UNIVERSITY OF CALIFORNIA SAN DIEGO

Surprising Empirical Phenomena of Deep Learning and Kernel Machines

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Like Hui

Committee in charge:

Professor Mikhail Belkin, Chair
Professor Yoav Freund
Professor Julian Mcauley
Professor Lily Weng
Professor Stephen Wright

2023

The Dissertation of Like Hui is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

DEDICATION

Dedicated to my family for their love and support.

EPIGRAPH

Don’t adventures ever have an end? I suppose not. Someone else always has to carry on the story.

The Fellowship of the Ring

TABLE OF CONTENTS

Dissertation Approval Page iii

Dedication iv

Epigraph v

Table of Contents vi

List of Figures ix

List of Tables xi

Acknowledgements xiii

Vita xvi

Abstract of the Dissertation xvii

Chapter 1 Introduction 1
- 1.1 Motivation 2
- 1.2 Loss function in multi-class classification 2
- 1.3 Shallow and deep models 3

Chapter 2 Square loss vs. Cross-entropy in classification 5
- 2.1 Introduction 6
- 2.2 Experiments 10
  - 2.2.1 NLP experiments 11
  - 2.2.2 Automatic Speech Recognition (ASR) experiments 13
  - 2.2.3 Computer vision experiments 15
- 2.3 Performance across different initializations 16
- 2.4 Observations during training 17
- 2.5 Implementation 18
- 2.6 Summary and discussion 20
- 2.7 Acknowledgements 20

Chapter 3 Precise asymptotic of rescaled square loss 22
- 3.1 Introduction 22
- 3.2 Related work 24
- 3.3 Preliminaries 26
  - 3.3.1 Data 26
  - 3.3.2 1-layer relu network 27
  - 3.3.3 Large system limit 29
  - 3.3.4 Main results 29
  - 3.3.5 State evolution of 1-relu VAMP and its fixed point solution 30

3.4 Numerical Results 33
3.4.1 Simulation results on synthetic data 33
3.4.2 Results on real data 36
3.5 Acknowledgements 36

Chapter 4 Cut your Losses with Squentropy 37
4.1 Introduction 37
4.2 The squentropy loss function 39
4.3 Experiments 42
4.3.1 Empirical results on test performance 44
4.3.2 Empirical results on calibration 46
4.3.3 Additional results on 121 Tabular datasets 49
4.3.4 Robustness to initialization 52
4.4 Observations 52
4.4.1 Predicted probabilities and decision boundary 52
4.4.2 Weight norm 53
4.5 Rescaled squentropy 54
4.6 Summary, thoughts, future investigations 55
4.7 Acknowledgements 56

Chapter 5 Limitation of Neural Collapse on Understanding Generalization in Deep Learning 57
5.1 Introduction 58
5.1.1 Related Works 60
5.1.2 Notation 63
5.2 Defining Neural Collapse 63
5.2.1 Remarks on Feasibility 67
5.3 Experiments: Train and Test Collapse 69
5.3.1 Measuring Collapse 69
5.3.2 Experimental Results 70
5.4 Collapsed Features Transfer Worse 74
5.4.1 Test Collapse implies Bad Representations 74
5.4.2 Experiments 76
5.5 Conclusion 77
5.6 Acknowledgements 78

Chapter 6 Kernel Machines in Speech Enhancement 79
6.1 Introduction 79
6.2 Kernel-Based Speech Enhancement 81
6.2.1 Kernel Machines 81
6.2.2 Exponential Power Kernel 82
6.2.3 Automatic Subbands Adaptive Kernels 83
6.3 Experimental Results 85
6.3.1 Regression Task 85

6.3.2 Classification Task 85
6.3.3 Single Kernel and Subband Adaptive Kernels 86
6.3.4 Time Complexity 87
6.4 Conclusion and Discussion 88
6.5 Acknowledgements 88

Chapter 7 Conclusion 89
7.1 Contributions 91
7.2 Future work 92

Appendix A 93
A.1 Datasets and tasks 93
A.2 Hyper-parameter settings 95
A.2.1 Hyper-parameters for NLP tasks 95
A.2.2 Hyper-parameters for ASR tasks 95
A.2.3 Hyper-parameters for vision tasks 96
A.3 Experimental results on validation and training sets 97
A.4 Our results compared with the original work 98
A.5 Regularization terms 100
A.6 Variance of accuracy among different random seeds 100

