Exploring Preferential Label Smoothing for Neural Network based classifiers

by

Paritosh Goyal

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Overfitting is a phenomenon when a machine learning system learns the patterns in training data so well that it starts to inauspiciously affect the model performance on unseen data. In practice, machine learning systems that overfit are not deployable, rather systems that generalize well and do well on both train and test data are deployed. One of the strategies used to prevent overfitting and help models generalize well is regularization. For neural network based machine learning systems, regularization can be applied using any of: the neural network architecture, the loss function and the training algorithm. One of the losses used to train neural network based classifiers is Cross Entropy Loss (CE). When using such a loss, the loss for a given data sample is computed solely using that sample's ground truth label, i.e., focusing on the ground truth label and neglecting the effect of other labels, this makes the classifier overconfident for the data sample on one ground truth label and degrades generalization. One method of regularization is to take some of the concentration (called Smoothing Ratio (SR)) from the data sample's ground truth label and distribute it uniformly among all the other labels. This method is called label smoothing and has been found to be quite effective. For brevity, we call the approach of distributing SR uniformly Uniform Label Smoothing (ULS).

In this work, we explore what happens if we distribute the SR to the non-ground truth labels based on how closely they are related to the ground truth label. The relation between the labels may come from an external source -

learnt from external data or provided by a subject matter expert. We call this approach of distributing the SR based on relationship between labels **Preferential Label Smoothing (PLS)**. PLS represents a more unified approach to label smoothing because even ULS is a special case of PLS.

Previous works on ULS suggest that ULS becomes redundant when the number of labels is high. Consider the case when there are only two labels (i.e., binary classification) then there is no point of using PLS. So, we investigate the effects of PLS when the number of labels in the dataset is high.
Another gap that we study in this work is about the effects of PLS and ULS on the training dynamics and how training dynamics differ when no label smoothing is used.
We demonstrate our study on image classification and text classification. Experimenting on text classification fills in one more gap in the previous works, since ULS has not been studied in the context of text classification.

Preface

No part of this thesis were published anywhere. Parts of this thesis may be restructured for submission to some conference.

To all the giants whose shoulders I have stood upon

To my family for their unconditional support, love and care.

Happiness is when what we think, what we say and what we do are all in harmony.

– Atman Upanishad

Acknowledgements

I am deeply grateful to my supervisor Dr. Osmar Zaiane for his kindness, care, guidance and belief in me. He has supported me in multiple facets during my time as a student, apart from remarkably guiding me in my research. He was very understanding and patient when I had to deal with difficult circumstances related to non-academic life. He never stopped motivating me whenever I was pushed against the wall. He has always provided me with ideas and insights which helped in shaping this research to a closure. I will always carry forward the teachings and lessons that Osmar has taught me.
I would like to thank Dr. Amine Trabelsi for helping me with my research, giving feedback on this dissertation and pushing me to think beyond my limits and apprehensions.
I appreciate my examining committee member Dr. Matthew Guzdial for taking time to read my manuscript and help it improve through his feedback. I am thankful to Dr. Mohammad R. Salavatipour for organizing my MSc. exam smoothly. Many thanks to AMII (Alberta Machine Intelligence Institute) and University of Alberta for providing the required funding through teaching assistantships and research assistantships.

I am glad to have the friends in Edmonton and across the world who have helped maintain my sanity through these years. I am thankful to many people in our department and AMII - my friends, other grad students, support staff and faculty for always being helpful over the years and providing a supportive and fun environment for research.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Thesis Statement . . . 4
- 1.3 Thesis Contributions . . . 5
- 1.4 Thesis Organization . . . 6

2 Background Material **7**
- 2.1 Classification problem . . . 7
- 2.2 Training a Classifier . . . 8
  - 2.2.1 Artificial Neural Networks . . . 9
  - 2.2.2 Loss Function . . . 9
  - 2.2.3 Training Algorithm . . . 13
- 2.3 Generalization and stability . . . 13
- 2.4 Problem of Overfitting . . . 15
- 2.5 Regularization . . . 16
  - 2.5.1 Regularization for deep learning . . . 16
  - 2.5.2 L2 Regularization . . . 17
  - 2.5.3 L1 Regularisation . . . 17
  - 2.5.4 Dropout . . . 17
  - 2.5.5 Label smoothing . . . 18

3 Label Smoothing **19**
- 3.1 Introduction to Label Smoothing . . . 19
- 3.2 Formulation for Label Smoothing . . . 21
  - 3.2.1 Advantages of ULS . . . 22
  - 3.2.2 Formulation for ULS . . . 22
- 3.3 Preferential Label Smoothing . . . 24
  - 3.3.1 Formulation for PLS . . . 25
- 3.4 Related work . . . 27
  - 3.4.1 Label smoothing and Knowledge Distillation . . . 29
- 3.5 Components of Our Study . . . 31
  - 3.5.1 Label Smoothing and Gradients . . . 31
  - 3.5.2 Generalization and Learning Rate . . . 31
  - 3.5.3 Effect of label smoothing on increasing and decreasing number of classes in a dataset . . . 32

4 Label Smoothing for Image Classification **33**
- 4.1 Introduction . . . 33
- 4.2 Datasets . . . 34
- 4.3 Approaches of PLS for image classification . . . 34
- 4.4 Neural Models for Image Classification . . . 40
- 4.5 Experiments and results . . . 44
  - 4.5.1 Experimental Setup . . . 44

4.5.2 Impact of label smoothing on model performance . . . 46
4.5.3 Training Dynamics 48
4.6 Conclusion 60

5 Label Smoothing for Text Classification 62
5.1 Introduction 62
5.2 Word Embeddings 63
5.3 Neural Models for Text Classification 64
5.3.1 Recurrent Neural Network 64
5.3.2 Long Short-Term Memory 64
5.4 Datasets 66
5.5 Approach of PLS 68
5.6 Experiments and results 71
5.6.1 Performance Criterion 71
5.6.2 Model Performance 72
5.6.3 Effect of label smoothing on changing the number of classes in dataset 73
5.7 Conclusion 74

6 Conclusion 76
6.1 Summary 76
6.2 Future Work 78

References 80

Appendix A Additional Info 86
A.1 CIFAR-100 Class Labels 86

List of Tables

4.1 Model performance under different label smoothing approaches CIFAR-10 47
4.2 Model performance under different label smoothing approaches CIFAR-100 47
4.3 Top - 1 classification accuracy as reported by Müller *et al.* [33] 48

5.1 Statistics of emotion classification datasets 67
5.2 Model performance of LSTM model on different datasets under different label smoothing approaches 72
5.3 Model performance of LSTM model on datasets sampled from CBET dataset with 3 classes trained with different label smoothing approaches 73
5.4 Model performance of LSTM model on datasets sampled from CBET dataset with 5 classes trained with different label smoothing approaches 74
5.5 Model performance of LSTM model on datasets sampled from CBET dataset with 7 classes trained with different label smoothing approaches 74

A.1 List of Classes in CIFAR-100 86

List of Figures

1.1 Uniform Label Smoothing Example with Goemotions Dataset 4

3.1 No Label Smoothing (NoLS) example with emotion classification task . . . 20
3.2 ULS example with emotion classification task . . . 20
3.3 PLS example with emotion classification task . . . 25
3.4 PLS example for emotion relation Δ matrix for emotion classification task . . . 26
3.5 PLS example for emotion normalized relation matrix θ for emotion classification task . . . 27
3.6 Example of a teacher distilling knowledge to train a student model 29

4.1 Representation of the labels in CIFAR-10 dataset using GloVe embedding . . . 37
4.2 Semantic similarity among the labels in CIFAR-10 dataset . . 38
4.3 θ matrix for Semantic Label Smoothing (SEMLS) . . . 39
4.4 Convolution operation on an input image of $5 \times 5 \times 1$ with a kernel of $3 \times 3 \times 1$ . . . 41
4.5 Pooling - Average and Max . . . 42
4.6 Complete architecture of Convolution Neural Network . . . 43
4.7 A building block for Residual learning . . . 44
4.8 ResNet34 architecture . . . 45
4.9 Multiple ResNet architectures . . . 46
4.10 Generalization error for ResNet18 trained on CIFAR-10 with different learning rates. . . . 50
4.11 Generalization error for ResNet34 trained on CIFAR-10 with different learning rates. . . . 51
4.12 Generalization error for ResNet18 trained on CIFAR-100 with different learning rates. . . . 52
4.13 Generalization error for ResNet34 trained on CIFAR-100 with different learning rates. . . . 53
4.14 Gradient norms obtained while training ResNet18 on Cifar-10 dataset across three smoothing at four different learning rates. 56
4.15 Gradient norms obtained while training ResNet34 on Cifar-10 dataset across three smoothing at four different learning rates. 57
4.16 Gradient norms obtained while training ResNet18 on Cifar-100 dataset across three smoothing at four different learning rates. 58
4.17 Gradient norms obtained while training ResNet34 on Cifar-100 dataset across three smoothing at four different learning rates. 59

5.1 Recurrent Neural Network . . . 65
5.2 Recurrent Neural Network Unrolled . . . 65
5.3 LSTM Block . . . 66

5.4 The representation of labels in different emotion classification datasets using SEMLS. These were sampled down using PCA from 50 dimensional GloVe embeddings to 2 dimensions for plotting. 68
5.5 Semantic similarity for SEMLS among the labels in the emotion classification datasets, the Δ matrices 69
5.6 θ matrices for SEMLS with emotion classification datasets . . 70

List of Algorithms

1 Stochastic Gradient Descent($\mathcal{L}, \mathbf{X}, \mathbf{t}$) 13

2 Stochastic Gradient Descent with ULS ($\mathcal{L}, \mathbf{X}, \mathbf{t}$) 24
3 Stochastic Gradient Descent with PLS ($\mathcal{L}, \mathbf{X}, \mathbf{t}$) 27

Glossary

Aritificial Neural Network (ANN)
  8, 9, 16

Aritificial Intelligence (AI)
  1, 2, 8

Cluster Label Smoothing (CLS)
  35, 46, 47, 60, 76, 77, 79

Convolutional Neural Network (CNN)
  33, 40, 41, 42, 43

Cross Entropy Loss (CE)
  ii, 11, 18, 19, 21, 22, 23, 26, 44, 79

Knowledge Distillation (KD)
  29

Label Smoothing Regularization (LSR)
  23, 24, 26, 27, 28, 31

Long Short-Term Memory (LSTM)
  64, 66, 71, 74

Neural Networks (NN)
  2, 3, 5, 9, 11, 13, 15, 16, 17, 18, 31, 33, 40, 64, 78, 79

No Label Smoothing (NoLS)
  xi, 20, 21, 22, 23, 26, 31, 32, 46, 47, 48, 49, 54, 55, 60, 61, 72, 73, 74, 77, 78

Preferential Label Smoothing (PLS)
  iii, viii, ix, xi, xiii, 3, 6, 19, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 46, 47, 48, 60, 61, 68, 74, 76, 77, 78, 79

Recurrent Neural Network (RNN)
  64, 66

Semantic Label Smoothing (SEMLS)
  xi, xii, 39, 46, 47, 48, 49, 54, 55, 60, 61, 68, 69, 70, 72, 73, 74, 76, 77, 78, 79

Smoothing Ratio (SR)

ii, iii, 5, 21, 22, 23, 24, 26, 27, 35, 36, 39, 48, 55, 72, 78

Stochastic Gradient Descent (SGD)

13, 15, 23, 24, 26, 27, 31, 44

Uniform Label Smoothing (ULS)

ii, iii, viii, xi, xiii, 20, 21, 22, 23, 24, 29, 31, 32, 46, 47, 48, 49, 54, 55, 60, 61, 72, 73, 74, 77, 78

Chapter 1

Introduction

1.1 Motivation

The quest for understanding the human and the animal intelligence has been in existence since the living organisms started thinking about thinking. For the entire animal kingdom including homosapiens, this quest continues - parents try to understand their young offsprings' behaviour to help facilitate their daily activities, hunters try to understand the behaviour of their prey when hunting, hunters try to understand the behaviour of other members in their community when they hunt together in groups and so on.

While we wonder about the beginning of this quest for intelligence, a notable inception from the scientific community occurred on August 31, 1955 when John McCarthy, Marvin L. Minsky, Nathaniel Rochester and Claude E. Shannon [29] proposed a 2 month long summer 1956 research project at Dartmouth College. The document was titled “A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence”. Since then Aritificial Intelligence (AI) research community has endured a bumpy journey and survived two major droughts of funding, known as “AI winters”, which occurred in 1974 – 1980 and 1987 – 1993. It was after Imagenet competitions [4], [38] which started in 2010s, that applied deep learning algorithms ([23], [22]) started to catch the eyes of the wider community due to availability of higher compute power and big data.

The value of this quest for intelligence, however does not rest upon its

history but upon the vital importance it brings to science, technology and our everyday life. Today, as a society, we look up to artificial intelligence for several potential advancements in sciences - controlled nuclear fusion [3], drug discovery [18], clinical and emotionally intelligent chatbots [5], [48], [16]; autonomous driving [51], smart homes [25], defect detection, smart digital legislation and social equality to AI-powered governance [9], [36].
Machine Learning is at the core of building many AI systems. Machine learning focuses on using data and algorithms to learn from data with the goal of predicting unknown data.

In this work, we address the fundamental problem of overfitting that any researcher, engineer, or a developer may encounter when trying to build or upgrade a machine learning system. An example of overfitting from our real life would be - let's say a person grows up in a country 'A', gets accustomed to (or gets 'trained') understanding social gestures (language, tone, humor, facial expressions), then moves to a different country 'B' after 25 years and finds out that the social gestures and interactions are completely different, for the person it seems that their intuition on how to judge these is wrong. In this case, the person is overfitted to perceive gestures in country 'A' and is not able to understand gestures in country 'B', so the person cannot generalize well. Perhaps a solution for the person could be to read some books from country 'B', or watch some tv series or movies from country 'B'.

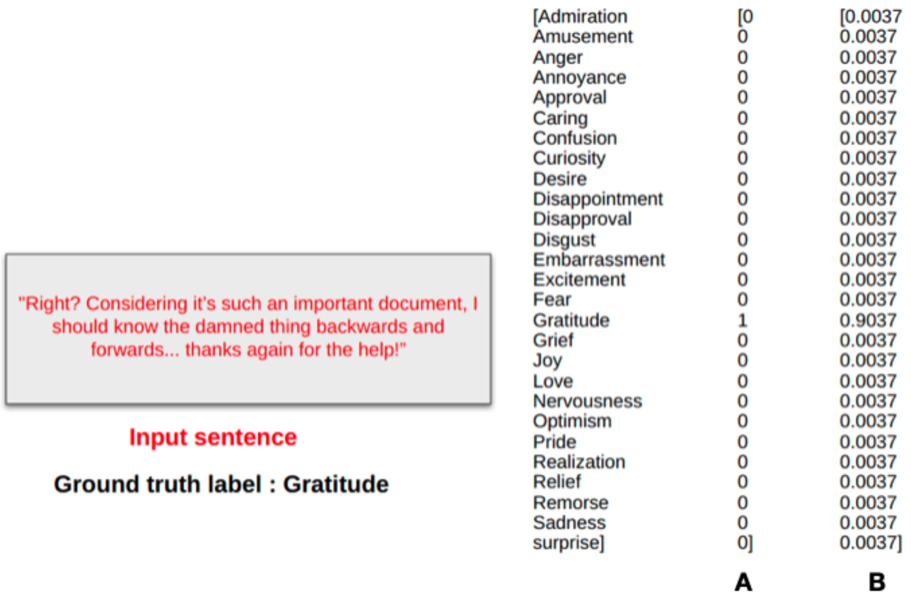
Similar to the overfitted person in our example, an overfitted machine learning model does not generalize well when unseen data comes in. A useful machine learning model must be able to generalize well. There are many methods to prevent overfitting and therefore help models generalize better and eventually improve a model's performance in the task it is intended to do. Szegedy *et al.* [45] proposed one such mechanism for preventing overfitting and overconfidence in Neural Networks (NN) called label smoothing. The idea is to use soft labels instead of hard labels (one-hot coding) while training a model, i.e. distributing some concentration from the ground truth label to all the labels uniformly rather than having full concentration on one ground truth label and no concentration on non ground truth labels.

The method proposed by Szegedy *et al.* [45] has a shortcoming that we illustrate through an example of emotion classification[1] in text classification in Figure 1.1. Given a sentence, the goal is to find out which emotion class the sentence belongs to. In Figure 1.1, the ground truth vector A is a one-hot coded vector while the embedded vector B used for training is the result of the label smoothing operation proposed by Szegedy *et al.* [45]. Although this type of arrangement would help in preventing overfitting, it unnecessarily gives equal importance to some classes which do not have any relationship with the true label. In Figure 1.1, if the true label of a sentence is gratitude then it is more likely that the sentence may be closer to joy or excitement rather than disgust or remorse.
The approach suggested by Szegedy *et al.* [45] appears to indiscriminately (equally) distribute the label concentration among all the non-ground truth labels. From the example in Figure 1.1 it is intuitive that an approach where the label concentration is distributed based on relationships between the ground truth label and non ground truth label might be more favourable. Part of this thesis proposes the idea of distributing concentration to non-ground truth labels based on how close or far in relationship non-ground truth labels are from the ground truth label, we call this approach as **PLS (Preferential Label Smoothing)**.

Prior studies of label smoothing do not involve text classification, which is one of the contribution in our work. Our experiments involve both text classification and image classification, thus bridging the gap in past studies.

Prior works in label smoothing study its effects on knowledge distillation, model calibration, a model's robustness to noise, and a model's robustness to adversarial attacks but do not discuss the effects of label smoothing on **training dynamics**. For study on training dynamics, we study how a change in learning rate combined with different label smoothing approaches affects the generalization error of a neural network. Another facet of training dynamics, we study how label smoothing affects the gradient norm while training a NN based classifier, since gradients during training has a connection with

[1]For more details please refer to Chapter 5


"Right? Considering it's such an important document, I should know the damned thing backwards and forwards... thanks again for the help!"

[Admiration	[0	[0.0037
Amusement	0	0.0037
Anger	0	0.0037
Annoyance	0	0.0037
Approval	0	0.0037
Caring	0	0.0037
Confusion	0	0.0037
Curiosity	0	0.0037
Desire	0	0.0037
Disappointment	0	0.0037
Disapproval	0	0.0037
Disgust	0	0.0037
Embarrassment	0	0.0037
Excitement	0	0.0037
Fear	0	0.0037
Gratitude	1	0.9037
Grief	0	0.0037
Joy	0	0.0037
Love	0	0.0037
Nervousness	0	0.0037
Optimism	0	0.0037
Pride	0	0.0037
Realization	0	0.0037
Relief	0	0.0037
Remorse	0	0.0037
Sadness	0	0.0037
surprise]	0]	0.0037]
	A	B

Figure 1.1: An example of emotion classification from a given sentence. There is an input sentence with ground truth label *Gratitude* There are a total of 27 plausible classes of emotion a sentence can get assigned to. *A* shows the one-hot coded vector of the ground truth. *B* shows the ground truth vector after label smoothing as suggested by Szegedy *et al.* [45]

generalization[2]. This is another research gap that we study in this thesis.

In the subsequent sections, we outline the thesis content and organisation in a more formal manner.

1.2 Thesis Statement

This thesis seeks to answer the following question:

How do preferential label smoothing and uniform label smoothing affect model performance and training dynamics compared to no label smoothing?

To attain our objective, we explore the use of preferential label smoothing for the problems of image classification and text classification.

[2]Check Chapter 2 for the connection between gradient norm and generalization

1.3 Thesis Contributions

The regularization technique of assigning concentration uniformly to the different labels at the time of training is called label smoothing. Many works in the literature today use label smoothing during training. In our investigation, there are few works that, when training from scratch, use different concentrations for different labels depending on the distance of each label from the ground truth label. To the best of our knowledge, prior works do not focus on text classification. We also bridge the gap in label smoothing research, where the effects of label smoothing on the training dynamics of the NN are missing. In this research, we only consider datasets that fall under supervised classification problems and have more than two classes. For binary classification, doing preferential label smoothing is not reasonable since there is only one extra label than the true label and that label gets full SR. Here are the questions that we study in this work:

1. **Does preferential label smoothing help improve model performance?** Label smoothing has helped in improving the performance of NN for image classification [45], and so we test whether preferential label smoothing helps in improving the performance on image classification and text classification.

2. **Does label smoothing or preferential label smoothing affect training dynamics of the NN?**. There is no prior work addressing this question. We use two approaches to study this - (*i*) effect of changing learning rate with label smoothing on the generalization error, (*ii*) length of gradients while training with different label smoothing approaches. We experiment on the image classification task for this problem.

