking close to 100 hours to run. Here, we propose some solutions to improve computation time, that however were not tested due to time constraints of the project

- Only sample MDS points every k-th iteration in our training. This is motivated by the fact that when the base models change slowly over iterations, the MDS samples should remain 'close' to each other, in a concept space distance sense. So, if we only perform the costly MDS optimisation every few iterations, we can vastly improve the computation time for the LDT algorithm. Furthermore, we can make k adaptive, where in the initial part of training, we compute MDS very infrequently, and as training progresses, we increase the frequency of MDS computation. This is motivated by the observation that in the beginning of training, training data already provided is enough to generate useful improvements in performance, and only as we tend to the plateau stage, we need MDS training data to further improve the performance.

- Everytime we compute MDS points, we cluster them and use the clusters to generate any further MDS points as training data. Combining with the previous improvement, we can still reap the benefits of optimising over MDS points at every iteration, but do so by sampling from the clusters rather than by performing the costly optimisation, and we essentially update our clusters every few iterations, according to our previous schedule.

4 Conclusions and Future Work

In this project, we shed light on the critical role of base model diversity and complexity in ensemble learning. First, we theoretically motivated the need for diverse base models in ensembles using the Bias-Variance-Covariance decomposition, and illustrated the lack of generalisation loss guarantees when model complexity increases. By exploring the current benchmark ensembling methods, we highlight their limitations and how it impedes the generalisation performance of ensembles.

As a starting point, we explore Local Indepedence Training (LIT), where we add a loss function that maximises the local independence to perturbations of base models. Although this method gives some useful intuition and performs well in toy datasets, we illustrate that it breaks down when we deal with more complex dataset, especially ones with large scalar curvature. Since real-world image datasets are shown to have a complex manifold structure, we require a better way to compute our independence loss along the manifold.

Manifold LIT solves this issue by utilising a latent space generative model such as VAEs, to transform the input space gradients into manifold gradients. By doing so, we are able to compute the true independence loss, and overcome the plateau we observed in the original LIT training for image datasets. This novel approach provides a more effective means of generating diverse models in the context of complex data manifolds. Furthermore, we empirically investigate this method on MNIST and CIFAR-10 to show superior performance over the benchmark models and LIT.

Although Manifold LIT performs better than our benchmarks, we are still unable to effectively probe the differences between the base models to understand where the diversity arises. Furthermore, as we discussed in Section 2.3, current methods of measuring diversity are inconsistent and do not provide any valuable intuition. Therefore, using GeValDi (Generative Validation of Discriminative Models), we are able to generate maximally different samples (MDS) that not only serve as useful visualisations of data points where the base models disagree, but also allow us to characterise regions of maximal uncertainty in ensembles. By looking at the convergent value of Jensen-Shannon (JS) divergence between the base model predictions, we are able to derive a diversity metric, Latent Space Divergence (LSD), that correlates well with generalisation performance of ensembles.

Leveraging the observed relationship between MDS points, JS divergence and generalisation performance of ensembles, we introduce a novel training procedure, Latent Divergence Training (LDT), which not only allows the base models to explicitly maximise diversity, but to also use MDS points as queries for additional training data by using conditional generative model labels. This alternating iterative procedure allows us to increase base model diversity by increasing JS divergence of their predictions and to improve performance in the regions of ensemble uncertainty characterised by MDS points. As experimentally evaluated on CIFAR-10 and ImageNet, we are able to beat the performance of the benchmarks and Manifold LIT with LDT. In particular, for ImageNet, we are able to use pre-trained classifiers as base models and encourage useful diversity between the base models via LDT, which was not effectively done in any of the other benchmark methods.

Our work, however, leaves a lot of scope for future evaluation and improvements, which we discuss below.

4.1 Future Work: Theoretical Foundations

Although we know that experimentally, LDT works well, it would be tremendously useful to establish some theoretical basis as to why the performance is expected. Given that we are optimising JS divergence, which stems from KL-divergence and information theory, we can attempt to derive some bounds on how the generalisation error will be affected by our ensemble training procedures. In particular, if we can relate using JS-divergence as the divergence metric, and increasing some functional distance between base models, as used in Section 2.2, we will be able to get interesting results that can further refine our method.

4.2 Future Work: LDT Experiments

Due to the computationally intensive nature of LDT, only minimal experiments could be performed. Given more time, we would test the effectiveness of the speed-up suggestions we gave in Section 3.3.3, and test the sensitivity of the method to DGM parameter noise over a large range of scenarios (different types of DGMs, datasets, families of classifiers, etc).

4.3 Future Work: GeValDi

In this project, we explored how we can use GeValDi to analyse the base models of ensembles. However, we can extend GeValDi to model selection by using MDS points as data for Human Selection Experiments (HSE). These selections serve as a method to choose between models that perform equally well on validation/test data, and have equivalent performance metrics. Furthermore, we can analyse the selection to reverse-engineer the selection criteria used by the human subjects. Doing so will allow us to imitate the selection criteria in our own training procedures.

4.4 Future Work: Base Model Weights

In some ensemble methods, the weights allocated to each base model is tuned to optimise some secondary objective. For example, in AdaBoost, the weights are tuned according to the error rates of the base models. However, in both Manifold LIT and LDT, we have used uniform weights, and did not explore how using different weighting schemes would improve the performance of the ensemble. An interesting experiment would be to consider how using weights proportion to pairwise JS divergence of base models would affect the performance. Furthermore, we can adopt a weighting scheme that adapts to base model complexity, inspired by [10].

Bibliography

[1] Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Zbc-ue9p_rE.