Appendix B 104
B.1 Proof of Lemma 4 104
B.2 Stieltjes Transform 106

Appendix C 110
C.1 Datasets 110
C.2 Hyperparameters 110
C.3 More reliability diagrams 111
C.4 Results for 121 tabular datasets 111

Appendix D 119
D.1 Experimental setup for figure 5.3 119
D.2 Experimental setup for transfer learning 119
D.3 Proof of Lemma 5 120

Bibliography 122

LIST OF FIGURES

Figure 2.1. Difference between accuracy (or error rate) between square loss and CE for each initialization. (Square loss acc. - CE acc.) is shown for accuracy, (CE - Square loss) for error rate. 16

Figure 2.2. Training curves 18

Figure 3.1. Flow of parameter $\boldsymbol{w}$ to model output Y 31

Figure 3.2. *Left:* The test accuracy with different rescaling parameter R for dataset with different class number k. *Right:* The test accuracy with different rescaling parameter R for dataset with different sample complexity $\beta = n/d$. 34

Figure 3.3. The test accuracy with different rescaling parameter R for dataset with different level of noise σ_ε is the variance of the noise. Note that when $\sigma_\varepsilon = 0$ means no label noise. 34

Figure 3.4. The prediction $\hat{z}$ of $y_{ij} = 1$ and $y_{ij} = 0$ for the i-th sample. 35

Figure 3.5. Optimal R for different class number k 35

Figure 3.6. The test accuracy of CIFAR-100. Note that for $k = 100$ case, we choose a subset with 5000 training samples to speed up. 36

Figure 4.1. Confidence histograms (top) and reliability diagrams (bottom) for a Wide Resnet on CIFAR-100. 46

Figure 4.2. Test accuracy and model calibration of 121 tabular datasets from [30] trained with a 3 layer (64-128-64) fully connected network. The results for each dataset are averaged over 5 runs with different random initializations. 48

Figure 4.3. Decision boundary along different epochs for test samples. 51

Figure 4.4. Weight norm along training. 53

Figure 5.1. Failure of Test Collapse. Neural Collapse for ResNet18 on CIFAR-10. Collapse appears to occurs on the train set, but not on test. 58

Figure 5.2. Neural Collapse on CIFAR-10. Collapse occurs on the train set, but not on the test set (neither Strong nor Weak). 71

Figure 5.3. Failure of Test Collapse. Training and test variance vs. SGD iterations, for various dataset and architecture combinations. All test sets (black line) do not collapse to negligible variance, and have much less collapse than the train sets (purple line). 72

Figure 5.4. Train vs. Test Anti-Correlation. 72

Figure 5.5. Collapsed Features Transfer Worse. 76

Figure 6.1. Kernel-based speech enhancement framework 81

Figure 6.2. MSE along per frequency channel 87

Figure A.1. Accuracy/error rate variance of results among 5 random seeds 102

Figure C.1. Reliability diagrams for a pretrained BERT on text5 data. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 111

Figure C.2. Reliability diagrams for a pretrained BERT on text20 data. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 112

Figure C.3. Reliability diagrams for a Transformer-XL on enwik8. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 112

Figure C.4. Reliability diagrams for a Transformer-XL on text8. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 112

Figure C.5. Reliability diagrams for a Attention+CTC model on TIMIT. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 113

Figure C.6. Reliability diagrams for a VGG+BLSTMP model on WSJ. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss. 113

Figure C.7. Reliability diagrams for a VGG+BLSTM model on Librispeech. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss. 113

Figure C.8. Reliability diagrams for a TCN on MNIST. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 114

Figure C.9. Reliability diagrams for a Resnet18 on CIFAR-10.*Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss. 114

Figure C.10. Reliability diagrams for a Wide Resnet on CIFAR-100 subset. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss. 114

Figure C.11. Reliability diagrams for a Resnet18 on STL10. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 115

Figure C.12. Reliability diagrams for a VGG on SVHN. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss. 115