3. **How does preferential label smoothing affect model performance when we change the number of labels (classes) in the dataset?** As stated earlier, doing preferential label smoothing for a binary classification case is not reasonable. In comparison, if there are

many labels, then the labels far from the true label get very little concentration and the labels closer to the ground truth get more concentration. So there might be an observable difference. By addressing this question we can identify whether preferential label smoothing (or label smoothing) is good for datasets which have a large number of labels or smaller number of labels. We experiment on the text classification task to study this problem.

1.4 Thesis Organization

The following chapters of this thesis are laid out in the following manner.

Chapter 2: Background

In this chapter we give an overview of classifiers, stability and generalization, overfitting, and regularization. This chapter is a bridge before we introduce label smoothing formally in Chapter 3.

Chapter 3: Label Smoothing

In this chapter we cover label smoothing and formulate our preferential label smoothing technique mathematically. We discuss past related works in label smoothing and discuss the components of our study to answer the questions in Section 1.3.

Chapter 4: Image Classification

We formally introduce the problem of image classification. We explain how we formulate our experiments and the approaches for PLS. We share our experimental setup and results.

Chapter 5: Text Classification

We formally introduce the problem of emotion classification in text classification. We explain how we formulate our experiments and the approach for PLS. We share our experimental setup and results.

Chapter 6: Conclusion

In this chapter we summarize our findings and discuss plausible future works.

Chapter 2

Background Material

In this chapter, we contextualize the use of label smoothing. First, we introduce classification, then we discuss how to train a classifier and the basic components required to train a classifier. We discuss generalization and overfitting, and describe how regularization is helpful to overcome overfitting and helps models generalize better. We initiate the discussion for label smoothing in this chapter and continue in the following chapters.

2.1 Classification problem

Classification is a predictive modeling problem where the goal is to predict a class label for a given data sample of the input data. Examples of classification problems can be - (i) given an email, classify if it is a spam or not, (ii) given textual dialogue, classify which emotion it expresses (from anger, joy, happy, sad or sorrow), (iii) given a handwritten character, classify it as one of the known characters. For solving a classification problem, we require a *dataset* with examples of inputs and target labels from which to learn. A classification model will use the training dataset and will calculate how to best map examples of input data to specific class labels. The training dataset must be sufficiently representative of the problem and have examples of each class label. There are three main types of classification tasks that a designer of a classifier may face:

- **Binary Classification:** There are two possible classes and the goal is to predict to which class a data sample belongs. Example: We are the admin of an e-mail server, and want to decide whether a given e-mail is spam.

- **Multi-Class Classification:** These classification tasks have more than two class labels involved and only one class label can be predicted for each data sample. Example: Given the image of an animal, our goal is to tell which animal it is (out of cat, dog, or horse).

- **Multi-Label Classification:** These classification tasks have more than one class label, where one or more class labels may be predicted for each data sample. Consider the example of photo classification, where a given photo may have multiple objects in the scene and a model may predict the presence of multiple known objects in the photo, such as "bicycle","apple", "person", etc.

The AI community uses different models for solving the classification problems that we discussed above. Among the multiple models of classification we discussed above, in this thesis we focus on Aritificial Neural Network (ANN) based classifiers for a variety of problems and hence we discuss training a ANN classifier below.

2.2 Training a Classifier

In the previous section we introduced classification problem. Our discussion here will be centered around ANN based classifiers because in this thesis our analysis is centered on them. In this section we discuss about the basic components that are required to train a classifier. These basic components are - the classifier model, loss function and training algorithm. Below, we give a brief overview of ANN in Section 2.2.1. We then discuss the components that are required to train a classifier. We then also describe the performance criteria to evaluate a classifier's performance once it is fully trained.

2.2.1 Artificial Neural Networks

An ANN is a brain inspired model - there are nodes in the network that are connected to each other similar to how the neurons in our brain are connected. The nodes are arranged in layers stacked one after the other. The number of nodes in each layer and the number of layers in the NN depend on what kind of application we are targeting. The first layer is called the *input layer* and the final layer is called the *output layer*. There are one or more hidden layers between the input layer and the output layer. In a fully connected NN, each node in each layer is connected to every other node in its adjacent layers. For any NN, how the nodes are connected to other nodes depends on the application of the NN. The connection between the nodes has an assigned weight w, the weight w controls the contribution of each input node to the output value. When a data input x, is given into the input layer, output of the input layer gets computed via matrix product of x and w, along with an additional bias term. The output of the input layer is fed into the hidden layers which modify this output term by passing it through an *activation function*. Succeeding layers follow the same process - compute the weighted sum of the input, pass through an activation function, and output to the next following layer. ANN can have different architectures depending on the problem they are intended to solve. We list different ANN architectures below but we will go into details of these in the Chapter 4 and Chapter 5

- Convolutional Neural Networks
- ResNet
- RNN
- LSTM

2.2.2 Loss Function

Let's say for a given classification problem we are given a training dataset, a test dataset, and we have chosen a classifier model. Now the question would be: how do we train the model? An intuitive approach would be - at a certain time step during training, quantify how far away the model is from the expected behaviour and then bring the model closer to the expected behaviour in the subsequent next time step of training. In order to quantify how far away the model is from the expected behaviour, at each training step, a loss function is used. To introduce loss functions, we first setup some notations. We first introduce loss functions using the case of binary classification and then we extend it to multi-class classification. Given a set of training examples $\{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$, where the $\mathbf{x}^{(i)}$ are vector-valued inputs, and $t^{(i)}$ are binary-valued targets. We would like to learn a binary classifier, where we learn the parameters $\mathbf{w}$ of the classifier, such that:

$$y = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Loss functions for the above problem would be:

0-1 Loss or Classification Loss- When solving the problem of classification, intuitively one natural criterion would be to minimize the number of mis-classified training examples, 0-1 loss or classification loss quantifies the number of mis-classified training examples. Below, we formalize 0-1 loss:

$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{otherwise} \end{cases} \tag{2.2}$$

where y is the predicted label and $\mathcal{L}$ is the loss. The loss function is the average of loss over the training examples, which corresponds to fraction of misclassified examples. This 0-1 loss is discontinuous and non-differentiable so we cannot compute the gradient with it, so it is not optimizable. So, we cannot use this loss for training our classifier. Ideally, we would want to use 0-1 loss as our loss function but since it is not optimizable we have to use a surrogate loss function.Using the classification error as a loss function for training is difficult so we use surrogate loss function like 'Cross-entropy loss'.

Cross Entropy Loss- This loss function is useful for training when the goal is to optimize for a classification problem. It was introduced by Bridle [2]. We use Cross Entropy loss in this thesis for our analysis. The formulation for CE is:

$$\mathcal{L}_{\text{CE}}(y,t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases} \tag{2.3}$$

The formulation without case notation is:

$$\mathcal{L}_{\text{CE}}(y,t) = -t\log y - (1-t)\log(1-y) \tag{2.4}$$

Now consider the case where we have a training example with ($t = 1$) but the classifier classifies it as ($y \approx 0$). If y is smaller than the smallest floating point value of the computer, then $y = 0$ and $\log y$ would be ∞, so to handle it, the logistic function is usually combined with cross-entropy loss, which is called **logistic-cross-entropy loss** and is formulated in equation 2.6. Note that $f(\mathbf{w};\mathbf{x})$ is the function representing the prediction function and the goal of optimization is to learn an appropriate prediction function such that the loss $\mathcal{L}$ is minimized.

$$\begin{aligned} z &= f(\mathbf{w};\mathbf{x}) \\ y &= \sigma(z) \\ \mathcal{L}_{\text{CE}} &= -t\log y - (1-t)\log 1 - y \end{aligned} \tag{2.5}$$

$$\mathcal{L}_{\text{LCE}}(z,t) = \mathcal{L}_{\text{CE}}(\sigma(z),t) = t\log\left(1+e^{-z}\right) + (1-t)\log\left(1+e^{z}\right) \tag{2.6}$$

z in the above equation represents the output of the final layer. The $y = \sigma(z)$ above, represents a logistic transformation of the prediction from the final output layer of the NN classifier. Often in binary classification the logistic transformation of the output from the last layer is done before computing the loss. In the case of multi-class classification a *softmax* transformation is done, which we discuss below.

For multi-class classification case- This case is very similar to binary classification case above. By analogy from the binary case, we formulate for multi-class classification. Targets are represented using **one-hot vectors**, or a **one-of-K-encoding**:

$$\mathbf{t} = \underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{entry } k \text{ is } 1} \tag{2.7}$$

For a multi-class model, we have K outputs, with each output representing the prediction for the corresponding class. The goal of optimization is still the same - to learn $f(\mathbf{w}; \mathbf{x})$, an appropriate prediction function such that the loss $\mathcal{L}$ is minimized. The output $\mathbf{z}$ and target $\mathbf{t}$ are K dimensional vectors.

$$\mathbf{z} = f(\mathbf{w}; \mathbf{x}) \tag{2.8}$$

In the binary classification case, we used a logistic transformation of the prediction. For the multi-class classification case we can use a **softmax** function which is a multivariate generalization of the logistic function. The outputs of a softmax function are nonnegative and sum to 1, this property of softmax ouput helps to interpret the outputs as a probability distribution over the K classes. The formulation is:

$$y_k = \operatorname{softmax}(z_1, \ldots, z_K)_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp z_j} \tag{2.9}$$

Cross-entropy loss for multi-class classification is:

$$\begin{aligned} \mathcal{L}_{\text{CE}}(\mathbf{y}, \mathbf{t}) &= -\sum_{k=1}^{K} t_k \log y_k \\ &= -\mathbf{t}^\top (\log \mathbf{y}) \end{aligned} \tag{2.10}$$

$(\log \mathbf{y})$ represents elementwise log, one of t_k's is 1 and rest all are 0, so the summation has the effect of picking the relevant entry of the vector $(\log \mathbf{y})$. Combining the equations 2.8, 2.9 and 2.10, we get the softmax-cross-entropy loss function($\mathcal{L}_{\text{SCE}}$) as in equation 2.12 for multi-class classification case.

$$\begin{aligned} \mathbf{z} &= f(\mathbf{w}; \mathbf{x}) \\ \mathbf{y} &= \operatorname{softmax}(\mathbf{z}) \\ \mathcal{L}_{\text{SCE}} &= -\mathbf{t}^\top (\log \mathbf{y}) \end{aligned} \tag{2.11}$$

$$\mathcal{L}_{SCE}(\mathbf{t}, f(\mathbf{w}; \mathbf{x})) = \sum_{i=1}^{K} -t_i \log \left(\frac{\exp(f_i(\mathbf{w}; \mathbf{x}))}{\sum_{j=1}^{K} \exp(f_j(\mathbf{w}; \mathbf{x}))} \right) \tag{2.12}$$

2.2.3 Training Algorithm

When we combine a classifier model with loss function, we get an optimization problem where we are trying to minimize the loss with respect to the model parameters of the classifier model (i.e. weights and bias). The optimization problem is as defined in 2.13, an expected loss over the data samples in the batch is minimized.

$$\min_{\mathbf{w}\in\mathcal{W}} G(\mathbf{w}) := \mathrm{E}_{(\mathbf{x},\mathbf{t})}[\mathcal{L}_{SCE}(\mathbf{t}, f(\mathbf{w};\mathbf{x}))] \tag{2.13}$$

The most commonly used algorithm for optimizing a loss is Stochastic Gradient Descent (Hardt *et al.* [12], Lan [21]). Below is the pseudo code of Stochastic Gradient Descent (SGD) in Algorithm 1. SGD and algorithms inspired by SGD (Kingma and Ba [19], Duchi *et al.* [6]) are commonly used as go to algorithms for optimizing loss functions.

Note that the gradient of the loss function for a NN is computed using backpropagation. We do not discuss backpropagation in this thesis in detail because it is beyond the scope of this thesis.

Algorithm 1: Stochastic Gradient Descent($\mathcal{L}, \mathbf{X}, \mathbf{t}$)

```
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$;
for $i = 1, \dots$ number of epochs do
    Shuffle data points from $i = 1, \dots, n$;
    for $j = 1, \dots, n$ do
        compute loss$\mathcal{L}_j$ $\mathbf{g} \leftarrow \nabla\mathcal{L}_j(\mathbf{w})$ ;    /* gradient is computed
         using backpropagation */
        $\mathbf{w} \leftarrow \mathbf{w} - \eta_t\mathbf{g}$ ;          /* $\eta_t$ is learning rate of SGD */
    end
end
```

2.3 Generalization and stability

In this section we define the notion of stability. We describe how stability and generalization relate. We will use definitions from this discussion to qualitatively compare different label smoothing methods including no-label smoothing. When we train a machine learning model, we don't just want it to learn to model the training data. We want it to generalize to data it hasn't seen before. Fortunately, there's a very convenient way to measure an algorithm's generalization performance: we measure its performance on a held-out test set, consisting of examples it hasn't seen before. We measure the absolute value of (test error - training error) to get the generalization error. If an algorithm works well on the training set but fails to generalize, we say it is overfitting. There is a connection between generalization and stability of a learning algorithm.

Now, we describe what it means when we talk about **stability** of a learning method. We follow the notion of stability as it is followed in Bousquet and Elisseeff [1] and Hardt *et al.* [12]. **A learning method is stable if a small change of the input to the algorithm does not change the output of the algorithm much**. Below we formalize this:

Given the training set Z^n and an additional example Z_{n+1}, let $Z^n_{i/n+1}$ be the training set obtained by replacing the i-th example of Z^n with Z_{n+1}; that is, $Z^n_{i/n+1} = (Z_1, Z_2, \ldots, Z_{i-1}, Z_{n+1}, Z_{i+1}, \ldots, Z_n)$. In our definition of stability "a small change of input" means that we feed algorithm M with $Z^n_{i/n+1}$ instead of with Z^n, i.e. we only change one training example. We measure the effect of this small change of the input on the output of M, by comparing the loss of the hypothesis $M(Z^n)$ on Z_i to the loss of the hypothesis $M(Z^n_{i/n+1})$ on Z_i. Given a loss function L, a good learning algorithm will have:

$$\left|L\left(M\left(Z^n_{i/n+1}\right), Z_i\right) - L\left(M\left(Z^n\right), Z_i\right)\right| > 0 \tag{2.14}$$

The Equation 2.14 which is derived from the definition of stability is used to derive the **generalization** bound on the learning method for the given data. Since in the first term of Equation 2.14, the learning method does not observe the example Z_i while in the second term Z_i is indeed observed. If the preceeding difference is very large, i.e. the learning algorithm is not stable, we suspect that the learning algorithm might lead to overfitting because the algorithm drastically changes its prediction on Z_i if it observes it in the training set. For the preceeding difference term, different learning algorithms can have various bounds, for the case of non-convex functions such as neural networks, trained with SGD, Hardt *et al.* [12](in their theorem 3.12) have proposed a generalization bound κ as given in theorem 2.3.1 below.

Theorem 2.3.1 *For a B-Lipschitz and β-smooth loss function f for every data point z, suppose that we run SGD for T steps with monotonically non-increasing step sizes α, then SGD has uniform stability with*

$$\kappa \leq \frac{1 + 1/\beta c}{n-1} \left(2cB^2\right)^{\frac{1}{\beta c+1}} T^{\frac{\beta c}{\beta c+1}}$$

The definition of B-Lipschitz is given below. A function $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ is B**-Lipschitz** if for all $x, y \in \mathbb{R}^d$,

$$|\mathcal{F}(x) - \mathcal{F}(y)| \leq B\|x - y\|$$

If $\mathcal{F}$ is differentiable, then $\mathcal{F}$ is B-Lipschitz if and only if $\|\nabla \mathcal{F}(x)\| \leq B$ for all x. For details about the proof of the theorem we suggest a reader to refer to Hardt *et al.* [12]. For our analysis of generalization, we will use the definition of the generalization bound from the Theorem 2.3.1. We consider B and T in our bound definition and ignore other terms. For our use, the generalization bound is proportional to B^2 and T as given in equation 2.15.

$$\kappa \propto B^2 T \tag{2.15}$$

2.4 Problem of Overfitting

Overfitting is a phenomenon that happens when a model learns the details in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model‘s ability to generalize.

Overfitting is more likely with nonparametric models (such as decision trees) and nonlinear models (such as NN) because such models have flexibility when learning a target function. As such, many nonparametric machine learning algorithms also include parameters or techniques to limit and constrain how much detail the model learns. Deep learning models are nonlinear and parametric models but still they might overfit if the training is not stable. By stable training, we imply that the gradients during training should be consistent and not changing drastically due to model parameters. There are a few techniques to reduce overfitting when training NN, such as - increasing size of training data, reducing model complexity, early stopping [37] during the training phase (keep a check on the validation loss over the training period as the loss begins to increase, stop training), regularization techniques - Ridge Regularization, Lasso Regularization, dropout, and label smoothing by Szegedy *et al.* [45]. We discuss the regularization techniques below so that we can explain how label smoothing relates to other regularization techniques.

2.5 Regularization

Regularization is the process of adding information in order to prevent overfitting. The extra information is added using a mathematical term called the regularization term, or penalty. Regularization can be added to any of– the model architecture (dropout), the loss function, the training algorithm (Sekhari *et al.* [40]), and the training data (data augmentation, mixup). The impact of regularization is that it imposes a cost on the optimization function and the net optimization trajectory. A regularization term is useful in preventing overfitting in classification problems. For the scope of this thesis, we focus on regularization for deep learning classifiers.

2.5.1 Regularization for deep learning

In practice, overfitting causes the neural network model to perform very well during training, but the performance gets much worse during inference time when faced with brand new data. To prevent overfitting or a high variance in deep learning, using regularization is recommended. In the section 2.2, we introduced the components of training an ANN based classifier - a model, a loss function, training algorithm and data, there are regularization techniques for each of these components of training a classifier. Depending on which component a user would like to regularize, there are various methods of regularization in deep learning, like - $l2$, $l1$ regularization, dropout and label smoothing.

2.5.2 L2 Regularization

$L2$ regularization is the most common type of all regularizations. In practice, it is also known as weight decay, Tikhonov regularization, or ridge regression. $L2$ regularization is implemented by altering the loss function of the NN directly. For $L2$ regularization the loss function of the NN is extended by an additional regularization term, which is called Ω_{l2}. The regularization term Ω_{l2} is defined as the Euclidean Norm (or $L2$ norm) of the weight matrices, which is the sum over all squared weight(ω) values of a weight matrix. The benefit of using $L2$ regularization is that it encourages the weight values to approach zero.

$$\Omega_{l2}(W) = \|W\|_2^2 = \sum_i \sum_j w_{ij}^2 \tag{2.16}$$

2.5.3 L1 Regularisation

$L1$ regularisation is also known as Lasso regression. For $L1$ regularization the loss function of the NN is extended by an additional regularization term, which is called Ω_{l1}. The regularization term is the sum of the absolute values of the weight parameters in a weight matrix. Performing $L1$ regularization encourages the weight values to be zero, so it helps in feature selection by making useless weights zero and assigning weights only to useful features.

$$\Omega_{l1}(W) = \|W\|_1 = \sum_i \sum_j |w_{ij}| \tag{2.17}$$

2.5.4 Dropout

Dropout [42], [8] is a regularization technique that is applied on the model architecture of the NN. The procedure behind dropout regularization is quite simple - during training, each neuron can be turned off with some probability p. The deactivation of neurons with a certain probability is applied at each forward propagation and weight update step.

2.5.5 Label smoothing

Label smoothing [45] is a regularization technique that is applied on the loss function of the NN, when training with a CE. It is different from $L1$ and $L2$ regularization as a fact that it is applied based on the target labels of the training dataset. Whereas, $L1$ and $L2$ are applied based on the weights of the NN. Since label smoothing is a major part of the thesis, so we have a separate Chapter 3 for label smoothing.

Chapter 3

Label Smoothing

In this chapter we introduce label smoothing formally and mathematically. We then introduce our proposed label smoothing called PLS in Section 3.3. Further, we discuss prior research in label smoothing. Finally, we revisit the contributions of this thesis from Section 1.3 and discuss the specifics here.

3.1 Introduction to Label Smoothing

Szegedy *et al.* [45] introduced label smoothing to improve accuracy the Inception architecture of GoogLeNet [44] on the ImageNet Dataset ([38], [4]) . Szegedy *et al.* [45] used CE (Cross Entropy Loss) as a loss function to better train a classifier using label smoothing. Choosing one-hot vectors as the probability mass function of target output vector while training a classifier implies that all the concentration is put on one true class label. The idea behind label smoothing is to modify the one-hot vector of target outputs and use a smaller concentration on the true class label and distribute the remaining concentration to all the class labels uniformly.