[2] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. 2020. doi: 10.48550/ARXIV.2006.06848. URL https://arxiv.org/abs/2006.06848.

[3] Sanjeev Arora, Nadav Cohen, Noah Golowich, Wei Hu, and Ruslan Salakhutdinov. Towards understanding the role of over-parametrization in generalization of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1007/bf00058655.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/a:1010933404324.

[7] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018. URL https://arxiv.org/abs/1809.11096.

[8] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: A survey and categorisation. *Information Fusion*, 6(1):5–20, 2005. doi: 10.1016/j.inffus.2004.04.004.

[9] George Casella and Roger L Berger. *Statistical Inference*. Duxbury Pacific Grove, CA, 2002.

[10] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1179–1187, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/cortesb14.html.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006. ISBN 978-0471241959.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[14] Thomas G. Dietterich. Ensemble methods in machine learning. *SpringerLink*, Jan 1970. URL https://link.springer.com/chapter/10.1007/3-540-45014-9_1#citeas.

[15] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Dover Publications, 1992. ISBN 978-0486656769.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[17] Lawrence C. Evans. *Partial Differential Equations*. 2010. ISBN 978-0-8218-4974-1.

[18] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15 (1):3133–3181, 2014.

[19] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49195-8.

[20] Salvador García, Alberto Fernández, Julian Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.

[21] Robert Geirhos, Carl R Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Exploring the landscape of spatial robustness. *International Journal of Computer Vision*, 127(6-7):764–777, 2019. doi: 10.1007/s11263-018-1083-3.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[24] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. doi: 10.1109/34.58871.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[27] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647.

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[29] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi: 10.1080/01621459.1963.10500830.

[30] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012. ISBN 978-0-521-38632-6.

[31] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and lt;0.5mb model size. 2016. doi: 10.48550/ARXIV.1602.07360. URL `https://arxiv.org/abs/1602.07360`.

[32] J.F. Kolen and J.B. Pollack. Backpropagation without weight transport. *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1991. doi: 10.1109/icnn.1994.374486.

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[35] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL https://proceedings.neurips.cc/paper_files/paper/1994/file/b8c37e33defde51cf91e1e03e51657da-Paper.pdf.

[36] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238, 1995.

[37] Harold W Kuhn and Albert W Tucker. Nonlinear programming. *Proceedings of the second Berkeley symposium on mathematical statistics and probability*, 481:481–492, 1951.

[38] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[39] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003. doi: 10.1023/a:1022859003006.

[40] Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/7bb060764a818184ebb1cc0d43d382aa-Paper.pdf.

[41] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 1998.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[43] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999. doi: 10.1016/s0893-6080(99)00073-8.

[44] Aravindh Mahendran and Andrea Vedaldi. Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1412.0035*, 2015.

[45] V. A. Marchenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematical Proceedings of the Academy of Sciences of the USSR*, 1(2):457–483, 1967. doi: 10.1070/SM1967v001n02ABEH001994.

[46] Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141:148–188, 1989.

[47] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

[48] Vivek Palaniappan, Matthew Ashman, Katherine M. Collins, Juyeon Heo, Adrian Weller, and Umang Bhatt. Gevaldi: Generative validation of discriminative models. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*, 2023. URL https://openreview.net/forum?id=2BZDR5JMMS_.

[49] Vivek Palaniappan, Matthew Ashman, Katherine M. Collins, Juyeon Heo, Adrian Weller, and Umang Bhatt. Gevaldi: Generative validation of discriminative models, 2023. URL https://openreview.net/forum?id=zwywBS3GyFs.

[50] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch, 2017.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[52] Mark A. Pinsky and Daniel W. Stroock. *Introduction to Fourier Analysis and Wavelets*. 2002. ISBN 978-0-8218-2957-4.

[53] Ilija Radosavovic, Vineel Pratap Kosaraju, Amir Sadeghian, Andrew Hsu, Andrew Sornborger, and Jonathon Shlens. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10785–10794, 2020.

[54] Lior Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009.

[55] Andrew Slavin Ross, Weiwei Pan, Leo Anthony Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. *CoRR*, abs/1911.01291, 2019. URL `http://arxiv.org/abs/1911.01291`.

[56] Andrew Slavin Ross, Weiwei Pan, Leo Anthony Celi, and Finale Doshi-Velez. Ensembles of locally independent prediction models. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. URL `https://arxiv.org/abs/1911.01291`.

[57] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[58] Robert E. Schapire. The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification*, page 149–171, 2003. doi: 10.1007/978-0-387-21579-2_9.

[59] Burr Settles. *Active Learning*. Morgan & Claypool Publishers, 2009.

[60] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[61] Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. The riemannian geometry of deep generative models. 2017.

[62] Ron Shoham and Haim Permuter. Amended cross-entropy cost: An approach for encouraging diversity in classification ensemble (brief announcement). *Lecture Notes in Computer Science*, page 202–207, 2019. doi: 10.1007/978-3-030-20951-3_18.

[63] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[64] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009. ISBN 9780980232714. URL `https://books.google.com/books?id=HH6aBgAAQBAJ`.

[65] Joshua B Tenenbaum, Vin De De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[66] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *ICML Deep Learning Workshop*, volume 2, 2015.

[67] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.

Risk Assessment

- As the project was entirely computational, the only risks were eyestrain and repetitive strain injury from excessive screen-time and typing respectively. Both of these were mitigated by taking regularly scheduled breaks, maintaining appropriate posture and engaging in regular exercise. Moreover, the workstations were set up to conform with the Display Screen Equipment regulations.
- COVID-19 has minimal impact on this project and I was able to meet with my supervisor both in-person and virtually, as permitted by the prevailing COVID-19 regulations.