LIST OF TABLES

Table 2.1. NLP task statistics and descriptions 12

Table 2.2. NLP results, accuracy 12

Table 2.3. NLP results, F1 scores 13

Table 2.4. ASR task statistics and descriptions 14

Table 2.5. ASR results, error rate 14

Table 2.6. Vision task statistics and descriptions 15

Table 2.7. Vision results, accuracy 16

Table 2.8. Standard deviation of test accuracy/error. Smaller number is bolded. 17

Table 2.9. Rescaling parameters 19

Table 4.1. Test performance (perf(%): accuracy for NLP&Vision, error rate for speech data) and calibration: ECE(%). 45

Table 4.2. Standard deviation of test accuracy/error. Smaller number is bolded. CE is short for cross-entropy. 50

Table 4.3. Test accuracy/error rate, and scaled sqen is short for rescaled squentropy. CE is short for cross-entropy. 55

Table 6.1. Kernel & DNN on TIMIT: (MSE: lowest is best, STOI and PESQ: highest is best. Best results bolded.) 84

Table 6.2. Kernel & DNN on HINT 86

Table 6.3. Comparison of kernel machines with 1 subband and 4 subbands 86

Table 6.4. Running time/epochs of Kernel & DNN 87

Table A.1. Hyper-parameters for NLP tasks 96

Table A.2. Hyper-parameters for ASR tasks 96

Table A.3. Hyper-parameters for vision tasks 97

Table A.4. NLP results on validation set, accuracy 97

Table A.5. NLP results on validation set, F1 scores 98

Table A.6. ASR results on validation set, error rate 98

Table A.7. NLP results on training and test set, accuracy 99

Table A.8. NLP results on training and test set, F1 scores 99

Table A.9. ASR results on training and test set, error rate 100

Table A.10. Vision results on training and test set, accuracy 100

Table A.11. Training with the cross-entropy loss, our results and the reported ones 101

Table A.12. Regularization term for each task 103

Table C.1. Hyper-parameters for CIFAR-100, SVHN, and STL-10. 111

Table C.2. Test accuracy (Acc)/ECE for 121 tabular datasets 116

Table C.3. Test accuracy (Acc)/ECE for 121 tabular datasets 117

Table C.4. Test accuracy (Acc)/ECE for 121 tabular datasets 118

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to all those who have supported and contributed to the completion of this thesis.

First and foremost, I am profoundly grateful to my supervisor, Professor Mikhail Belkin for his guidance, expertise, and unwavering support throughout this research journey. I am lucky to work with him, and his valuable insights, constructive feedback, and encouragement have been instrumental in shaping this thesis and my research during my Ph.D. journey. I learn a lot from his critical thinking, curiosity to fundamental research questions, and his last-long research taste. Misha always points me to the right direction and gives valuable suggestions under different stages of this journey. Mostly importantly, he makes realize what types of research are good and what kind of research I'd like to do.

I am also thankful to the members of my thesis committee, Yoav Freund, Julian Mcauley, Lily Weng and special thanks to Stephen Wright, for their time, expertise, and valuable suggestions. Their insightful comments and constructive criticisms have immensely contributed to the refinement of this work.

I would also like to extend my gratitude to all of my co-authors, including Siyuan Ma, Preetum Nakkiran, Parthe Pandit, Stephen Wright and Chaoyue Liu. Siyuan guided me to be familiar with the work of the lab and we collaborated on my first project at the lab. The discussions with him really helped me to build the initial confidence for the start of my Ph.D.. I am lucky to collaborate with Preetum for the neural collapse project. I learn a lot from his broad knowledge in machine learning, critical thinking in science and good time management. Parthe is the one who introduces me a lot detailed theoretical knowledge with great patience, and I did learn the thinking style of theory people from him. I am also very lucky to be able to collaborate with Steve for the squentropy paper. I will always remember his rigorous thinking, great attention to details and hard working, and learn from him. I would also give special thanks to Chaoyue, who introduces and explains me a lot of theoretical staff and he answers all of my questions with great patience. He can always give simple answers that I can understand for even very hard questions. It is nice to have him as a labmate.

I would like to acknowledge my co-authors participants of this study, without whom this research would not have been possible. Their willingness to contribute their time, insights, and data has enriched this thesis and added depth to its findings.