We illustrate with an example, consider the case when the task at hand is to train an emotion classifier[1]. The training dataset has sentences as inputs and emotions (anger, joy, fear, disgust, excitement) as labels. Consider that in the training dataset, one of the data samples is - "OMG, yep!! That is the answer! Thanks a lot", which is labeled as 'excitement' in the dataset. A one-hot target vector for this training sample would look as in Figure 3.1. A

[1]Check the Chapter 5 on text classification

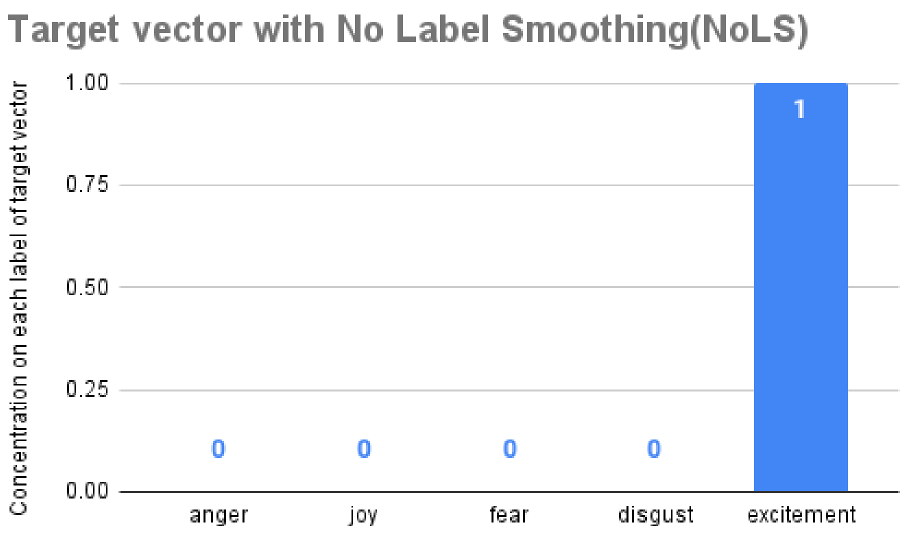


Figure 3.1: An example of the concentration distribution over a target vector with NoLS. The ground truth label in the target vector is ‘excitement’.

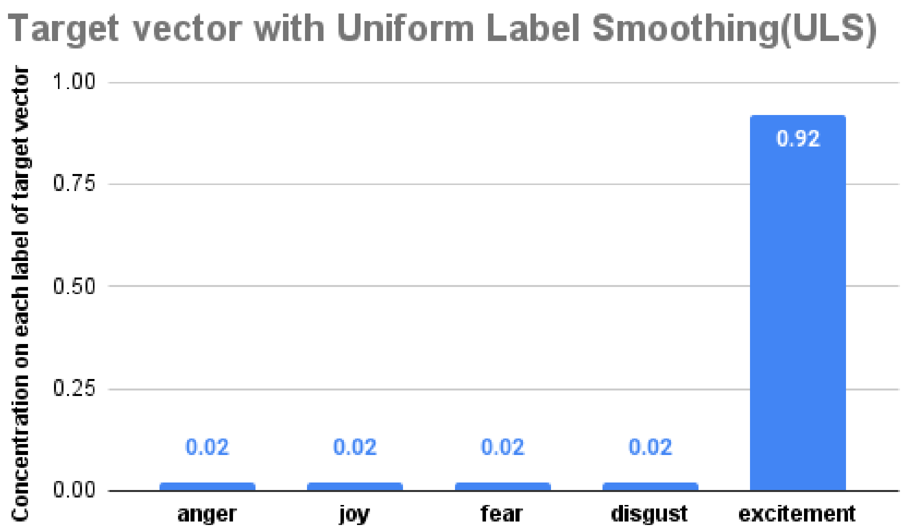


Figure 3.2: An example of the concentration distribution over a target vector with ULS. The ground truth label in the target vector is ‘excitement’, so it is assigned 0.9 concentration, and then 0.1 concentration is distributed among all the labels uniformly, so each label gets $0.1/5 = 0.02$ and ‘excitement’ gets $0.9 + 0.02 = 0.92$ concentration.

target vector for the same training sample with label smoothing as proposed by Szegedy *et al.* [45] is shown in Figure 3.2. Observe that 0.9 concentration is given to 'excitement' - the ground truth and 0.1/5 = 0.02 is distributed among all the 5 labels uniformly. The quantity that is distributed uniformly among all the labels is called the **SR (Smoothing Ratio)**, which is 0.1 in this example.

In this manuscript, we refer to the label smoothing method proposed by Szegedy *et al.* [45] as **ULS (Uniform Label Smoothing)** because the SR is distributed among all the labels uniformly. We refer to the one-hot method as **NoLS No Label Smoothing**. We describe ULS formally in Section 3.2.

3.2 Formulation for Label Smoothing

In this section we use the same notation that we used in Section 2.2.2. We described CE and softmax in the Section 2.2.2. We extend from Section 2.2.2 the formulation for ULS as presented by Szegedy *et al.* [45].

For each training example in set $\left\{\left(\mathbf{x}^{(i)}, t^{(i)}\right)\right\}_{i=1}^{N}$, a classifier model computes the probability for each label $k \in \{1 \dots K\} : p(k \mid \mathbf{x}^{(i)})$. From equation 2.9, we get equation 3.1.

$$\begin{aligned} z &= f(w; x) \\ y_k = \operatorname{softmax}(z_1, \dots, z_K)_k &= \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} \\ p(k \mid \mathbf{x}^{(i)}) &= y_k \\ p(k \mid \mathbf{x}^{(i)}) = \operatorname{softmax}(z_1, \dots, z_K)_k &= \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} \end{aligned} \tag{3.1}$$

Here, z_j are the logits or unnormalized log probabilities. Let the concentration distribution over the labels of the ground truth target vector be $q(k \mid \mathbf{x}^{(i)})$ and normalized such that $\sum_k q(k \mid \mathbf{x}^{(i)}) = 1$ for the i-th training example. The CE loss is as in 3.2.

$$\mathcal{L}_{\mathrm{CE}}(\mathbf{p}, \mathbf{q}) = -\sum_{k=1}^{K} q(k \mid \mathbf{x}^{(i)}) \log p(k \mid \mathbf{x}^{(i)}) \tag{3.2}$$

Consider the case of NoLS where ground-truth vectors are one-hot encoded, so $q(\mathbf{t} \mid \mathbf{x}^{(i)}) = 1$ and $q(k \mid \mathbf{x}^{(i)}) = 0$ for all $k \neq \mathbf{t}$. For this case, minimizing the cross entropy is equivalent to maximizing the log-likelihood of the one single correct label. For a particular example $(\mathbf{x}^{(i)})$ with label t, the log-likelihood is maximized for $q(k \mid \mathbf{x}^{(i)}) = \delta_{k,t}$, where $\delta_{k,t}$ which equals 1 for $k = t$ and 0 otherwise as shown in 3.1. Achieving this maximum is not possible for finite prediction from logits z_k. It is only approachable if $z_t \gg z_k$ for all $k \neq t$, i.e., if the logit corresponding to the ground truth label is much greater than all other logits. Training and optimizing with such a goal is possible but it causes two problems identified in 3.2.1

3.2.1 Advantages of ULS

ULS helps in alleviating the following two problems that come up when training with NoLS.

- Overfitting: if the classifier model learns to allocate full concentration to the ground-truth label for each training data sample, then it is expected to not generalize well. By using ULS, some concentration is given to non ground truth labels too, due to this the model has a chance of learning other predictions and hence the chance overfitting is limited.

- Over confident model: The difference between the largest logit and all other logits becomes very large but the gradient of CE with respect to each logit is bounded, so it makes it difficult for the model to adapt to newer data since the model becomes over confident about its predictions. By using ULS, some concentration is given to non ground truth labels as a result of which non ground truth logit can also attain a finite value, so the logit for ground truth can also attain a finite value.

3.2.2 Formulation for ULS

Let a uniform distribution over labels be $u(k)$. $u(k)$ is independent of the training data sample $\mathbf{x}^{(i)}$. Let ϵ be the SR. For a training data sample with the ground-truth label t, concentration distribution over the ground truth vector with NoLS is $q(k \mid \mathbf{x}^{(i)}) = \delta_{k,t}$. Concentration distribution over the ground truth vector with ULS is noted $q^{ULS}(k \mid \mathbf{x}^{(i)})$ (see Equation 3.3) and is a mixture of the original ground-truth distribution $q(k \mid \mathbf{x}^{(i)})$ and a fixed distribution $u(k)$ with ratios $1-\epsilon$ and ϵ, respectively. ϵ is SR, which decides the level of smoothing and falls between 0 and 1, usually it is a small value of the order < 0.2 . Szegedy *et al.* [45] proposed to use $u(k)$ as a uniform distribution on all the labels, so $u(k) = 1/K$, which gives us equation 3.4

$$q^{ULS}(k \mid \mathbf{x}^{(i)}) = (1-\epsilon)\delta_{k,t} + \epsilon u(k) \tag{3.3}$$

$$q^{ULS}(k \mid \mathbf{x}^{(i)}) = (1-\epsilon)\delta_{k,t} + \frac{\epsilon}{K} \tag{3.4}$$

We refer to any change in concentration distribution over the ground truth label vector as **Label Smoothing Regularization (LSR)**. From here onwards we write $q(k \mid \mathbf{x}^{(i)}) = \delta_{k,t}$ as $q^{NoLS}(k \mid \mathbf{x}^{(i)}) = \delta_{k,t}$. Computing CE for $q^{ULS}(k \mid \mathbf{x}^{(i)})$ with the predictions $p(k \mid \mathbf{x}^{(i)})$ gives:

$$\begin{aligned} \mathcal{L}_{CE}(p(k \mid \mathbf{x}^{(i)}), q^{ULS}(k \mid \mathbf{x}^{(i)})) &= -\sum_{k=1}^{K} \log p(k \mid \mathbf{x}^{(i)}) q^{ULS}(k \mid \mathbf{x}^{(i)}) \\ &= (1-\epsilon)\mathcal{L}_{CE}(q^{NoLS}(k \mid \mathbf{x}^{(i)}), p(k \mid \mathbf{x}^{(i)})) \\ &\quad +\epsilon\mathcal{L}_{CE}(u(k), p(k \mid \mathbf{x}^{(i)})) \end{aligned} \tag{3.5}$$

From Equation 3.5 it is clear that LSR is similar to replacing a single CE $\mathcal{L}_{CE}(q^{NoLS}, p)$ with a pair of losses $\mathcal{L}_{CE}(q^{NoLS}, p)$ and $\mathcal{L}_{CE}(u, p)$. The second loss punishes deviation of prediction distribution p from the prior $u(k)$, with relative ratio $\frac{\epsilon}{1-\epsilon}$. In algorithm 2 we show the SGD update for training with ULS. The target vector takes into account the LSR change and we denote them by $\mathbf{t}_{LSR}^{(j)} = q^{ULS}(k \mid \mathbf{x}^{(j)})$

Algorithm 2: Stochastic Gradient Descent with ULS $(\mathcal{L}, \mathbf{X}, \mathbf{t})$

```
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$;
for $i = 1, \ldots$ number of epochs do
    Shuffle data points from $i = 1, \ldots, n$;
    for $j = 1, \ldots, n$ do
        sample $(\mathbf{x}^{(j)}, \mathbf{t}^{(j)})$;
        set $\mathbf{t}_{LSR}^{(j)} = q^{ULS}(k, \mathbf{t}^{(j)} \mid \mathbf{x}^{(j)})$;
        $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} \mathcal{L}_j(\mathbf{t}_{LSR}^{(j)}, f(\mathbf{w}; \mathbf{x}^{(j)}))$ ;      /* gradient is computed
         using backpropagation */
        $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \mathbf{g}$ ;     /* $\eta_t$ is the learning rate of SGD */
    end
end
```

3.3 Preferential Label Smoothing

Consider the example in the Figure 3.2, the ground truth label for the input sentence is ‘excitement’, the goal of label smoothing is to distribute the SR among all the labels. If we observe the labels, we can see that if a sentence has a label of ‘excitement’ then it’s more likely that the sentence might have more similarity with ‘joy’ than ‘anger’ or ‘fear’ or ‘disgust’. While distributing SR among all the labels we should make sure that we give more concentration to ‘joy’ than we give to ‘fear’ or ‘anger’ or ‘disgust’. By considering ‘joy’ as equal to ‘fear’ or ‘anger’ or ‘disgust’ we are basically forcing the model to treat all non ground truth labels equally, which is intuitively an incorrect thing to do. Imagine if some emotion psychologist gives us qualitatively rich information on the relationships between different emotion labels and we use that information to model a PLS scheme, we might be able to come up with a more reasonable concentration distribution such as the one shown in Figure 3.3.

Based on this intuition we propose **PLS - Preferrential Label Smoothing**. The idea is to distribute the SR on non ground truth labels based on the relationship of the non ground truth labels with the ground truth labels, this relationship can be learned from some external data or provided by an expert in the area.

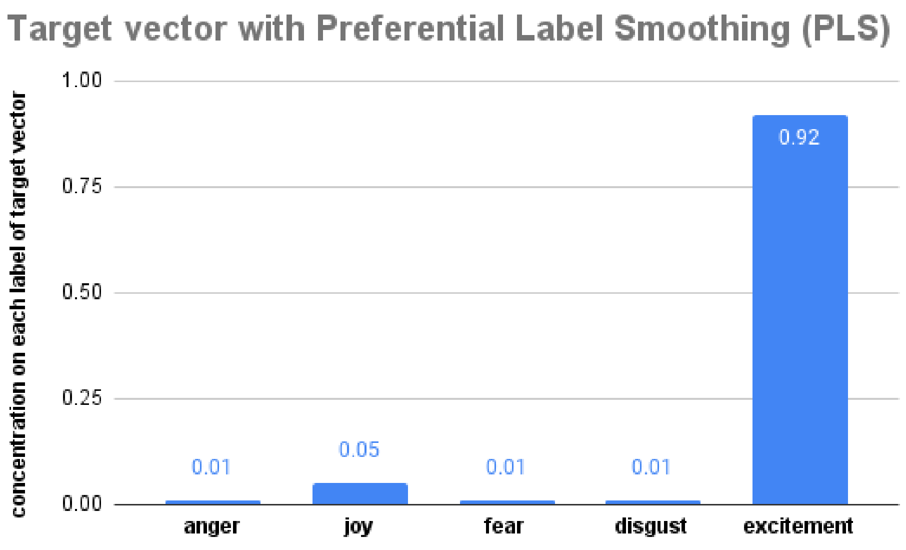
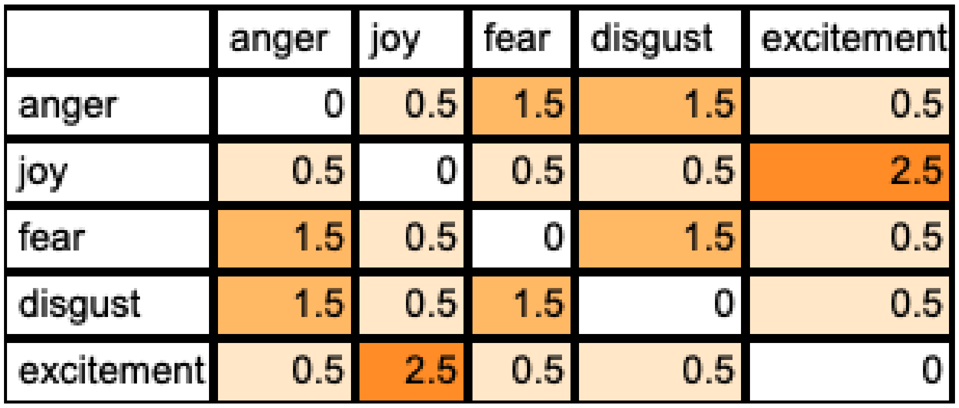


Figure 3.3: An example of the concentration distribution over a target vector with PLS. The ground truth label in the target vector is ‘excitement’, so it is assigned 0.9 concentration, and then $\epsilon = 0.1$ concentration is distributed among all the labels. The emotion ‘excitement’ gets $0.1/5 = 0.02$, ‘joy’ gets 0.05; ‘anger’, ‘disgust’ and ‘fear’ get 0.01 concentration.

3.3.1 Formulation for PLS

Analogous to Equation 3.3, we can define the concentration distribution over a ground truth label vector when using PLS as in Equation 3.6. Let Δ be an oracle function represented as a matrix, which contains the information of relationship between the ground truth label and the non ground truth labels. Δ can be constructed based on information given by a subject matter expert or from some external data based on relationships between the labels. Such a Δ matrix for emotion classification is shown in Figure 3.4, for example. θ is another oracle function derived from Δ by normalizing each row of the Δ matrix, the corresponding θ matrix for emotion classification is shown in Figure 3.5.

$$q^{PLS}(k \mid \mathbf{x}^{(i)}) = (1-\epsilon)\delta_{k,t} + \epsilon\theta(k) \tag{3.6}$$



	anger	joy	fear	disgust	excitement
anger	0	0.5	1.5	1.5	0.5
joy	0.5	0	0.5	0.5	2.5
fear	1.5	0.5	0	1.5	0.5
disgust	1.5	0.5	1.5	0	0.5
excitement	0.5	2.5	0.5	0.5	0

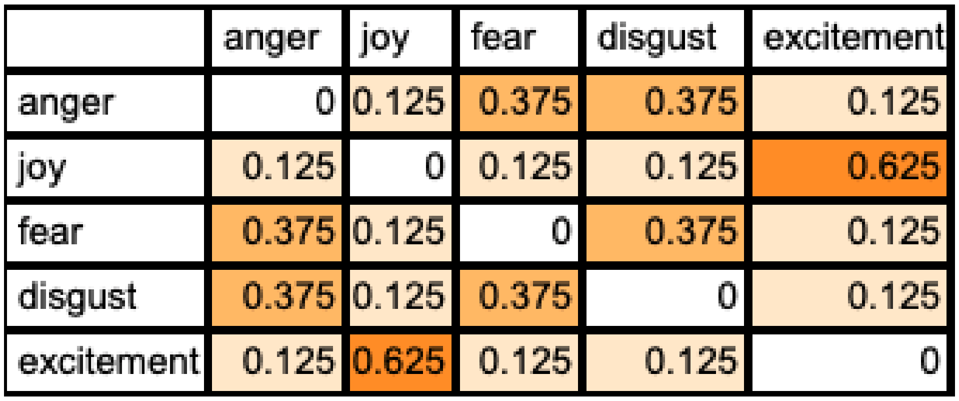
Figure 3.4: An example of a relationship matrix Δ for the emotion classification task. $\Delta(k)$ gives the relationship of all labels with the k-th label. Δ is a symmetric matrix. We chose to keep the relationship between the same emotion pairs (anger, anger), (joy, joy), (fear, fear),(disgust, disgust),(excitement, excitement) as 0 for the $\Delta(k)$ matrix to imply that whole SR is distributed among non ground truth labels.

Computing CE for $q^{PLS}(k \mid \mathbf{x}^{(i)})$ with the predictions $p(k \mid \mathbf{x}^{(i)})$ gives:

$$\begin{aligned} \mathcal{L}_{CE}(p(k \mid \mathbf{x}^{(i)}), q^{PLS}(k \mid \mathbf{x}^{(i)})) &= -\sum_{k=1}^{K} \log p(k \mid \mathbf{x}^{(i)}) q^{PLS}(k \mid \mathbf{x}^{(i)}) \\ &= (1-\epsilon)\mathcal{L}_{CE}(q^{NoLS}(k \mid \mathbf{x}^{(i)}), p(k \mid \mathbf{x}^{(i)})) \\ &\quad + \epsilon \mathcal{L}_{CE}(\theta(k), p(k \mid \mathbf{x}^{(i)})) \end{aligned} \tag{3.7}$$

Equation 3.7 again suggests that LSR in this case is similar to replacing a single CE $\mathcal{L}_{CE}(q^{NoLS}, p)$ with a pair of losses $\mathcal{L}_{CE}(q^{NoLS}, p)$ and $\mathcal{L}_{CE}(\theta, p)$. The second loss punishes deviation of prediction distribution p from the prior relationship function $\theta(k)$, with relative ratio $\frac{\epsilon}{1-\epsilon}$. In algorithm 3 we show the SGD update for training with PLS. The target vectors taking in to account LSR now change and we denote them by $\mathbf{t}_{LSR}^{(j)} = q^{PLS}(k \mid \mathbf{x}^{(j)})$.

We present θ for example in Figure 3.3 as a matrix in Figure 3.4. From the matrix, $\theta(excitement) = [0.5, 2.5, 0.5, 0.5, 1.0]$.



	anger	joy	fear	disgust	excitement
anger	0	0.125	0.375	0.375	0.125
joy	0.125	0	0.125	0.125	0.625
fear	0.375	0.125	0	0.375	0.125
disgust	0.375	0.125	0.375	0	0.125
excitement	0.125	0.625	0.125	0.125	0

Figure 3.5: An example of a normalised relationship matrix θ for the emotion classification task. Each row in the Δ matrix was normalized to obtain the θ matrix and this can make θ asymmetric since the sum for each row in θ matrix can be different, but in this example it is symmetric. Diagonals are kept 0 to ensure that all the SR is distributed to the non ground truth labels.

Algorithm 3: Stochastic Gradient Descent with PLS $(\mathcal{L}, \mathbf{X}, \mathbf{t})$

```
w ← random vector in $\mathbb{R}^d$;
for $i = 1, \ldots$ *number of epochs* do
    Shuffle data points from $i = 1, \ldots, n$;
    for $j = 1, \ldots, n$ do
        sample $(\mathbf{x}^{(j)}, \mathbf{t}^{(j)})$;
        set $\mathbf{t}^{(j)}_{LSR} = q^{PLS}(k, \mathbf{t}^{(j)} \mid \mathbf{x}^{(j)})$;
        $\mathbf{g} \leftarrow \nabla_{\mathbf{w}} \mathcal{L}_j(\mathbf{t}^{(j)}_{LSR}, f(\mathbf{w}; \mathbf{x}^{(j)}))$ ;      /* gradient is computed
         using backpropagation */
        $\mathbf{w} \leftarrow \mathbf{w} - \eta_t \mathbf{g}$ ;     /* $\eta_t$ is the learning rate of SGD */
    end
end
```

3.4 Related work

Since label smoothing helps in improving generalization of deep learning models, it has become a common practice. Many architectures use label smoothing for various tasks like classification, speech recognition and machine translation. For machine translation, highly cited architectures include the SEQ2SEQ by Sutskever *et al.* [43] and the transformer by Vaswani *et al.* [46]. Label smoothing was used with a transformer [46] which helped improve its BLEU score, all of which supports usage of label smoothing.

Research in label smoothing is in its relatively initial stages. Work by Szegedy *et al.* [45] introduced LSR in 2015 as a technique to improve image classification with the inception architecture. There were many other techniques in their paper which helped improve image classification with inception, so LSR did not get the attention from the research community as a research topic. Although many models and tasks in practice kept using LSR for different purposes, the research on LSR itself was not there. Müller *et al.* [33] in 2019 brought the attention of research community to study LSR as a research topic. Müller *et al.* [33] discuss the effects of label smoothing in the domains of image classification and machine translation by studying how the representations differs between the penultimate layer of the networks with and without label smoothing. The visualization technique to visualize penultimate layers of the output makes it clear that label smoothing encouraged representations in the penultimate layer to group labels in tight equally distant clusters. Müller *et al.* [33] also show that label smoothing helps in improving model calibration for both machine translation and image classification tasks, label smoothing shows effects similar to temperature scaling by Guo *et al.* [11]. Additionally, Müller *et al.* [33] show that LSR weakens knowledge distillation in students. Although LSR improves the accuracy of the teacher network, teachers trained with label smoothing produce less useful student networks compared to teachers trained with hard targets. Müller *et al.* [33] show that less information passes from teachers (trained with label smoothing) to students by comparing mutual information between input and output of the two models. Müller *et al.* [33] spurred and encouraged more work and discussions around label smoothing.

Prior related approaches other than works by Szegedy *et al.* [45] are instance based approaches of label smoothing (Zhang *et al.* [49], Maher and Kull [28], Zhang and Sabuncu [50]). In these approaches, even for the data points belonging to the same class label, the way label concentration is distributed changes depending on the data point. All of these previous work can be unified under the term PLS (Preferential Label Smoothing).

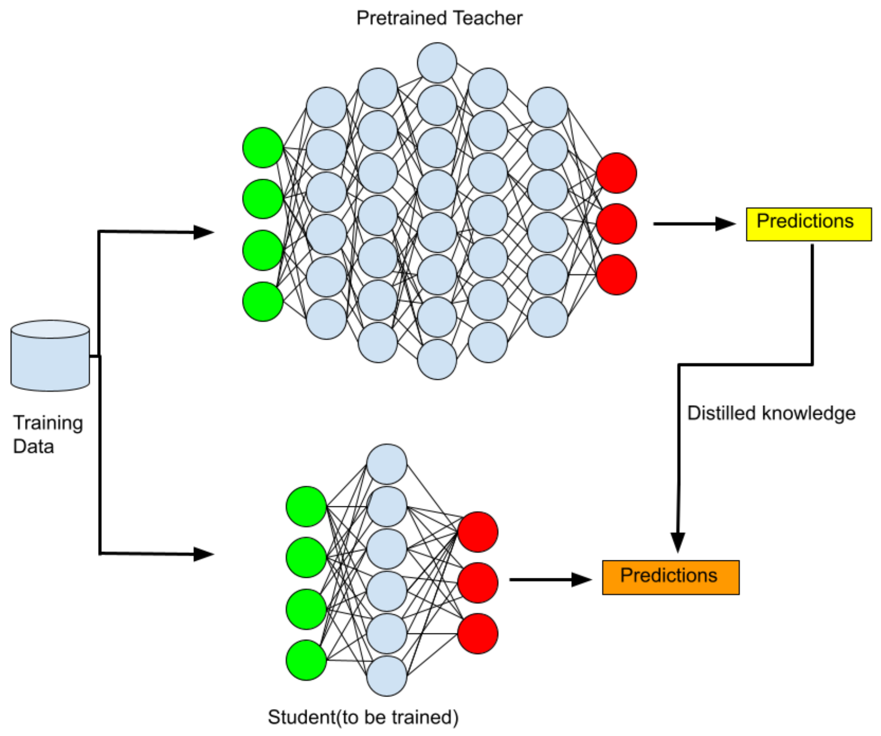


Figure 3.6: Example of a teacher distilling knowledge to train a student model.

3.4.1 Label smoothing and Knowledge Distillation

In this section we explain Knowledge Distillation (KD) and then we explain how it is related to label smoothing.

Knowledge Distillation - KD by Hinton *et al.* [14] is the idea of transferring knowledge from a large and complex model with high number of parameters (called a *teacher*) to a smaller and simpler model with fewer parameters (called a *student*). By transferring knowledge from teacher to student, KD helps in compressing a larger model into a smaller model. The predictions of the teacher model are used to train the student model. The steps to KD are (*i*) train the teacher model with the entire data, which may be computationally expensive, (*ii*) forward pass through the teacher network to get outputs, (*iii*) use the outputs of the teacher to compute loss for the student, the schematic in Figure 3.6 shows this for more clarity. Müller *et al.* [33] show that a teacher trained with ULS trains poor quality of student networks.

Yuan *et al.* [47] draw a parallel between label smoothing and knowledge distillation by claiming that label smoothing regularisation is a special case of knowledge distillation which outputs a uniform distribution. Also, Yuan *et al.* [47] suggest that the knowledge from knowledge distillation of a teacher does not just include similarity between categories but it also imposes regularization on the student training. Yuan *et al.* [47] perform two kinds of experiments to understand the behavior of knowledge distillation and then use mathematical formulations for label smoothing and knowledge distillation to draw a line between both of them. The experiments are - Reveresed KD (use a student model to train a teacher model), Defective KD (use a poorly trained teacher model with very low accuracy to train a student model). Contradictory to the expectation, the experimental results show that students can improve teachers significantly and poorly trained teachers can improve student models. These experimental results are the basis for the conclusion that label smoothing regularization is a special case of knowledge distillation. There's more work by Zhang *et al.* [49] which suggests an online label smoothing technique in which the output distribution of the label is changed after each epoch based on the output of a trained teacher model. The Zhang *et al.* [49] approach is memory-wise and computation wise more intensive than the versions of PLS that we propose in Chapter 4 and Chapter 5.

3.5 Components of Our Study

We study effects of NoLS, ULS, and PLS on classification problem for both image and text data. To the best of our knowledge classification with text data was not studied in the past so we do it in our study and bridge the research gap. We introduce our study here and give details in the following chapters on image Classification (Chapter 4) and text Classification (5).

3.5.1 Label Smoothing and Gradients

In Section 3.2.1 we discuss how a model is too over confident when training with NoLS, because the largest logit becomes too large than all other logits. This means theoretically, while training with NoLS the model should go through a trajectory of higher gradients so that finally the logit attains a higher value. Using LSR stops the model from becoming too over confident, which means theoretically the model should go through a trajectory of smaller gradients.

In Section 2.3, we discussed a relationship between generalization and the length of gradients while training a NN with SGD. The relationship is given by Hardt *et al.* [12], according to whom, for a NN trained with SGD, the generalization error is bounded by the square of the gradients and the time taken to train, as we mentioned before in Theorem 2.3.1 and Equation 2.15.

Since NoLS does not generalize well, the generalization bound should be higher for NoLS as compared to ULS and PLS. The training trajectory should go through the region of higher gradients so that the generalization bound attains a higher value. This motivates us to run a gradient based analysis of all LSR schemes against NoLS to find out whether this intuition is correct empirically.

3.5.2 Generalization and Learning Rate

The effects of LSR include benefits to the generalization error and improved accuracy. The generalization effects of LSR are not studied with different learning rates. This is important to study to check whether the generalization effects are as prominent at smaller rates as they are at larger rates. Studying this will help us answer whether we can learn faster with NoLS, ULS or PLS.

3.5.3 Effect of label smoothing on increasing and decreasing number of classes in a dataset

We propose PLS because we expect that using PLS would help improve classification performance since we can distribute the concentration based on relationships between the labels. If there were only two labels, then using PLS is not useful because there is one ground truth label and one non-ground truth label. As we increase the number of classes in the dataset, there are more non-ground truth labels and the labels far away from the ground truth get less concentration, while the labels close to the ground truth label get more concentration. As we increase the number of labels in the dataset further, the labels far from the ground truth labels get even less concentration. Studying how this phenomenon affects the model performance would be interesting. It is interesting because the current ULS approach by Müller *et al.* [33] (i.e. allocating a concentration to all the classes uniformly) shows that as the number of classes increase, the effect of label smoothing on model performance keeps decreasing to the point that with a very high number of labels there is no effect (the case of Imagenet), whereas the PLS would be more interesting because it allocates more concentration to the related labels and less concentration to unrelated labels.

Chapter 4

Label Smoothing for Image Classification

In this chapter we give a brief of overview of an empirical study, we then describe the datasets and the neural models that we use for image classification. We describe our approaches of PLS for image classification in Section 4.3. We discuss models used for image classification in Section 4.4. Experimental details and results are presented in Section 4.5. We reflect back on the results in Section 4.6.

4.1 Introduction

Image classification is one of the fundamental and well studied task in supervised learning using NN. Since the advent of Convolutional Neural Network (CNN) for the Imagenet challenge [4], [38], NN have propelled the growth of technology in many fields, some of these fields are automobile industry, ecommerce, healthcare and wildlife monitoring. Image classification can be studied under the use case of single-label and multi-class classification or multi-label classification as described in Section 2.1.

For our experiments we study image classification for the single-label and multi-class classification problem. In our experiments, the goal is to study the effects of label smoothing on model performance and training dynamics.

4.2 Datasets

We describe the datasets that we use in our experiments because the labels in the dataset are useful for explaining our approach of PLS in the following section. We use CIFAR-10 and CIFAR-100 image datasets [20], which are standard datasets for image classification.

CIFAR-10

CIFAR-10 has $60,000$ images and 10 classes. CIFAR-10 is a balanced dataset and each class has $6,000$ images. It has $50,000$ training images and $10,000$ test images. So, there are 5000 training images and 1000 test images per class. Classes in CIFAR-10 are mutually exclusive of each other, so there is no overlap among any of the classes. The class labels in this dataset are the following:

Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck

CIFAR-100

CIFAR-100 has $60,000$ images and 100 classes. CIFAR-100 is also a balanced dataset and each class has 600 images. It has $50,000$ training images and $10,000$ test images, similar to CIFAR-10 but there are 500 training images and 100 test images per class. Classes in CIFAR-100 have groupings. The 100 classes in CIFAR-100 are grouped into 20 superclasses. Each image in the dataset has two labels - (*i*) a "fine" label for the class it belongs to, (*ii*) a "coarse" label for the superclass it belongs to. Since there are 100 classes in CIFAR-100 we present the class labels (along with the hierarchy of superclasses) in the Appendix A.1. For our experiments, we use all the 100 fine labels.

4.3 Approaches of PLS for image classification

In this section, we describe two approaches for PLS.

Cluster Label Smoothing - CLS

Cluster Label Smoothing (CLS) is a special case of PLS where the preference is approximated by which group of superclass a label belongs to. For instance, for training CIFAR 100 dataset (Appendix A.1), if the ground truth label is 'clock' then we distribute SR among all the non ground truth labels uniformly that come under superclass 'household electrical devices' (i.e., 'clock', 'computer keyboard', 'lamp', 'telephone', 'television') and assign no concentration to the rest of the classes (which come under other superclasses). This type of label smoothing represents a label smoothing suggested by the data expert, i.e., the curators of the CIFAR-100 dataset.

Semantic Label Smoothing - SEMLS

Semantic Label Smoothing is a special case of PLS where the preference is approximated by semantic similarity among the labels. We use the word vectors from GloVe embeddings[1] by Pennington *et al.* [35] to get the vector representations of the labels in the CIFAR-10 and CIFAR-100 datasets. Figure 4.1 shows the representation of the labels of the CIFAR-10 dataset projected in 2D. Using the vector representation, we compute the Euclidean distance between the labels. The distances between the labels represent how far away they are from each other. The inverse of the distances between the labels represents the similarity between the labels. Figure 4.2 depicts the pair wise similarity among the labels of the CIFAR-10 dataset. This is the relationship matrix Δ (as introduced in 3.3.1) among the labels of the CIFAR-10 dataset. Note that Δ is a symmetric matrix. From the matrix in the Figure 4.2 we obtain the θ matrix (introduced in Section 3.3.1) for CIFAR-10 dataset as shown in Figure 4.3. Each cell in the θ matrix contains the fraction of the SR concentration that should be assigned to the column label if the row is a ground truth label. For obtaining the θ matrix each row of the Δ matrix is normalized, i.e. each cell in a row (except the diagonal cell) is divided by the sum of all the cells in the row. Normalization is necessary because the sum of each row should be

[1]For more details on word embeddings, refer to Section 5.2

1 for maintaining consistency with Equation 3.6. This θ matrix implies that we distribute SR among all the non-ground truth labels depending on their relationship with the ground truth label.

There is a work by Lukasik *et al.* [27] entitled ‘Semantic label smoothing for sequence to sequence problems’. In their work, they are using semantic similarity between target sequences to choose top-k sequences for training, and then distribute label concentration uniformly among the chosen target sequences. In our work, we are using semantic similarity between target labels to distribute label concentration preferentially (based on how much related ground truth label and non-ground truth label are). We are focusing on text classification task and not on machine translation, unlike Lukasik *et al.* [27], which is focusing on the machine translation.

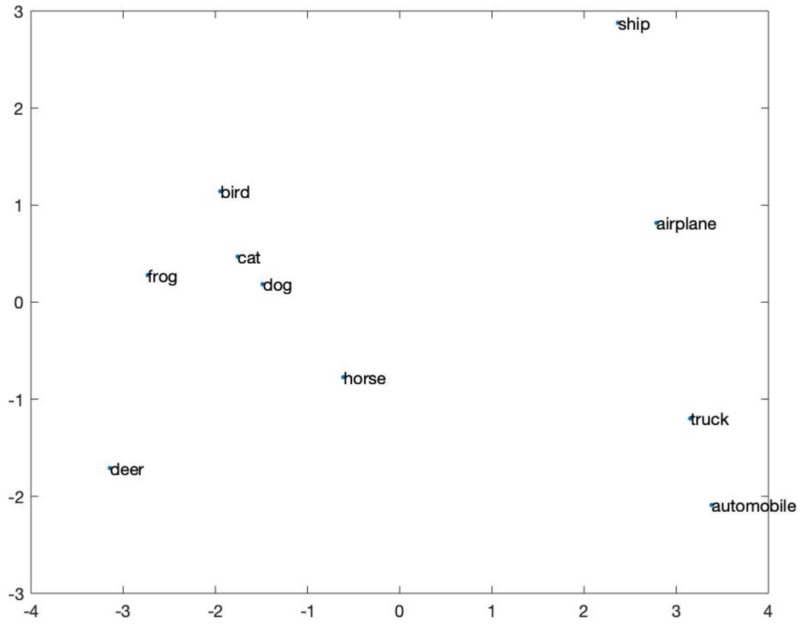


Figure 4.1: Representation of the labels in the CIFAR-10 dataset using GloVe embedding. We obtain 50 dimensional representation via GloVe embeddings, then we sample down the representations to a 2 dimension space using Principal Component Analysis (PCA) to plot them. The closer labels have less distance between one another and are more semantically similar. For instance, ‘cat’ and ‘dog’ are closer to each other so they are semantically more similar than ‘cat’ and ‘ship’ which are farther from each other. This is used to compute semantic similarity.

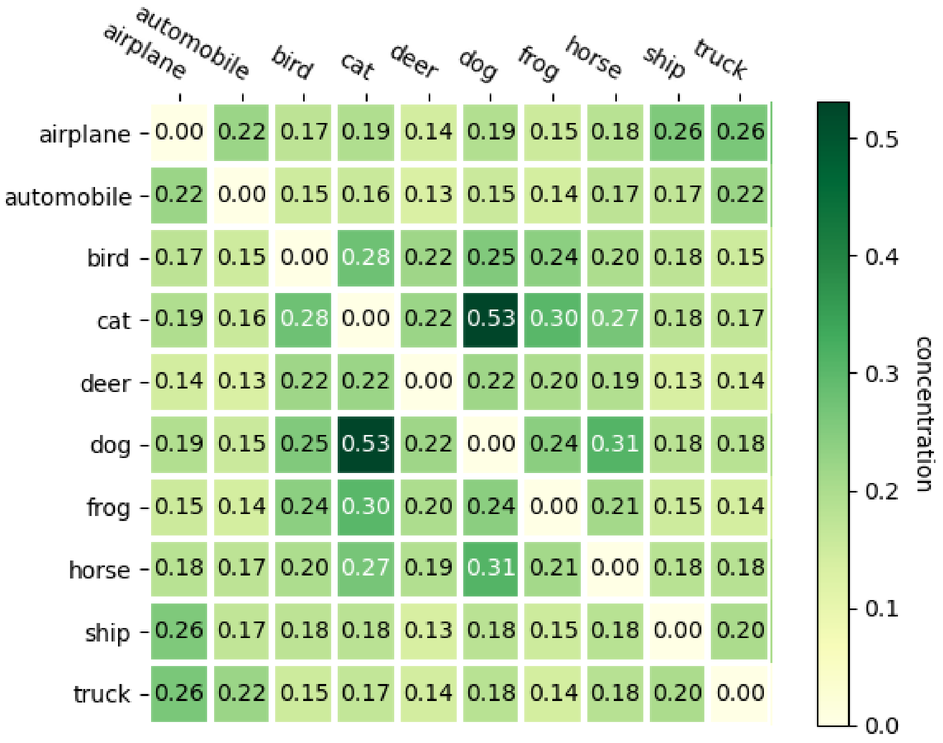


Figure 4.2: **Semantic similarity among the labels in the CIFAR-10 dataset, Δ matrix.** Distances were obtained for each of the label pairs. The inverse of the distance represents the similarity between the label pairs. The inverse of distances for each pair is shown in the Figure. Semantically similar labels such as ‘cat’ and ‘dog’ are closer to each other so their similarity score is higher than label pairs that are semantically less similar such as ‘cat’ and ‘ship’. This is consistent with the observation in Figure 4.1. The diagonals are marked as 0 because the distances for them are 0, the inverse of which would not be possible to represent on the heatmap, so we mark them as 0. This Δ matrix is symmetric.