I would also like to thank all other past or current labmates, Libin Zhu, Neil Mallinar, Amirhesam Abedsoltan, Abhishek Roy, Jonathan Shi, Parsa Mirtaheri, Soumik Mandal, and Justin Eldridge. The discussion with them on research and life sharing social events made my Ph.D. life colorful and not lonely. Special thanks to Libin Zhu and Abhishek Roy, as we discuss much more in personal life and also research projects.

I would like to thank all my mentors during my internships, Yong Xu, Chao Weng, Jianming Liu, Dong Yu, Mahaveer Jain, Duc Le and Yun Wang. The collaboration with them made me saw what kinds of projects are like in the industry. Their support, encouragement and suggestions made my internships successful and enjoyable.

I would like to express my heartfelt appreciation to my parents, Xianfu Hui and Junlian Hua for their unconditional love and support. Their unwavering believe on me always gives me courage to pursue what I desire and persist the right thing instead of easy choice under difficult situations. I am really grateful to my husband, Wuwei Lan, who always understand me well and provide support and encouragement for me to be better. I am grateful to my grandfather and great-grandmother who have left us but their optimism and perseverance will always be with me. Finally, I was lucky to have my little son, who is so cute and makes me experience the miracle of life. I also thank all other family members from both my father's side, my mother's side and my parents in-law. Their belief in my abilities and constant encouragement have been my driving force throughout this academic pursuit.

In conclusion, this thesis would not have been possible without the invaluable support and contributions of the aforementioned individuals and institutions. Although their names may be listed here, the magnitude of their impact on this research goes far beyond words. I am truly grateful for their presence in my academic journey.

Chapter 2, in full, is a reprint of Like Hui, and Mikhail Belkin. “Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks.” ICLR 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a preprint of Like Hui, Parthe Pandit, Mikhail Belkin, “Precise asymptotics of Rescaled square loss for Multiclass Classification”. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of Like Hui, Mikhail Belkin, and Stephen Wright. “Cut your Losses with Squentropy.” ICML, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of Like Hui, Mikhail Belkin, and Preetum Nakkiran. “Limitations of neural collapse for under- standing generalization in deep learning.” arXiv preprint arXiv:2202.08384 (2022). The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, is a reprint of Like Hui, Siyuan Ma, and Mikhail Belkin. “Kernel Machines Beat Deep Neural Networks on Mask-based Single-channel Speech Enhancement”, Interspeech 2019. The dissertation author was the primary investigator and author of this paper.

VITA

2014	B.S. in Information Science and Engineering, Central South University, Changsha, Hunan, China
2017	M.S. of Electronic Engineering, Tsinghua University, Beijing, China
2023	Ph.D. in Computer Science, University of California San Diego, La Jolla, California, USA

PUBLICATIONS

Like Hui, Mikhail Belkin, and Stephen Wright. “Cut your Losses with Squentropy.” ICML, 2023.

Like Hui, Mikhail Belkin, and Preetum Nakkiran. “Limitations of neural collapse for understanding generalization in deep learning.” arXiv preprint arXiv:2202.08384 (2022).

Like Hui, and Mikhail Belkin. “Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks.” ICLR 2021.

Like Hui, Siyuan Ma, and Mikhail Belkin. “Kernel Machines Beat Deep Neural Networks on Mask-based Single-channel Speech Enhancement”, Interspeech 2019.

Chaoyue Liu, and Like Hui. “ReLU soothes the NTK condition number and accelerates optimization for wide neural networks.” arXiv preprint arXiv:2305.08813 (2023).

Yong Xu, Chao Weng, Like Hui, Jianming Liu, Meng Yu, Dan Su, and DongYu. “Joint Training of Complex Ratio Mask Based Beamformer and Acoustic Model for Noise Robust ASR”, Proceedings of The 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019).