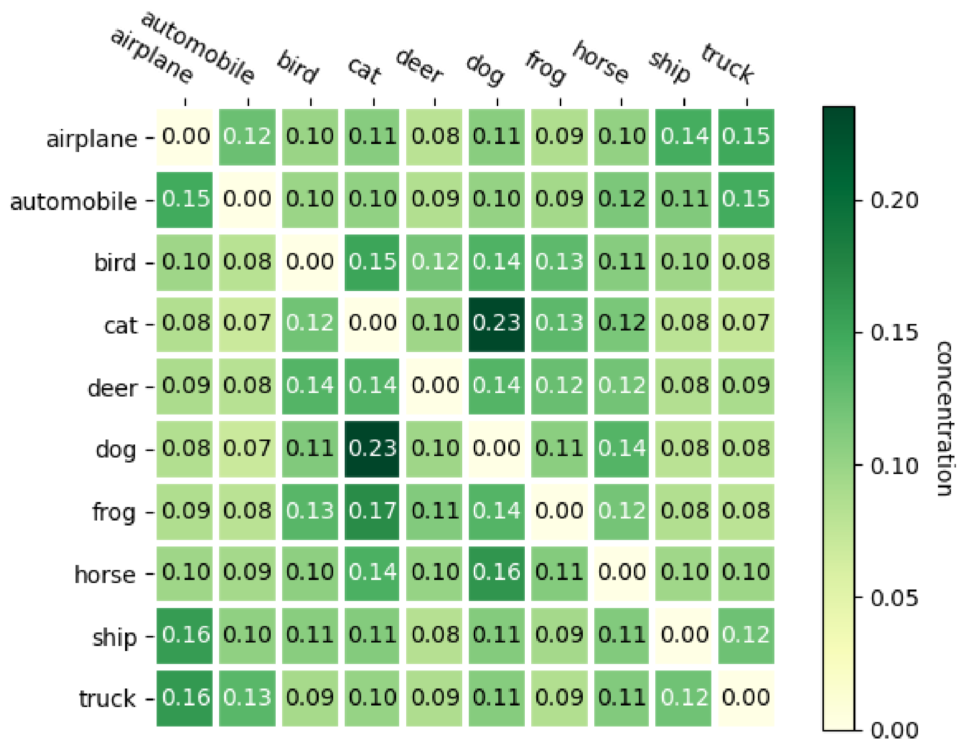


Figure 4.3: θ **matrix for SEMLS.** Each row in the Δ matrix was normalized to obtain the θ matrix which makes θ asymmetric because the sum for each row in the Δ matrix is different. Diagonals are at 0 to imply that all the SR is distributed to the non ground truth labels.

4.4 Neural Models for Image Classification

We use Convolutional Neural Networks (CNN) based neural models for our image classification tasks. We describe a CNN model and then we describe the variant that we use for our experiments - ResNet [13]. We mentioned CNN and ResNet models before in Section 2.2.1. We use two versions of ResNet - ResNet18 and ResNet34. The details of the models are as follows:

Convolutional Neural Networks

As we have mentioned before, NN have a connectivity pattern similar to neurons in the human brain. The CNN architecture is inspired by the Visual Cortex. In our brain, the visual span for each neuron is limited to a certain area of the entire image that we seeing (or view), so the neuron responds to stimuli only in the restricted visual span which is called a receptive field. Multiple such receptive fields overlap and cover the image (or view). The input to a CNN is a matrix containing values representing intensity of light in each cell (each pixel of the image). For an image with single channel (such as grayscale) there is one matrix as input, for an image with multiple channels (such as RGB) there are multiple matrices as input. CNN that we use have three types of layers - Convolution layers, Pooling layers and Fully-connected layers. We describe these layers below.

The convolution layer We demonstrate the convolution operation that happens at each convolution layer in Figure 4.4. The red matrix is an input image containing a single channel with dimensions $5 \times 5 \times 1$, i.e., there are 5 pixels in the row and 5 pixels in the column of the image. Each pixel contains a numeric value corresponding to the intensity of light in the pixel. The element that performs convolution is called **Kernel K**, which is represented in green. For this example we chose **K** as a $3 \times 3 \times 1$ matrix. The Convoluted feature is shown in blue. The convolution operation starts from the top left by multiplying intensity in each pixel with the corresponding value in **K** and then summing over these to obtain the top left value in the convoluted feature. After getting the top left value (43) for the convoluted feature, the kernel is

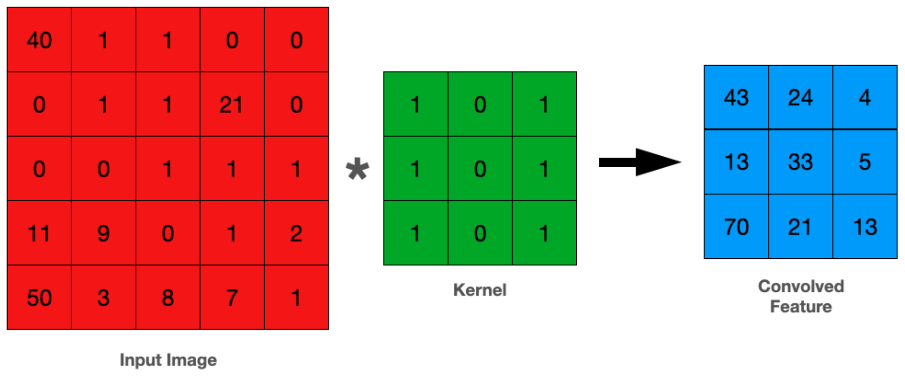


Figure 4.4: Convolution operation on an input image of $5 \times 5 \times 1$ with a kernel of $3 \times 3 \times 1$ and Stride Length = 1.

moved to the adjacent column by **Stride=1** to obtain the next value of the convoluted feature, i.e., 24 in the top middle column. This operation keeps going until the entire image is convoluted with the kernel to obtain all the values for the convoluted feature.

The pooling layer For the example in Figure 4.4 the dimension of the convoluted feature was reduced, however, this is not always the case. The dimension of a convoluted feature might increase if we perform a convolution with stride = 2. So, to reduce the dimension of the feature, pooling is done. Pooling helps to decrease the spatial size of the convoluted feature and as a result it helps in reducing the computational cost for training the CNN. Pooling is useful for extracting dominant features that are positionally and rotationally invariant. There are two types of commonly used pooling, they are: max pooling, and average pooling. An example of each of these is depicted in Figure 4.5).

Fully connected layer The output from convolution layers is flattened and fed in to fully connected layers. The fully connected layers have an activation function and hence they learn non-linear representations of the features that the convolution layers have learnt. The output from the fully connected layer is fed to a softmax layer for classification. A loss is computed using the output of the softmax layer, the ground truth and the loss function; backprop-

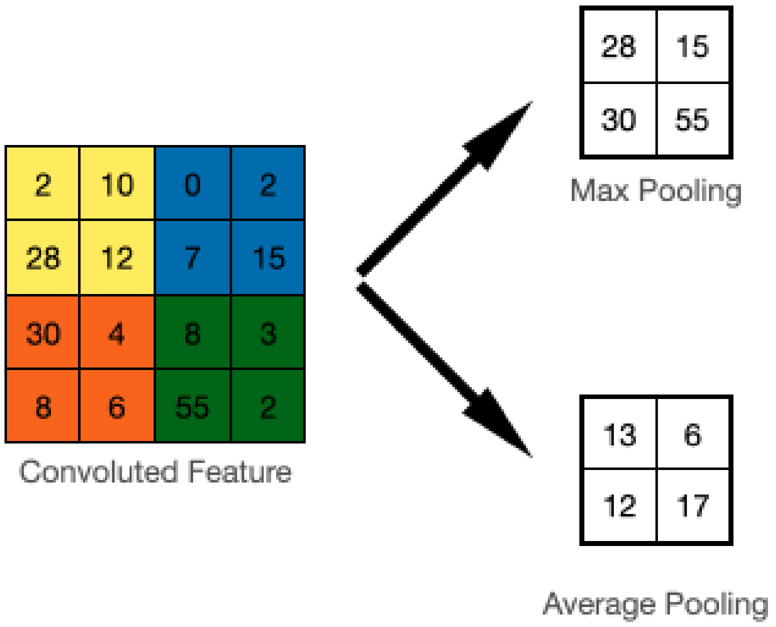


Figure 4.5: Pooling operation via average pooling and max pooling

agation is used to compute the gradient on every iteration of training. We depict a CNN in Figure 4.6.

ResNet

The first CNN by Lecun *et al.* [24] is known as 'LeNet'. As research in CNN architectures kept growing, the community learnt that deeper CNN are better because the model becomes more capable of learning features since the parameter space to explore increases. However, there are two challenges that have to be addressed with the increase in the number of layers - (i) vanishing gradients and (ii) exploding gradients. While training, the loss function is differentiated partially with respect to all of the weights. The weights are updated backwards (from the output layer towards the input layer). The gradient flowing from the following to a subsequent backward layer is multiplied by the input of the backward layer. Weights are multiplied with the gradient again and again as the gradient flows backward. If there are a lot of layers in the network (such

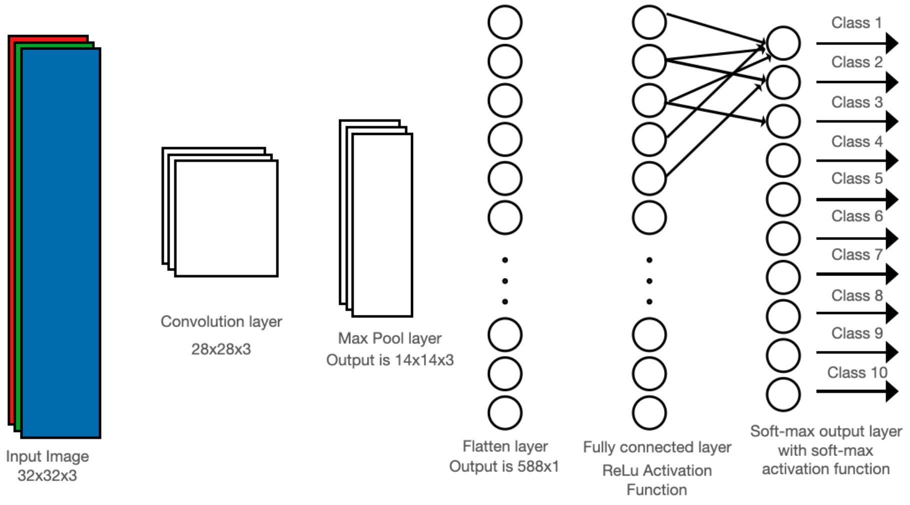


Figure 4.6: Architecture of Convolution Neural Network for classification. Input is an RGB image of dimension 32×32

as in a deep network) the gradients may get value close to zero (vanishing gradients) or they may attain very high values (exploding gradients). Both situations are bad because optimization gets hampered. Exploding gradients can be addressed using batch normalization [17] and gradient clipping [30]. ResNet [13] helps alleviate the problem of vanishing gradient while training Deep CNN.

The architecture of ResNet is such that it helps in preventing gradients from becoming zero. One such residual block is shown in Figure 4.7. ResNet introduces a residual block which has bypass connections in the network. These bypass connections help the flow of gradients without multiplying them with the weights of the layers. ResNet contains multiple such residual blocks and hence the gradients never reach values of zero in deep residual networks. One such example of ResNet with 34 layers is given in Figure 4.8. We use ResNet18 and ResNet34 [13] for our experiments which have 11 million and 22 million parameters respectively.

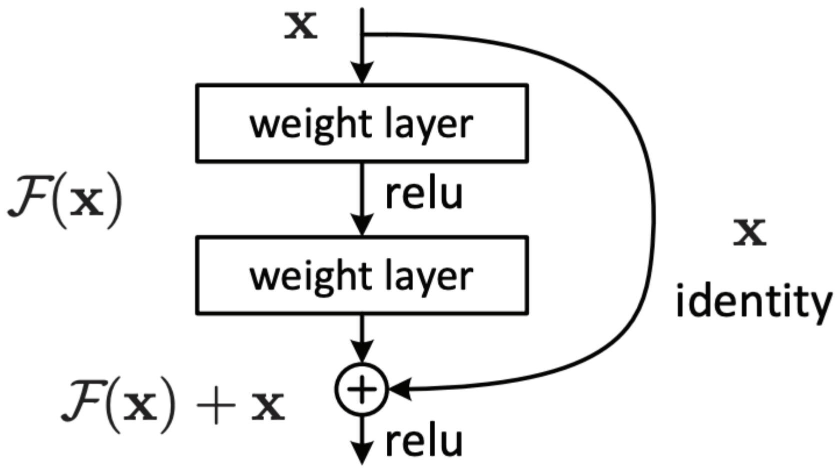

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4.7: Architecture of a building block of ResNet by He *et al.* [13]. Identity branch bypasses the layers and allows the flow of gradients without several multiplications with weights.

4.5 Experiments and results

Our experiments for image classification are divided in two parts - (*i*) Model performance and (*ii*) Training dynamics. In Section 4.5.2, we describe our methodologies and show results for model performance. We describe our methodologies and demonstrate results for training dynamics in the Section 4.5.3. The experimental setup common to both the parts is described below in Section 4.5.1.

4.5.1 Experimental Setup

ResNet18 and ResNet34 models are trained with CIFAR-10 and CIFAR-100 datasets. There are $50K$ training images and predetermined $10K$ test images in each dataset. From the $50K$ images, $40K$ images (selected randomly) are used for training and the remaining $10K$ images are used for parameter tuning. We use CE loss and SGD for training as introduced in Section 2.2.2 and Section 2.2.3 respectively. The output of the last layer of ResNet18 and ResNet34 is passed through a softmax layer before computing the CE loss. We use Nesterov momentum [21] $= 0.9$ as an optimizer along with SGD for optimization, and mini-batch size is kept 128. For ResNet18 and ResNet34 architectures, we use

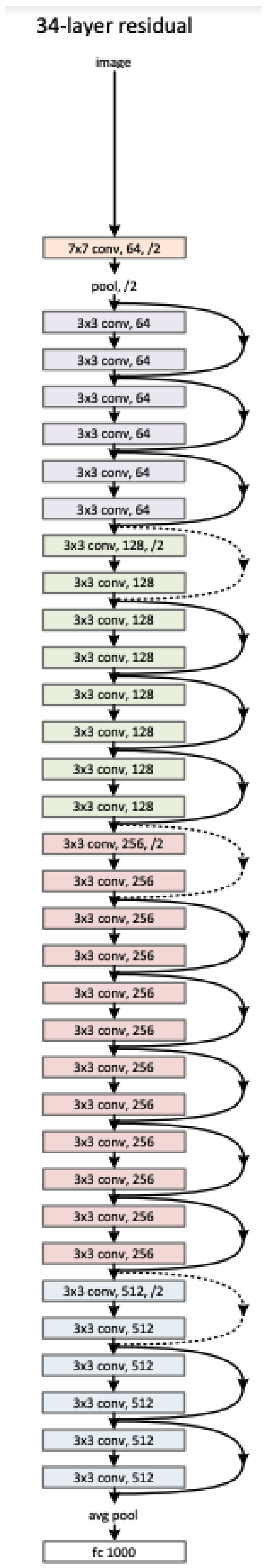
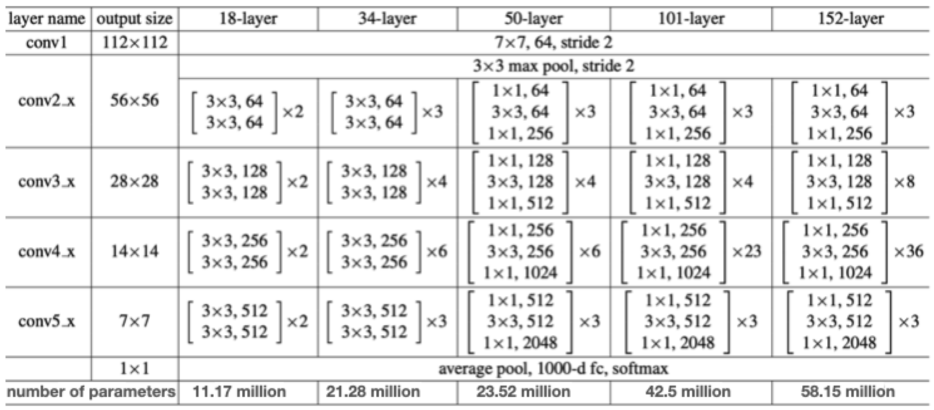


Figure 4.8: ResNet34 as proposed by He *et al.* [13].



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
number of parameters		11.17 million	21.28 million	23.52 million	42.5 million	58.15 million

Figure 4.9: Multiple ResNet architectures as proposed by He *et al.* [13].

Kernel size = 3 × 3, Stride = 1, average pooling is used for the pooling layer and the ReLu activation [7] function is used for the fully connected layers. We employ Batch Normalization [17] to prevent exploding gradients.

4.5.2 Impact of label smoothing on model performance

We use NoLS, ULS and PLS for the experiments here. For CIFAR-100, we use both CLS and SEMLS approaches as presented in Section 4.3. For CIFAR-10, we use SEMLS as presented in Section 4.3. Each of the experiments is run five times at different random seeds. For each run of the experiment, validation loss is used as the criterion for early stopping [37]. We start training with a learning rate of 0.1 and decrease the learning rate twice - after 40 epochs and after 80 epochs, by a factor of 10 times. Weight decay used is 0.0005. We use accuracy as the performance measure since the prior work applying label smoothing on image classification task [33] have used accuracy as the performance measure. The held out test set is used to compare accuracy scores. The results on the test set are presented in Table 4.1 for CIFAR-10 and in Table 4.2 for CIFAR-100.

The results in Table 4.1 and Table 4.2 suggest that when it comes to overall accuracy, CLS and SEMLS (both of which are special cases of PLS) are slightly better than NoLS and ULS for all of the four cases. However, we see that CLS in case of CIFAR-100 and ResNet-34 seems to do slightly better than SEMLS which means that the relationship chosen among the labels matters.

Dataset + Model	**NOLS**	**ULS**	**SEMLS**
CIFAR-10 + ResNet-18	93.809 ± 0.487	94.085 ± 0.273	**94.112 ± 0.568**
CIFAR-10 + ResNet-34	93.762 ± 0.578	93.557 ± 0.608	**93.793 ± 0.514**

Table 4.1: Top-1 classification accuracies of CIFAR-10 dataset with ResNet architectures trained with NoLS, ULS and SEMLS (which is a special case of PLS).

Dataset + Model	**NOLS**	**ULS**	**CLS**	**SEMLS**
CIFAR-100 + ResNet-18	72.789 ± 0.435	72.665 ± 0.323	72.618 ± 0.724	**72.798 ± 0.247**
CIFAR-100 + ResNet-34	71.839 ± 0.668	71.902 ± 0.787	**72.813 ± 0.689**	71.78 ± 0.912

Table 4.2: Top-1 classification accuracies of CIFAR-100 dataset with ResNet architectures trained with NoLS, ULS, CLS and SEMLS (as described in Section 4.3, CLS and SEMLS are special cases of PLS)

Both CLS and SEMLS represent relationships from two different sources of information (CLS comes from knowledge of CIFAR-100 data curators and SEMLS comes from semantic similarity between the class labels), we see that there is a difference in the results when we change this relationship function, suggesting that if these relationships are chosen precisely and carefully then model performance could be further improved. (For instance, in case of CLS, right now we are distributing concentration uniformly among the labels that belong to the same superclass, referring to the Appendix A.1 for 'aquatic mammals' the label 'beaver' is close to the label 'dolphin', if a zoologist gives us a better relationship between these labels, then we can use that relationship for the CLS).

ULS is better than NoLS only twice and worst among all four ones. If we take into account the standard deviation, we can notice that the confidence of all label smoothings overlap with each other. The same observation is made in the results of work by Müller *et al.* [33] as reported in Table 4.3. In their work there is overlap in confidence intervals of models trained on CIFAR-10 and CIFAR-100 datasets. Although they use different models than ours, the observation is consistent as ours - overlapping of confidence intervals. Since the confidence here are overlapping, we expect the overlap to happen in the generalization error too.

Dataset	Architecture	Accuracy($\alpha = 0.0$)	Accuracy($\alpha = 0.1$)
CIFAR-10	ALEXNET	86.8±0.2	86.7±0.3
CIFAR-100	ResNet-56	72.1±0.3	72.7±0.3
IMAGENET	Inception	80.9	80.9

Table 4.3: Top - 1 classification accuracy as reported by Müller *et al.* [33]. In this table α is the smoothing ratio which we are referring to as SR through out this dissertation.

4.5.3 Training Dynamics

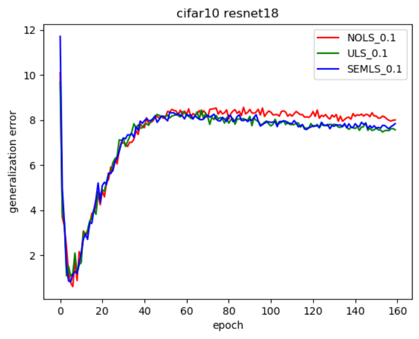
We use NoLS, ULS and SEMLS (which is a special case for PLS) for the experiments here. The goal of the experiments in this section is to study how label smoothing affects training dynamics of a classifier. Since weight decay brings extra regularization to the models, we keep our models free of weight decay so that we can isolate the effect of label smoothing. Getting the best accuracy is not the goal of these experiments rather the goal is to study the training dynamics such as - (*i*) interdependence of gradient norms, learning rate and the type of smoothing while training, (*ii*) interdependence of generalization error, learning rate and type of smoothing while training.

We train ResNet-18 and ResNet-34 on the CIFAR-10 and CIFAR-100 datasets with the same three label smoothing approaches as before. We keep SR = 0.1. We use four different learning rates of $[0.1, 0.01, 0.001, 0.0001]$ to train the classifiers and keep the learning rate constant throughout one training run. This helps us in analysing how changing learning rate may affect the gradient norm and generalization error with different label smoothing approaches. We train for up to 250 epochs, and observed while running experiments that the training loss and test loss changed for all the learning rates (except 0.0001) significantly within the first 100 epochs and do not change after 150 epochs, so we present our plots for up to 160 epochs so that the difference between the plots of ULS and SEMLS is visible. All of the experiments in this section are run five times on different random seed values and all the forthcoming plots are averaged over these five runs.

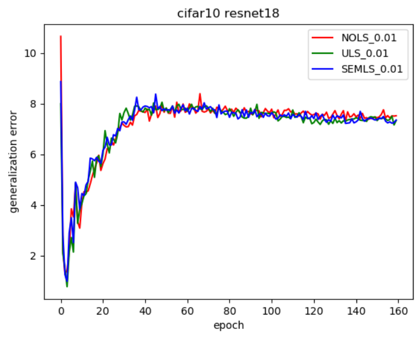
Generalization and learning rate

Generalization error is defined as the absolute difference between training error and test error. The plots of generalization error versus epochs at different learning rates are shown in Figure 4.10, 4.11, 4.12 and 4.13.

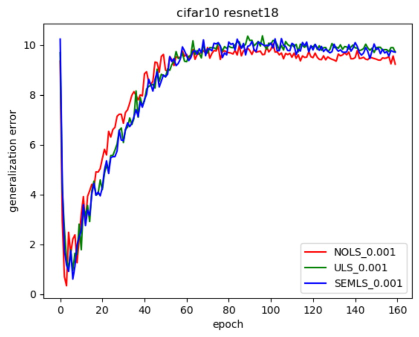
Inferences from generalization plots From the plots it is evident that the generalization error saturates after about 140 epochs except for the plots for the learning rate of 0.0001, indicating that 0.0001 is not reaching the saturation state and so it is not a good learning rate to train and there is no benefit of adding any label smoothing when using such a small learning rate. At learning rates of $[0.1, 0.01, 0.001]$, from the plots it is clear that the generalization curves are overlapping for different label smoothing methods. For CIFAR-100, ULS and SEMLS perform better than NoLS in terms of generalization error but they also overlap very strongly. Overlapping of generalization error is an expected behaviour since the accuracy measures are also overlapping. This suggests that both ULS and SEMLS are more advantaged with CIFAR-100 when the learning rate is high, this can be because there are 100 classes in CIFAR-100. In the previous section 4.5.2 and previous works [33], we observed that the confidence intervals of the accuracy results are not distinct enough to come to a conclusion about generalizability of different label smoothing approaches. The generalization error plots here imply the same. There is overlap in the model performance results in Table 4.1 and Table 4.2 since the plots show that there is overlap among the generalization error curves.



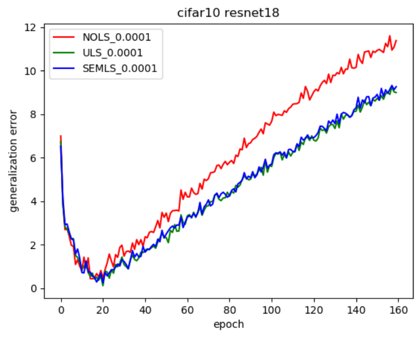
(a) Learning rate = 0.1.



(b) Learning rate = 0.01.

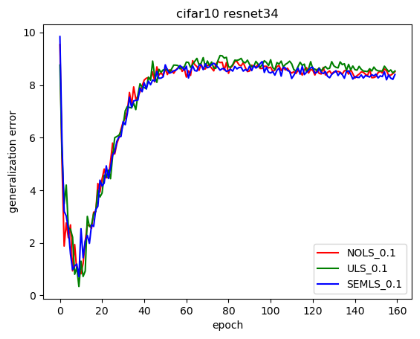


(c) Learning rate = 0.001.

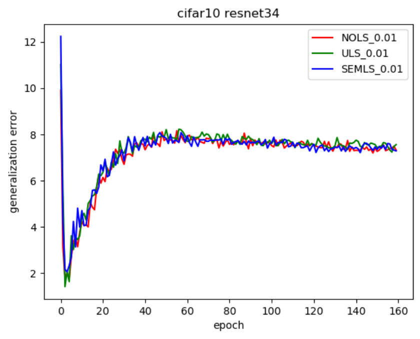


(d) Learning rate = 0.0001.

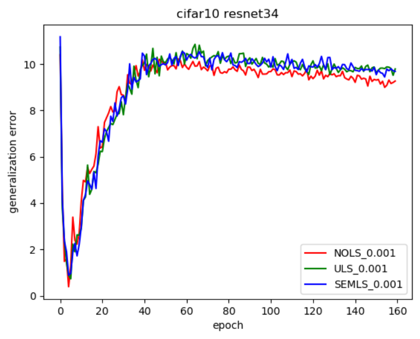
Figure 4.10: Generalization error for ResNet18 trained on CIFAR-10 with different learning rates.



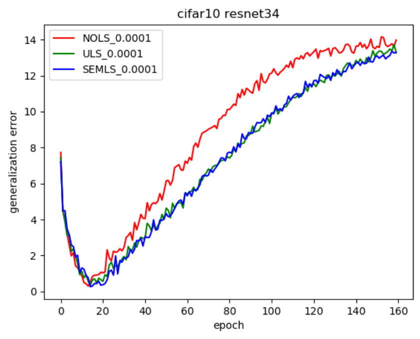
(a) Learning rate = 0.1.



(b) Learning rate = 0.01.

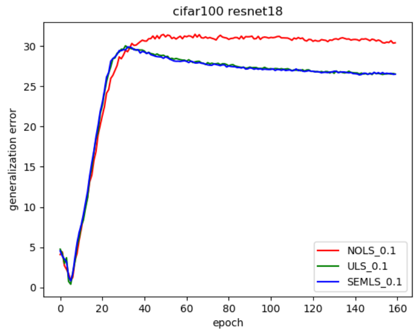


(c) Learning rate = 0.001.

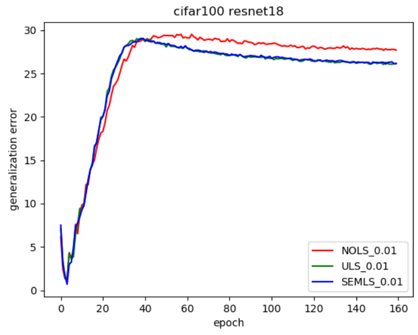


(d) Learning rate = 0.0001.

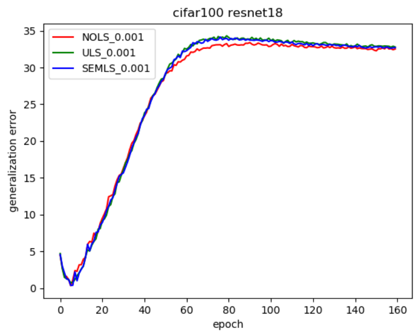
Figure 4.11: Generalization error for ResNet34 trained on CIFAR-10 with different learning rates.



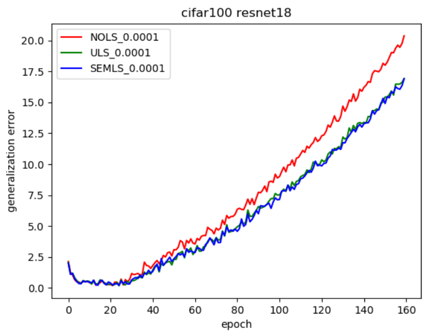
(a) Learning rate = 0.1.



(b) Learning rate = 0.01.

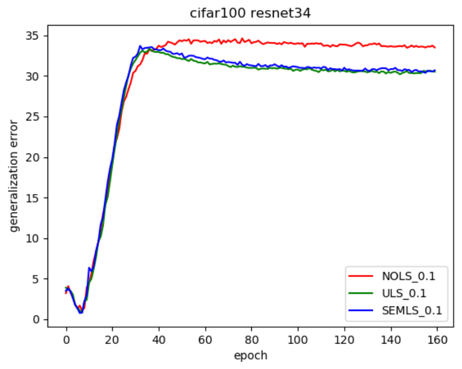


(c) Learning rate = 0.001.

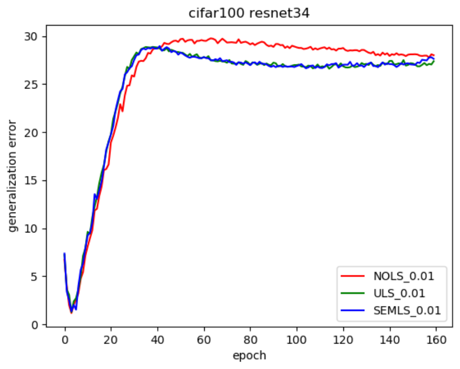


(d) Learning rate = 0.0001.

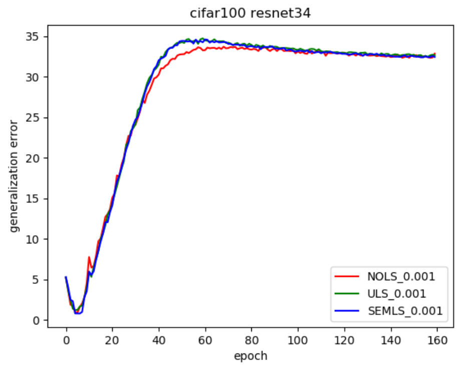
Figure 4.12: Generalization error for ResNet18 trained on CIFAR-100 with different learning rates.



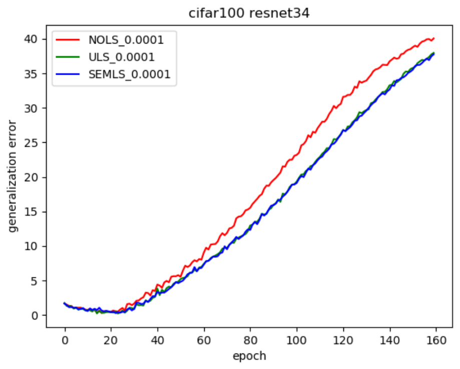
(a) Learning rate = 0.1.



(b) Learning rate = 0.01.



(c) Learning rate = 0.001.



(d) Learning rate = 0.0001.

Figure 4.13: Generalization error for ResNet34 trained on CIFAR-100 with different learning rates.

Gradient norm

Here we experiment with an approach that was proposed by Hardt *et al.* [12] theoretically to define bounds on generalization error. We introduced this mathematically in the Theorem 2.3.1 and Equation 2.15, according to which generalization error is directly proportional to the square of Lipschitz constant, which is proportional to the gradient norm of the weights of the network. Based on Hardt *et al.* [12] we compare gradient norms as the training progresses for the different label smoothing approaches to get an idea of their generalizability. The plots for gradient norm are presented in Figures 4.14, 4.15, 4.16, 4.17.

Takeaways from the plots of gradient norm-

- Gradient norms of models trained on CIFAR-10 (Figures 4.14, 4.15) with ULS and SEMLS are smaller than the gradient norms of models trained with NoLS, which explains why the models trained using NoLS have slightly more tendency to overfit. Except at the end of the training phase, the norms of the models trained with NoLS are closer to the ones trained with ULS and SEMLS. The reason is - the training and test loss do not change by that point.

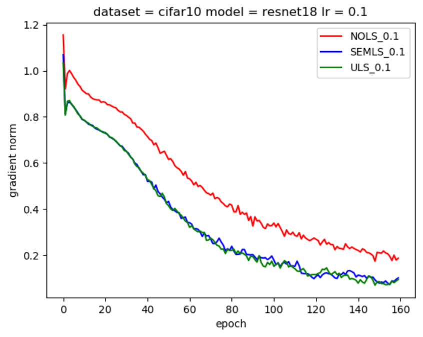
- Gradient norms of models trained on CIFAR-100 (Figures 4.16, 4.17) with ULS and SEMLS are smaller than those with NoLS for the beginning part of the training but the situation changes around 80 epochs (except for learning rate = 0.1).

- For the learning rate of 0.1 in all cases, the gradient norm for models trained with NoLS is larger than those trained with ULS and SEMLS which might be due to the fact that with higher learning rate, the optimizer can move away from the region of non-optimality more quickly, and so the consistency of the gradient norms is maintained through the training process.

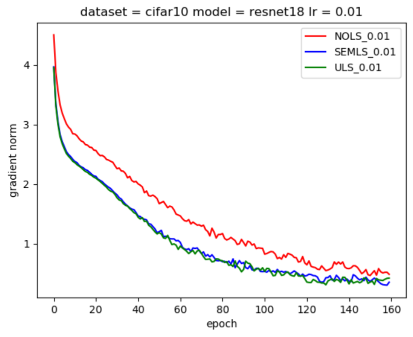
- Gradient norms when using ULS and SEMLS on CIFAR-100 are very close to each other, there is very fine gap between the blue curve and green curve whereas the gap between the blue curve and green curve is larger in the case of CIFAR-10. This may be due to the fact that there are 100 labels in CIFAR-100, when distributing SR (SR = 0.1 means that ground truth gets 0.9 concentration, increasing SR means reducing ground truth concentration so we cannot increase SR) among the non ground truth labels in CIFAR-100, each gets smaller label concentration such that ULS and SEMLS end up assigning the same concentration to the non ground truth labels. In CIFAR-10 there are 10 labels and each non ground truth label gets concentration which might be why we can distinguish between ULS and SEMLS.

- The Gradient norm for all label smoothings, models and datasets increases as the learning rate is decreased, this happens since by decreasing the learning rate, the optimizer moves more slowly to the region of optimality. We observe that for a learning rate of 0.1, the gradient norm follows the pattern as expected,i.e., NoLS has the gradient norm.

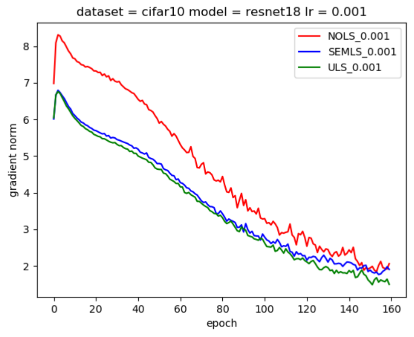
- When the learning rate is 0.0001, no learning is happening and the gradients behave very differently than with other learning rates. This is because the learning rate of 0.0001 is too small and the optimizer is not able to proceed towards a region where it can actually help reducing the training and testing loss.



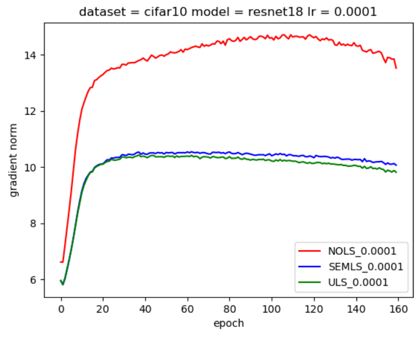
(a) Gradient norms while training ResNet18 on Cifar10 dataset with a learning rate of 0.1.



(b) Gradient norms while training ResNet18 on Cifar10 dataset with a learning rate of 0.01.

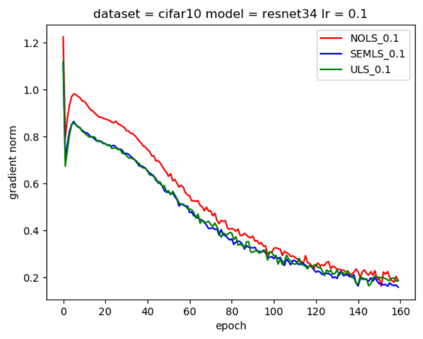


(c) Gradient norms while training ResNet18 on Cifar10 dataset with a learning rate of 0.001.

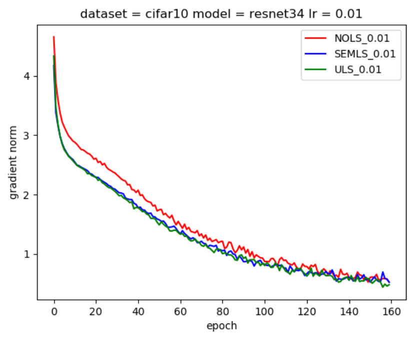


(d) Gradient norms while training ResNet18 on Cifar10 dataset with a learning rate of 0.0001.

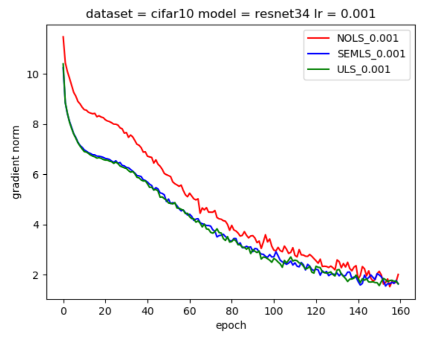
Figure 4.14: Gradient norms obtained while training ResNet18 on Cifar-10 dataset across three smoothing at four different learning rates.



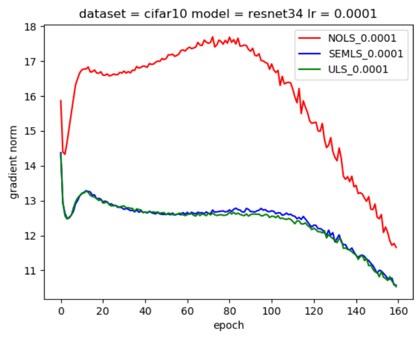
(a) Gradient norms while training ResNet34 on Cifar10 dataset with a learning rate of 0.1.



(b) Gradient norms while training ResNet34 on Cifar10 dataset with a learning rate of 0.01.

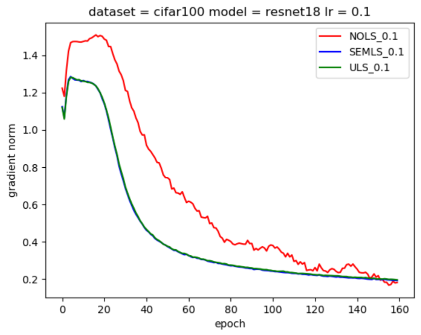


(c) Gradient norms while training ResNet34 on Cifar10 dataset with a learning rate of 0.001.

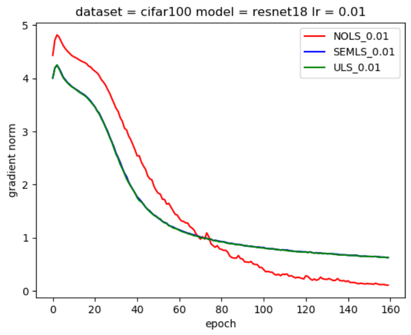


(d) Gradient norms while training ResNet34 on Cifar10 dataset with a learning rate of 0.0001.

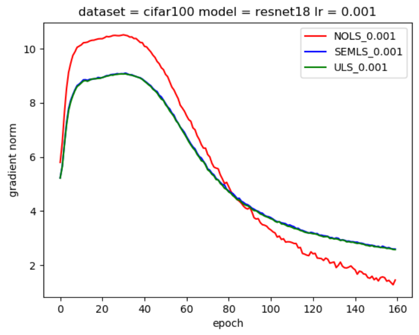
Figure 4.15: Gradient norms obtained while training ResNet34 on Cifar-10 dataset across three smoothing at four different learning rates.



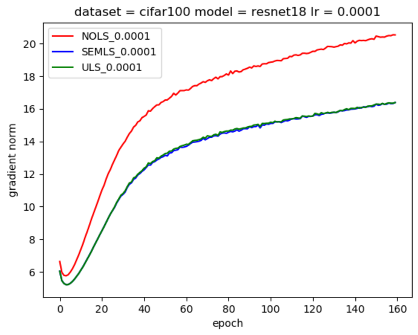
(a) Gradient norms while training ResNet18 on Cifar100 dataset with a learning rate of 0.1.



(b) Gradient norms while training ResNet18 on Cifar100 dataset with a learning rate of 0.01.

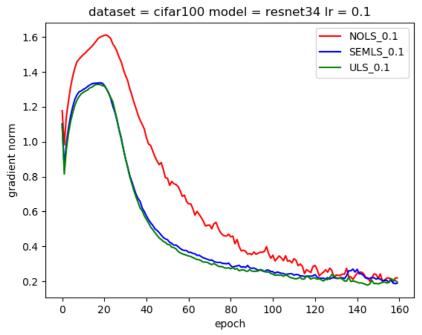


(c) Gradient norms while training ResNet18 on Cifar100 dataset with a learning rate of 0.001.

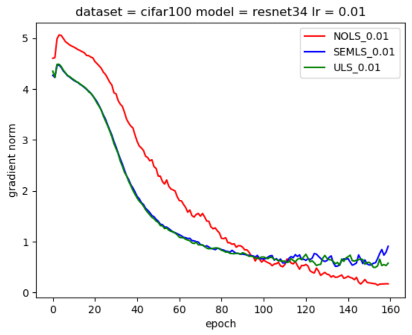


(d) Gradient norms while training ResNet18 on Cifar100 dataset with a learning rate of 0.0001.