ABSTRACT OF THE DISSERTATION

Surprising Empirical Phenomena of Deep Learning and Kernel Machines

by

Like Hui

Doctor of Philosophy in Computer Science

University of California San Diego, 2023

Professor Mikhail Belkin, Chair

Over the past decade, the field of machine learning has witnessed significant advancements in artificial intelligence, primarily driven by empirical research. Within this context, we present various surprising empirical phenomena observed in deep learning and kernel machines. Among the crucial components of a learning system, the training objective holds immense importance. In the realm of classification tasks, the cross-entropy loss has emerged as the dominant choice for training modern neural architectures, widely believed to offer empirical superiority over the square loss. However, limited compelling empirical or theoretical evidence exists to firmly establish the clear-cut advantage of the cross-entropy loss. In fact, our findings demonstrate that training with the square loss achieves comparable or even better results than the

cross-entropy loss, even when computational resources are equalized.

However, it remains unclear how the rescaling hyperparameter *R*, needs to vary with the number of classes. We provide an exact analysis for a 1-layer ReLU network in the proportional asymptotic regime for isotropic Gaussian data. Specifically, we focus on the optimal choice of *R* as a function of (i) the number of classes, (ii) the degree of overparameterization, and (iii) the level of label noise. Also, we provide empirical results on real data, which supports our theoretical predictions.

Afterwards, to avoid extra parameters brought by the rescaling of the square loss (in cases when class number is large), later on we propose the "squentropy" loss, which is the sum of the cross-entropy loss and the average square loss over the incorrect classes. We show that the squentropy loss outperforms both the pure cross entropy and rescaled square losses in terms of the classification accuracy and model calibration. Also, squentropy loss is a simple "plug-and-play" replacement of cross-entropy as it requires no extra hyperparameters and no extra tuning on optimization parameters.

Also, we investigate the model part of a learning system by applying theoretically well-understood kernel machines to practical challenging tasks, speech enhancement. We found that kernel machines actually outperform fully connected networks and require less computation resources. We investigate the Neural Collapse phenomenon proposed by Papyan, Han, & Donoho (2020), which gives the precise formulation of the patterns of features and classifiers during the terminal phase of training. We study the correlation between neural collapse and generalization in deep learning and we give precise definitions and their corresponding feasibility on generalization, which clarify neural collapse concepts. Moreover, our empirical evidence supports our claim that neural collapse is mainly an optimization phenomenon.

Chapter 1

Introduction

Deep learning has revolutionized the field of artificial intelligence and machine learning, enabling remarkable advancements in various domains such as computer vision, natural language processing, and robotics. Modern deep learning systems are at the forefront of cutting-edge research and real-world applications, pushing the boundaries of what is possible in terms of pattern recognition, data analysis, and decision-making. Typically, a machine learning system includes 4 main components, training objectives, model architectures, data preprocessing and optimizers.

Training objectives play a fundamental role in deep learning systems. These objectives define the tasks that the models aim to accomplish. For instance, in image classification, the objective could be to correctly assign a label to an input image, while in natural language processing, it could involve generating coherent and contextually relevant text. Defining clear and appropriate training objectives is crucial for guiding the learning process and evaluating the performance of the models accurately.

In this thesis, I will mainly discuss the selection of training objectives for modern classification tasks and the phenomenons happened during the terminal phase (training beyond zero training error to close zero training loss) of training classifiers. Finally we focus on the model part and compare the kernel machines, which are shallow models to deep fully-connected networks.

1.1 Motivation

It is exciting to see the development of machine learning in recent years, and especially the big improvement brought by deep learning to many kinds of applications. The come out of ChatGPT seems to bring us to a new stage in our way to artificial general intelligence. Nowadays, models that give best performance usually have huge parameters and are trained with a large amount of data. However, the complexity of machine learning systems brings challenges to the theoretical analysis of the model. That is, people mostly could not predict the outcome by changing the parameters in realistic machine learning systems. Then the system can be somewhat uncontrolled and bring risks to many situations.

On the other hand, in theoretical machine learning, the analysis of optimization and generalization are mostly for very simple models, such as linear models, quadratic models, or two-layer ReLU networks. Also, there are some other gaps between theory and practice in machine learning, such as the loss function used in practice for classification is mostly cross-entropy, while in theoretical analysis it is mostly based on the square loss, especially in the interpolation regime.

The motivation of the work of this thesis is to have fundamental understanding of the important components in modern machine learning and we would like to see whether choices that are good for theoretical analysis, such as square loss or kernel machines can get good generalization results in practical tasks.