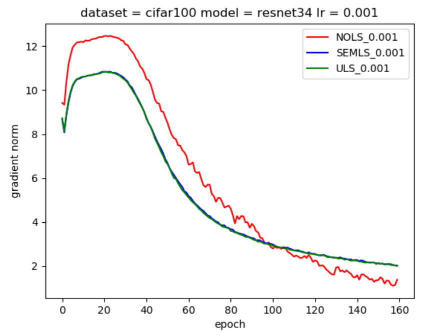
Figure 4.16: Gradient norms obtained while training ResNet18 on Cifar-100 dataset across three smoothing at four different learning rates.



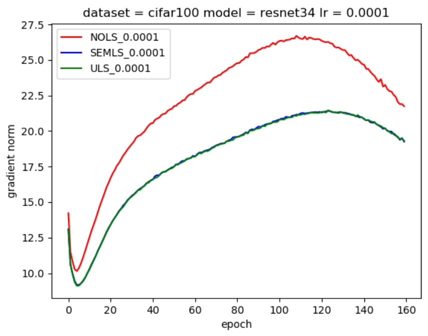
(a) Gradient norms while training ResNet34 on Cifar100 dataset with a learning rate of 0.1.



(b) Gradient norms while training ResNet34 on Cifar100 dataset with a learning rate of 0.01.



(c) Gradient norms while training ResNet34 on Cifar100 dataset with a learning rate of 0.001.



(d) Gradient norms while training ResNet34 on Cifar100 dataset with a learning rate of 0.0001.

Figure 4.17: Gradient norms obtained while training ResNet34 on Cifar-100 dataset across three smoothing at four different learning rates.

4.6 Conclusion

In this chapter, we first review the image classificaiton datasets and the labels in those datasets. We use the formalised definition of PLS from Section 3.3.1 to introduce two label smoothing approaches - CLS and SEMLS. We compare these approaches against ULS and NoLS on the basis of model performance. We compare SEMLS against ULS and NoLS on the basis of generalization error and norm of gradients with changing learning rates. The approach of using the norm of gradients has not been used before in prior works for label smoothing.

The following are our conclusions from the experiments in this chapter

The model performance results suggest that PLS approaches work slightly better than ULS and NoLS. For CIFAR-10, SEMLS (a special case of PLS) has a slightly better performance compared to NoLS and ULS. For CIFAR-100, CLS (a special case of PLS) has a slightly better performance once and SEMLS (a special case of PLS) is slightly better once. By changing the PLS approach, we can see that there is a difference in the model performance suggesting us that the type of relationship among the labels matters. So, if an expert gives us the relationship between the data labels, then the model performance could further be improved.
Generalization error results suggest that there is an overlap in the confidence intervals of training with or without label smoothing, which is also supported by the results of [33]. Generalization error in NoLS is slightly higher than ULS and SEMLS (which is a special case of PLS) for the CIFAR-100 dataset at high learning rates, in CIFAR-10 it is higher sometimes but lower sometimes. This suggests that generalization error is not conclusive for CIFAR-10 but for CIFAR-100 at higher learning rates it is conclusive.

One thing that can be very clearly concluded is that the gradient norm of SEMLS (which is a special case of PLS) and ULS are smaller than those of NoLS for - (*i*) learning with higher learning rates (0.1), and (*ii*) initial phases of learning with smaller learning rates. So, using a larger learning rate in the initial phase with (both ULS and SEMLS) should help in achieving a smaller gradient norm and thus a smaller generalization bound. This bolsters a theoretical claim by Szegedy *et al.* [45] that label smoothing helps in model generalization. Until now, this claim was made using accuracy and standard deviation, the confidence intervals of which were overlapping.

When the number of classes are large in the dataset, i.e., 100 classes in the case of CIFAR-100, the generalization curves of the two smoothing approaches (ULS and SEMLS) overlap very well and the same is observed about the gradient norm curves of the two smoothing approaches. This suggests that the strategy (i.e. ULS or SEMLS) of distributing the concentration among the labels does not play a large role. This may be the case with SEMLS but for other kind of PLS there may be a difference.

Chapter 5

Label Smoothing for Text Classification

5.1 Introduction

Similar to image classification, text classification is a supervised machine learning problem. Instead of images, the input is a document D_i and a suitable label Y_i has to be assigned to D_i. The training set for text classification task is denoted by the pair (D, Y), where $D = [D_1, D_2, \ldots, D_m]$ and $Y = [Y_1, Y_2, \ldots Y_m]$. Each document D_i can be represented as a sequence of words or *tokens* such that $D_i = [d_1, d_2, \ldots, d_n]$, where d_j is the $j - th$ word or token and n is the length of the document. The finite set of all the tokens in all the documents for a given classification task is called *Vocabulary*. In this work, we focus on single-label multi-class text classification, so there is only one label associated with each document. To the best of our knowledge, there are no works studying text classification with label smoothing, so herein we attempt to fill the gap.

Emotion classification is a kind of text classification where each document D has an emotion as its label. The task is - given a document classify which emotion it represents. We study label smoothing for emotion classification for our experiments. The inspiration behind studying the problem of emotion classification is the fact that emotions drive us - due to the COVID-19 pandemic, people were confined and there were many cases of emotional break down. Diagnosing emotions of such people can play an important role in tasks like detecting depression or suicidal intent. Emotion classification can be useful in building emotionally intelligent chatbots whose goal is to detect the emotion expressed and then generate a suitable response. Emotion classification can also be used in the gaming industry, where an emotion expressed by a human can be classified by the gaming engine and this classification may be used as a feedback in generating suitable scenarios in the game. Due to these applications of emotion classification, we study it and apply label smoothing to the problem of emotion classification.

5.2 Word Embeddings

Only numbers can be fed into machine learning models. To train a machine learning model with text data we need to represent text data as numeric data so that the machine learning classifiers can understand it. Word embeddings are used for this task. A word embedding is a learned representation of text where words that have the same meaning have a similar representation.
There are sparse word embeddings such as a Bag of Words (BOW) embedding, where each token in the vocabulary has a unique one-hot vector representation. BOW becomes more sparse as the vocabulary increases, i.e., as the number of tokens in the corpus increases.
GloVe (Global Vectors for Word Representation) and Word2Vec word embeddings that are learned through unsupervised deep learning and have achieved success in multiple Natural Language Processing tasks. These embeddings are continuous and dense vector representations of the tokens in a vocabulary. They also capture the syntactic and semantic relationships between the tokens efficiently. Word2Vec has two models to convert words to a vector representation proposed by Mikolov *et al.* [31]- (*i*) Continuous Bag of Words (CBOW) where the training strategy is to predict the word given its context, (*ii*) Skip-gram where the training strategy is to predict the context given the word. GloVe emphasises the co-occurence of the words in the provided window of the sequence and it was proposed by Pennington *et al.* [35].

5.3 Neural Models for Text Classification

Text is a sequence of words (or tokens) and so, text classification is a sequential learning task and can be modelled well by sequential models based on the Recurrent Neural Network (RNN) such as the Long Short-Term Memory (LSTM). We use the LSTM model for text classification in our experiments.

5.3.1 Recurrent Neural Network

Recurrent Neural Networks RNN are a type of NN distinct from the ones we used in Chapter 4, the difference is that the output for each time step depends upon the current input to the network and the output from the previous time step, the architecture of an RNN is shown in Figure 5.1 with the feedback loop to itself. Figure 5.2 shows the unfolded version. The output equation for an RNN is shown below in 5.1. x_t represents the input at time step t, h_t is a hidden state at time step t, h_t is computed using the hidden state h_{t-1} at time step $t-1$ and the input x_t at time step t. U and W are the learnable parameter matrices and f is the activation function. Pascanu *et al.* [34] show that when training an RNN, the gradient of the loss function decays very fast with the time steps and so the vanishing gradient problem is observed. This means that RNN is not capable of capturing long term dependencies. So training an RNN with longer sequences such as in text classification is not feasible.

$$h_t = f\left(Ux_t + Wh_{t-1}\right) \tag{5.1}$$

5.3.2 Long Short-Term Memory

Hochreiter and Schmidhuber [15] proposed the LSTM to achieve better performance on longer time steps. The LSTM architecture is based on RNN. LSTM units include a 'memory cell' which can capture information in memory for long periods of time. There is a set of gates which control the state of the information - when it enters the memory, when it is output and when it is forgotten. The LSTM block is shown in Figure 5.3. A LSTM has a set of three gates - input gate i_t, forget gate f_t and output gate o_t. The cell state

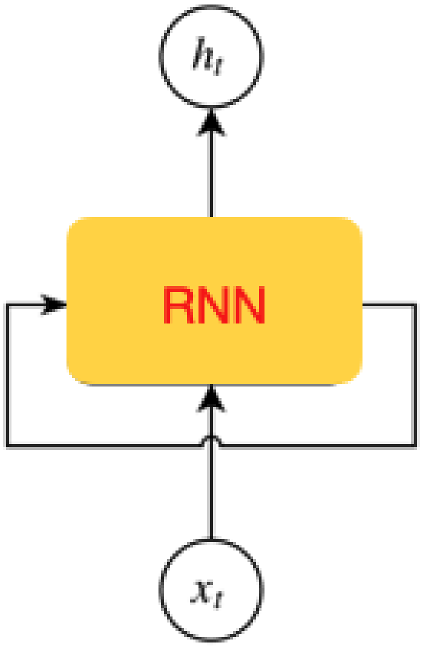


Figure 5.1: Architecture of a basic Recurrent Neural Network.

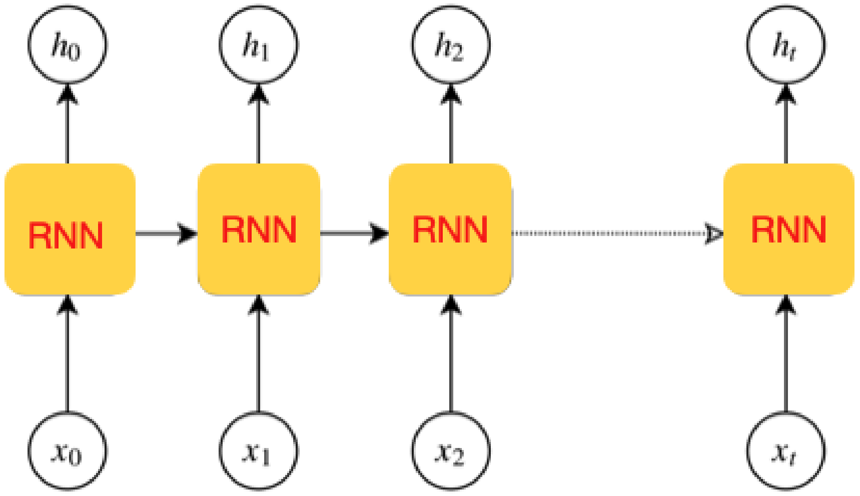


Figure 5.2: Architecture of an unrolled Recurrent Neural Network.

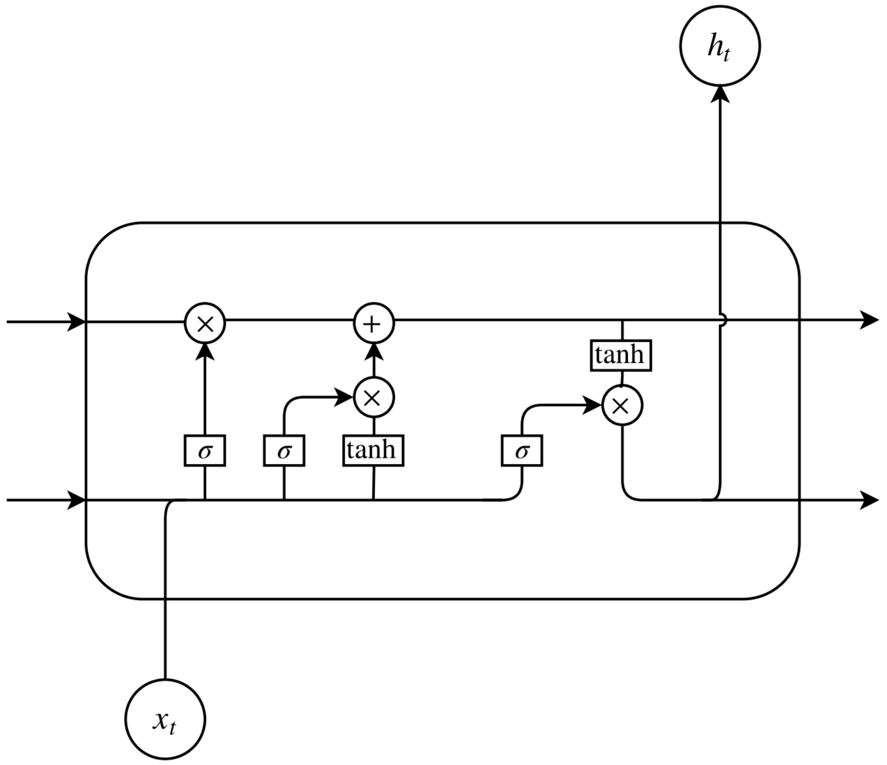


Figure 5.3: LSTM Block [16]

is denoted by c_t. Equations below represent the output of the LSTM. l_t is a temporary parameter. The notations from RNN are used here.

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i + b_i\right) \tag{5.2}$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f + b_f\right) \tag{5.3}$$

$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o + b_o\right) \tag{5.4}$$

$$l_t = \tanh\left(x_t U^l + h_{t-1} W^l + b_l\right) \tag{5.5}$$

$$c_t = f_t * c_{t-1} + i_t * l_t \tag{5.6}$$

$$h_t = o_t * \tanh\left(c_t\right) \tag{5.7}$$

5.4 Datasets

Datasets for emotion classification are either annotated by humans or they are extracted from the internet using some metadata. In our study, we use datasets formed using both strategies. The datasets are described below -

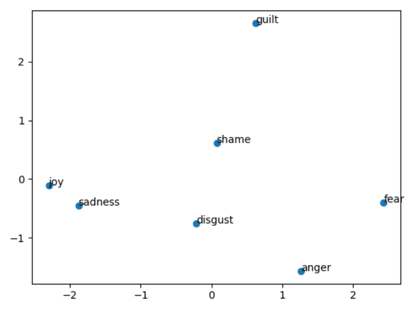
TEC - Twitter Emotion Corpus [32]: This dataset was created from tweets on Twitter. The hashtags with the tweet were assigned as emotion labels to each tweet. This dataset has six emotions as listed in 5.1. This is an unbalanced dataset and the percentage of label distributions are show in 5.1.

CBET - Cleaned Balanced Emotional Tweets [41]: This dataset was formed using tweets and hashtags from the Twitter similar to TEC, the method of dataset formation is available in Shahraki [41]. This dataset has nine emotions as listed in 5.1. Few instances in CBET have more than one emotion, i.e., there are few multi-label instances in CBET, we ignore these multiple label instances and use $76,860$ single labeled instances from CBET. Single labeled instances from CBET represent a balanced dataset.

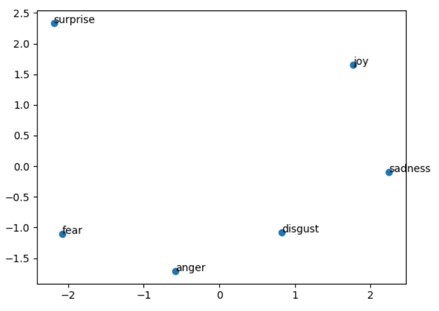
ISEAR - International Survey on Emotion Antecedents and Reactions [39]: This is a human annotated dataset. Psychologists collected this data and asked many students to tag sentences in data from out of seven emotions as listed in 5.1. This is a balanced dataset.
The statistics of these datasets is shown in Table 5.1.

Dataset	No. of labels	No. of instances	Label Distribution
ISEAR (Single label)	7	7,666	anger, disgust, fear, guilt, joy, sadness, shame
TEC (Single label)	6	21,051	anger: 7%, disgust: 4%, fear: 13%, joy: 39%, sadness: 18%, surprise: 18%
CBET (Single + multi label)	9	81,163	anger, disgust, fear, guilt, joy, love, sadness, surprise, thankfulness
CBET (Single label only)	9	76, 860	anger, disgust, fear, guilt, joy, love, sadness, surprise, thankfulness

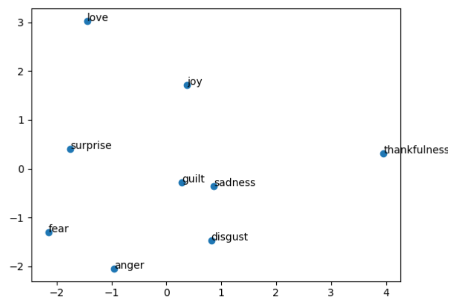
Table 5.1: Statistics of the emotion classification datasets.



(a) ISEAR Dataset



(b) TEC Dataset

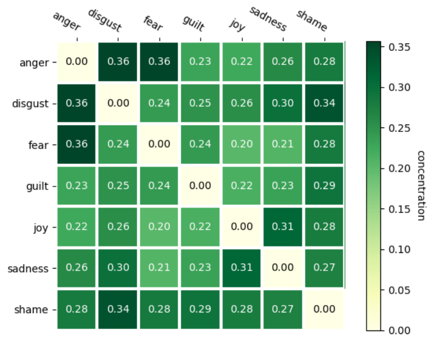


(c) CBET Dataset

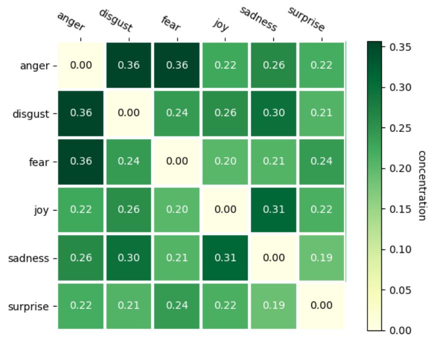
Figure 5.4: The representation of labels in different emotion classification datasets using SEMLS. These were sampled down using PCA from 50 dimensional GloVe embeddings to 2 dimensions for plotting.

5.5 Approach of PLS

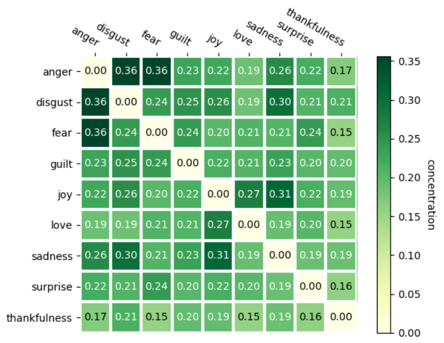
We use SEMLS (which is a special case of PLS) approach as in Section 4.3 for PLS using GloVe for emotion classification task. Figure 5.4 shows the representation of the labels of the datasets used in emotion classification in 2D. Figure 5.5 depicts the pairwise similarity among the labels of emotion datasets, these are the Δ matrices (introduced in Section 3.3.1) for emotion classification. Figure 5.6 has the θ matrices (as introduced in Section 3.3.1) for the emotion classification datasets.



(a) ISEAR Dataset

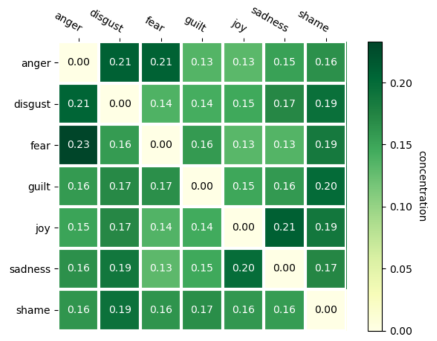


(b) TEC Dataset

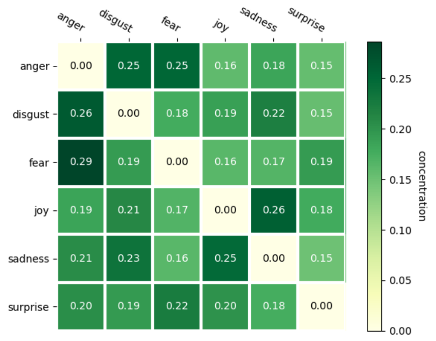


(c) CBET Dataset

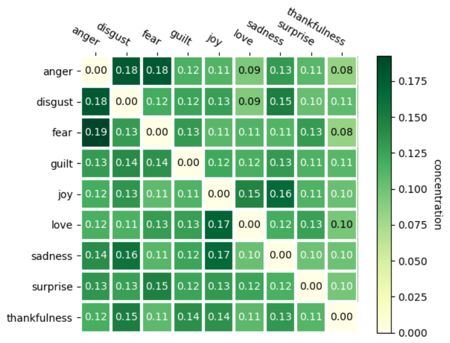
Figure 5.5: Semantic similarity for SEMLS among the labels in the emotion classification datasets, the Δ matrices



(a) ISEAR Dataset



(b) TEC Dataset



(c) CBET Dataset

Figure 5.6: θ matrices for SEMLS with emotion classification datasets

5.6 Experiments and results

Experiments for emotion classification are divided into two parts - (*i*) Model performance is presented in Section 5.6.2, (*ii*) analysing the model performance with label smoothing approaches when the number of classes in the dataset are changed, the results of which are presented in Section 5.6.3. We discuss the performance criterion used to compare the performance of different smoothing techniques on emotion classification datasets in Section 5.6.1. Below we describe the experimental setup for these analyses.

Experimental Setup

For evaluating the model performance, the LSTM [15] model is trained on the single label datasets in Table 5.1. Since, there are no standard train and test split in datasets (ISEAR, TEC and CBET), we use 5-fold cross validation, with the train-test split ratio of 4:1. For splitting data, stratified split is used. The reported results are average and standard deviation.

5.6.1 Performance Criterion

Since the emotion classification datasets do not have clearly defined train and test splits and they are a mixture of balanced and unbalanced datasets, so we do not use accuracy rather we use Macro F1 score as the performance criterion. Macro F1 score is the multiclass version of F1 score (which is for binary classification). Below we discuss how Macro F1 relates with F1 score. For binary classification problem, precision, recall and F1 scores are widely used. Precision, recall and F1 scores for binary class are shown in Equation 5.8. Where TP is True Positive, FP is False Positive, FN is False Negative and TN is True Negative.

$$\begin{aligned} \mathrm{P} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ \mathrm{R} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\ \mathrm{F} &= \left(\frac{\mathrm{P}^{-1}+\mathrm{R}^{-1}}{2}\right)^{-1} \end{aligned} \tag{5.8}$$

For multi class classification the values above can be seen as binary classification where a certain class c_i is positive class and rest all other classes as one negative class, so we use subscript i to denote these quantities for class c_i - P_i, R_i, F_i, TP_i, FP_i, TP_i and TN_i. If the total number of classes is $|C|$, then Macro-averaged precision (MAP) = $\frac{1}{|C|}\sum_i^{|C|} P_i$, Macro-averaged recall (MAR) = $\frac{1}{|C|}\sum_i^{|C|} R_i$ and Macro-averaged F1 (MAF) = $(\frac{\text{MAP}_i^{-1}+\text{MAR}_i^{-1}}{2})^{-1}$.

5.6.2 Model Performance

The results for experiments on ISEAR, TEC and CBET with different label smoothing approaches are presented below in Table 5.2. Note that the re-

Dataset No.	**NOLS**	**ULS**	**SEMLS**
ISEAR	0.5802 ± 0.0382	0.5871 ± 0.0313	**0.5906 ± 0.0351**
TEC	**0.5212 ± 0.0373**	0.5184 ± 0.0331	0.5148 ± 0.0341
CBET	0.5601 ± 0.0397	0.5688 ± 0.0362	**0.5702 ± 0.0323**

Table 5.2: Macro averaged F1 score on emotion classification datasets trained with LSTM model with NoLS, ULS and SEMLS. SR is kept 0.1 for these experiments because this means that ground truth gets 0.9 concentration and we shouldn't reduce concentration on ground truth too much.

sults with the best Macro-averaged F1- score are highlighted. SEMLS has the slightly better performance for ISEAR (well curated dataset made by human annotations) and CBET (dataset which has more training samples than the other two). There can be an explanation for this - (*i*) ISEAR case - SEMLS might be capturing the relationship between emotion labels, while annotating the data it might be possible that humans are also considering the relationship of labels with the sentence they are annotating. (*ii*) CBET case - SEMLS might be working slightly better because this dataset is larger than the other two. So, when we train our models with data labeled cleanly from human annotators, SEMLS might be providing some kind of leverage for the model to learn slightly better. Training models using SEMLS when we have more data (case of CBET) then SEMLS might be facilitating better performance.

5.6.3 Effect of label smoothing on changing the number of classes in dataset

For experiments related to the effect of changing the number of classes in a dataset, we take CBET dataset and vary the number of classes in the dataset, i.e., we sample sets of three datasets with 3, 5 and 7 classes from the CBET dataset, (i.e., three datasets with 3 classes, three datasets with 5 classes and three datasets with 7 classes), and then evaluate the model performance on these datasets. The results of this experiment are presented in Tables 5.3, 5.4, 5.5.

When using 3 classes, ULS works well on Dataset 1, SEMLS works well on Dataset 2 and ULS works well on Dataset 3. When using 5 classes, NoLS is better for Dataset 1, ULS is better for Dataset 2 and SEMLS is better for Dataset 3. With 7 classes, ULS, SEMLS and NoLS are better for Dataset 1, Dataset 2 and Dataset 3, respectively. From the tables it is not certain which label smoothing has the best performance with the change in the number of classes in the dataset.

Dataset	**NOLS**	**ULS**	**SEMLS**
3 Classes Dataset 1	0.5731 ± 0.0321	0.5738 ± 0.0301	$\mathbf{0.5753 \pm 0.0306}$
3 Classes Dataset 2	0.5811 ± 0.03201	0.5813 ± 0.0334	$\mathbf{0.5847 \pm 0.0318}$
3 Classes Dataset 3	0.5719 ± 0.0329	$\mathbf{0.5848 \pm 0.0387}$	0.5841 ± 0.0339

Table 5.3: Macro averaged F1 score on emotion classification datasets of 3 classes sampled from CBET, trained with LSTM model with NoLS, ULS and SEMLS.

Dataset	NOLS	ULS	SEMLS
5 Classes Dataset 1	**0.5718 ± 0.0381**	0.5702 ± 0.0341	0.5715 ± 0.0321
5 Classes Dataset 2	0.5724 ± 0.0318	**0.5731 ± 0.0394**	0.5714 ± 0.0397
5 Classes Dataset 3	0.5701 ± 0.0329	0.5742 ± 0.0314	**0.5759 ± 0.0326**

Table 5.4: Macro averaged F1 score on emotion classification datasets of 5 classes sampled from CBET, trained with LSTM model with NoLS, ULS and SEMLS.

Dataset No.	NOLS	ULS	SEMLS
7 Classes Dataset 1	0.5703 ± 0.0312	**0.5753 ± 0.0341**	0.5714 ± 0.0342
7 Classes Dataset 2	0.5706 ± 0.0317	0.5738 ± 0.0334	**0.5743 ± 0.0324**
7 Classes Dataset 3	0.5707 ± 0.0359	0.5717 ± 0.0332	**0.5730 ± 0.0318**

Table 5.5: Macro averaged F1 score on emotion classification datasets of 7 classes sampled from CBET, trained with LSTM model with NoLS, ULS and SEMLS.

5.7 Conclusion

In this chapter, first we introduced the problem of text classification and emotion classification. Second, we discussed the components required for text classification such as word embeddings and the LSTM model. Further we introduced datasets for emotion classification. We discussed approach of SEMLS (which is a special case of PLS) for emotion datasets. In Section 5.6.2, given the experiments, we observe that SEMLS might be more favourable to the carefully human annotated dataset and dataset having more data samples (and collected from twitter). When it comes to model performance in Section 5.6.3, with the changing number of classes, we observe that for the three classes case, SEMLS (a special case of PLS) does slightly better than ULS and NoLS for two datasets. For the five classes case SEMLS, ULS and NoLS do well for one dataset. For the seven classes case SEMLS does slightly better for two datasets. Overall, SEMLS is slightly better for five datasets while ULS is slightly better for three datasets. The confidence intervals of these results are overlapping which is suggesting that there is not a certain conclusion.

Chapter 6

Conclusion

6.1 Summary

Label smoothing is a regularization technique used to address overfitting in neural networks. In label smoothing, some concetration from the ground truth labels is taken and distributed among the non ground labels equally. In this manuscript, we explored the concept of PLS (Preferential Label Smoothing) - assigning label concentration to non ground truth labels based on their relationship to the ground truth labels. The approaches of PLS that we introduce here are -

- CLS (Cluster Label Smoothing) where the preference is approximated by the group of superclass a label belongs to, such as super classes in CIFAR-100 (Appendix A.1). This type of label smoothing represents a label smoothing suggested by the expert of the field, i.e., the curators of the CIFAR-100 dataset.

- SEMLS (Semantic Label Smoothing) which is based on the distance between a word embedding representation of the label words.

Reflecting back to the initial thesis statement:

How do preferential label smoothing and uniform label smoothing affect model performance and training dynamics compared to no label smoothing?

To attain our objective of studying model performance, we experimented with Image Classification and Text Classification. We evaluate their performance when trained with NoLS, ULS and PLS. For image classification, standard CIFAR-10 and CIFAR-100 datasets were used. SEMLS was experimented on CIFAR-10 while both SEMLS and CLS were tried on CIFAR-100. These were implemented with ResNet-18 and ResNet-34 models. Accuracy was used as the performance criterion. We find that SEMLS works slightly better on the CIFAR-10 dataset with both models. For the CIFAR-100 dataset SEMLS works slightly better for one model, and CLS works better for the other model. Suggesting that the choice of the preferential function that we choose could make a difference. For Text Classification, we use the LSTM classifier and evaluate its performance on three datasets - TEC, CBET, ISEAR. We use Macro-average F1 as the performance criterion. For ISEAR dataset, which is the smallest of all datasets and which is labeled by human annotators, we observe that SEMLS is slightly better. For TEC, it seems that NoLS is slightly better, for CBET, it seems that SEMLS is slightly better. So, the SEMLS is slightly better for dataset created using human knowledge (ISEAR) and dataset with many data samples (CBET).

We also study the model performance when the number of classes in the dataset is changed in text classification. For this we vary the number of classes in CBET dataset and observe the macro-averaged F1 score. We found that the results are inconclusive because the confidence intervals are overlapping, However, for our experimental setting, SEMLS is slightly better five times, ULS is slightly better thrice and NoLS is better once. This is the case with SEMLS but it can be different for other PLS approaches.

To attain our objective of studying training dynamics, we experimented with Image classification under minimal regularization (we only used label smoothing as the regularization). We study generalization error and smoothing approaches at different learning rates and plot generalization curves. We find that at faster learning rates ($[0.1, 0.01]$), the generalization error when using ULS and SEMLS is smaller or equal to generalization error when using NoLS. At slower learning rates, ULS and SEMLS have higher generalization errors

than NoLS, or at very slow learning rates, training is not supported at all.
In Chapter 2, we formalized that the generalization bound for a NN model depends on the norm of the gradients during training. One more experimental study for training dynamics that we do is, to study the gradient norms during the training phase of the network at different learning rates. We study gradient norms during the training phase and find that gradient norms of SEMLS and ULS are smaller than those of NoLS when learning with a higher learning rate (0.1) and during the initial phase of training, so ULS and SEMLS should help in reaching a lower generalization bound. This is a result that was empirically never shown before anywhere with regard to label smoothing to prove label smoothing helps achieve a lower generalization bound.
Also, when the number of classes in the dataset is large, i.e., 100 for CIFAR-100, gradient norm curves of ULS and SEMLS overlap with each very well, and the same is observed in generalization error suggesting that since there are 100 labels, SR gets distributed so much that SEMLS and ULS end up assigning close to very same concentration to non-ground truth labels. This may be the case only for our approach of PLS, i.e., SEMLS but there might exist an approach of PLS which is better than ULS.

6.2 Future Work

The possible future directions to pursue could be:

- Next step in future research would be to experiment with SEMLS on the Imagenet dataset for images (which has 1000 classes, 10 times more than CIFAR-100), which would give a clear idea about the effect of PLS when the number of labels in the dataset is very high. For a similar experiment for the emotion classification problem, it would be interesting to see PLS with the Goemotions Dataset, which has 27 labels for emotion classification.

- Golatkar *et al.* [10] shows that for training a NN, regularization methods such as Weight Decay and Data Augmentation do not have significant effect on generalization if they are removed after the initial transient period of training. In some cases the generalization improves if the regularization is removed after initial transient period. However, Golatkar *et al.* [10] do not analyse label smoothing with respect to when is the appropriate time during training to apply label smoothing. So, running an analysis for label smoothing will help in understanding when is the appropriate time to apply label smoothing during training phase.

- Few prior works [26], [28], [33] study effects of label smoothing on model calibration. Doing a similar study for SEMLS and CLS and identifying whether SEMLS and CLS are effective when it comes to calibrating a NN would be useful.

- We would like to study how PLS can be applied to multi label classification problems, we explain the reason in the next sentences. Currently, to address class imbalance in multilabel problems, oversampling (among other techniques) is used for minority class and undersampling (among other techniques) is used for majority class. Using PLS with an appropriate relation matrix could be useful in dealing with multilabel problem.

- PLS can also be used for sequence generation tasks such as - machine translation [43] and dialogue generation. At each step of generating these sequences, the model has to decide which word should be the next word from the vocabulary based on the target sequence used for training, and then CE loss is used as a loss function. So PLS and SEMLS can be potentially more useful because it will allow assigning concentration to target using vector representation for the targets.

References

[1] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, pp. 499–526, Jun. 2002. DOI: 10.1162/153244302760200704.

[2] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*, F. F. Soulié and J. Hérault, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 227–236, ISBN: 978-3-642-76153-9.

[3] J. Degrave, F. Felici, J. Buchli, *et al.* "Magnetic control of tokamak plasmas through deep reinforcement learning." (2022), [Online]. Available: https://www.nature.com/articles/s41586-021-04301-9 (visited on 02/16/2022).

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[5] A. H. F. Dinevari and O. R. Zaiane, "Automated nursing agent: A software agent for at-home elderly care," in *eTELEMED 2016*, 2016.

[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html.

[7] K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological Cybernetics*, vol. 20, no. 3, pp. 121–136, Sep. 1975, ISSN: 1432-0770. DOI: 10.1007/BF00342633. [Online]. Available: https://doi.org/10.1007/BF00342633.

[8] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, Jun. 2016, pp. 1050–1059. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html.

[9] U. Gasser and V. A. Almeida, "A layered model for ai governance," *IEEE Internet Computing*, vol. 21, no. 6, pp. 58–62, 2017. DOI: `10.1109/MIC.2017.4180835`.

[10] A. Golatkar, A. Achille, and S. Soatto, "Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 1321–1330.

[12] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, New York, NY, USA: JMLR.org, 2016, pp. 1225–1234.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv e-prints*, arXiv:1512.03385, arXiv:1512.03385, Dec. 2015. arXiv: `1512.03385 [cs.CV]`.

[14] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015. arXiv: `1503.02531`. [Online]. Available: `http://arxiv.org/abs/1503.02531`.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. [Online]. Available: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[16] C. Huang. "Towards emotion intelligence in neural dialogue systems." (2019), [Online]. Available: `https://era.library.ualberta.ca/items/7f192eb3-04a5-458d-a71e-cb7d67ef85b5/view/29ec1ccd-1254-4f0d-bbb9-45e0b0fbe501/Huang_Chenyang_201906_MSc.pdf`.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456. [Online]. Available: `https://proceedings.mlr.press/v37/ioffe15.html`.

[18] J. Jumper, R. Evans, A. Pritzel, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, ISSN: 1476-4687. DOI: `10.1038/s41586-021-03819-2`. [Online]. Available: `https://doi.org/10.1038/s41586-021-03819-2`.

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, arXiv:1412.6980, arXiv:1412.6980, Dec. 2014. arXiv: 1412.6980 [cs.LG].

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[21] G. Lan, "An optimal method for stochastic composite optimization," *Math. Program.*, vol. 133, no. 1–2, pp. 365–397, Jun. 2012, ISSN: 0025-5610. DOI: 10.1007/s10107-010-0434-y. [Online]. Available: https://doi.org/10.1007/s10107-010-0434-y.

[22] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989, ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, ISSN: 1558-2256. DOI: 10.1109/5.726791.

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

[25] S. Lee and D.-H. Choi, "Energy management of smart home with home appliances, energy storage system and electric vehicle: A hierarchical deep reinforcement learning approach," *Sensors*, vol. 20, no. 7, 2020, ISSN: 1424-8220. DOI: 10.3390/s20072157. [Online]. Available: https://www.mdpi.com/1424-8220/20/7/2157.

[26] W. Li, G. Dasarathy, and V. Berisha, *Regularization via structural label smoothing*, 2020. DOI: 10.48550/ARXIV.2001.01900. [Online]. Available: https://arxiv.org/abs/2001.01900.

[27] M. Lukasik, H. Jain, A. K. Menon, *et al.*, *Semantic label smoothing for sequence to sequence problems*, 2020. DOI: 10.48550/ARXIV.2010.07447. [Online]. Available: https://arxiv.org/abs/2010.07447.

[28] M. Maher and M. Kull, "Instance-based label smoothing for better calibrated classification networks," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 746–753. DOI: 10.1109/ICMLA52953.2021.00124.

[29] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI Magazine*, vol. 27, no. 4, p. 12, Dec. 2006. DOI: 10.1609/aimag.v27i4.1904. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/1904.

[30] T. Mikolov. "Statistical language models based on neural networks." (2012), [Online]. Available: `https://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf`.

[31] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. [Online]. Available: `http://arxiv.org/abs/1301.3781`.

[32] S. Mohammad, "#emotional tweets," in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 246–255. [Online]. Available: `https://aclanthology.org/S12-1033`.

[33] R. Müller, S. Kornblith, and G. Hinton, "When Does Label Smoothing Help?" *arXiv e-prints*, arXiv:1906.02629, arXiv:1906.02629, Jun. 2019. arXiv: `1906.02629 [cs.LG]`.

[34] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., ser. Proceedings of Machine Learning Research, vol. 28, Atlanta, Georgia, USA: PMLR, Jun. 2013, pp. 1310–1318. [Online]. Available: `https://proceedings.mlr.press/v28/pascanu13.html`.

[35] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[36] B. Perry and R. Uuk, "Ai governance and the policymaking process: Key considerations for reducing ai risk," *Big Data and Cognitive Computing*, vol. 3, no. 2, 2019, ISSN: 2504-2289. DOI: `10.3390/bdcc3020026`. [Online]. Available: `https://www.mdpi.com/2504-2289/3/2/26`.

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

[38] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: `10.1007/s11263-015-0816-y`.

[39] K. R. Scherer and H. G. Wallbott, ""evidence for universality and cultural variation of differential emotion response patterning": Correction.," *Journal of Personality and Social Psychology*, vol. 67, no. 1, pp. 55–55, 1994. DOI: 10.1037/0022-3514.67.1.55. [Online]. Available: https://doi.org/10.1037/0022-3514.67.1.55.

[40] A. Sekhari, K. Sridharan, and S. Kale, "Sgd: The role of implicit regularization, batch-size and multiple-epochs," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 27 422–27 433. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/e64c9ec33f19c7de745bd6b6d1a7a86e-Paper.pdf.

[41] A. G. Shahraki. "Emotion mining from text." (2015), [Online]. Available: https://era.library.ualberta.ca/items/27ae961f-d9a6-4a5a-9b6f-f180478ea573/view/e18f9340-7a28-462b-9d03-67c226bf8aa1/Gholipour-20Shahraki_Ameneh_201509_MSc.pdf.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.

[43] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 3104–3112.

[44] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *arXiv e-prints*, arXiv:1512.00567, arXiv:1512.00567, Dec. 2015. arXiv: 1512.00567 [cs.CV].

[46] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention Is All You Need," *arXiv e-prints*, arXiv:1706.03762, arXiv:1706.03762, Jun. 2017. arXiv: 1706.03762 [cs.CL].

[47] L. Yuan, F. E. H. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," 2019. DOI: 10.48550/ARXIV.1909.11723. [Online]. Available: https://arxiv.org/abs/1909.11723.

[48] O. Zaiane and S. Murray. "Magnetic control of tokamak plasmas through deep reinforcement learning." (2022), [Online]. Available: https://www.amii.ca/latest-from-amii/ana-automated-nursing-agent/ (visited on 08/31/2020).

[49] C.-B. Zhang, P.-T. Jiang, Q. Hou, *et al.*, "Delving deep into label smoothing," *IEEE Transactions on Image Processing*, vol. 30, pp. 5984–5996, 2021. DOI: 10.1109/tip.2021.3089942. [Online]. Available: https://doi.org/10.1109%2Ftip.2021.3089942.

[50] Z. Zhang and M. Sabuncu, "Self-distillation as instance-specific label smoothing," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 2184–2195. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1731592aca5fb4d789c4119c65c10b4b-Paper.pdf.

[51] J. Zhao, T. Qu, and F. Xu, "A deep reinforcement learning approach for autonomous highway driving," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 542–546, 2020, 3rd IFAC Workshop on Cyber-Physical and Human Systems CPHS 2020, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.04.142. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589632100272X.

Appendix A

Additional Info

A.1 CIFAR-100 Class Labels

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table A.1: Class labels name in CIFAR-100 [20]