1.2 Loss function in multi-class classification

The cross-entropy loss function is commonly used for classification, particularly in practical applications. It is widely adopted by popular toolkits such as Huggingface [135], Espnet [133], and torchvision [84]. While the literature often analyzes the loss function for binary classification [31], real-world classification tasks predominantly involve multi-class scenarios [73]. Our research specifically targets multi-class classification, where we extensively evaluate the performance of various loss functions through systematic empirical studies. These experiments encompass multiple benchmarks in NLP, speech, and vision, and encompass diverse modern neural architectures.

In our initial findings, we discovered that the square loss can yield comparable or even superior results to the widely-used cross-entropy loss in most of our experiments. However, when the number of classes, denoted as C, is large, it becomes necessary to rescale the square loss. Subsequently, we present a detailed asymptotic analysis of the rescaled square loss in multi-class classification. We establish a relationship between the optimal rescaling parameter, the class number, and the degree of overparameterization.

Moreover, we introduce a novel loss function called ”squentropy.” Notably, our experiments demonstrate that squentropy achieves the highest test accuracy and superior calibration results in the majority of cases. Compared with the square loss, squentropy eliminates the need for rescaling.

1.3 Shallow and deep models

DNNs are powerful models that consist of multiple layers of interconnected artificial neurons. These networks are capable of learning complex patterns and representations from input data through a process called training. During training, the parameters of the network, such as weights and biases, are adjusted to minimize a given loss function and improve the model's performance on a specific task, such as classification or regression.

On the other hand, Neural Tangent Kernel (NTK) is proposed [56] as mathematical tools that provide insights into the convergence and generalization behavior of deep neural networks. Afterwords, a rich literature focus on understanding the properties of different deep architectures based on the NTK regime. The NTK approach focuses on analyzing the kernel function associated with a DNN. The kernel function measures the similarity between pairs of inputs and plays a fundamental role in many machine learning algorithms.

The key connection between DNNs and NTKs lies in the observation that as the width of a DNN with certain activation functions approaches infinity, the dynamics of the network become governed by the NTK. In this so-called ”neural tangent regime,” the evolution of the network during training can be described by the NTK, allowing for theoretical analysis and understanding of DNN behavior.

By studying the NTK, researchers have gained insights into various aspects of DNNs, such as the initialization, optimization landscape, generalization properties, and learning dynamics. The NTK framework has been particularly useful for analyzing the behavior of DNNs in the overparameterized regime, where the number of parameters exceeds the number of training samples.

Overall, the connection between DNNs and NTKs enables researchers to gain theoretical understanding and make practical advancements in training and analyzing deep neural networks. Essentially deep neural networks can be taken as kernel machines under the NTK regime.

Hence, we think understanding deep models requires more understanding of kernel machines and we compare kernel machines to deep fully-connected networks in a challenging speech enhancement tasks. The observation is that kernel machines can give even better results and also requires less computation resources than the deep fully-connected networks. Also with more recent techniques developed for kernel machines [2] to deal with large datasets and fast optimization, we believe that understanding kernel machines is the starting point to understand deep models.

Chapter 2

Square loss vs. Cross-entropy in classification

Modern neural architectures for classification tasks are trained using the cross-entropy loss, which is widely believed to be empirically superior to the square loss. In this work we provide evidence indicating that this belief may not be well-founded. We explore several major neural architectures and a range of standard benchmark datasets for NLP, automatic speech recognition (ASR) and computer vision tasks to show that these architectures, with the same hyper-parameter settings as reported in the literature, perform comparably or better when trained with the square loss, even after equalizing computational resources. Indeed, we observe that the square loss produces better results in the dominant majority of NLP and ASR experiments. Cross-entropy appears to have a slight edge on computer vision tasks.

We argue that there is little compelling empirical or theoretical evidence indicating a clear-cut advantage to the cross-entropy loss. Indeed, in our experiments, performance on nearly all non-vision tasks can be improved, sometimes significantly, by switching to the square loss. Furthermore, training with square loss appears to be less sensitive to the randomness in initialization. We posit that training using the square loss for classification needs to be a part of best practices of modern deep learning on equal footing with cross-entropy.

2.1 Introduction

Modern deep neural networks are nearly universally trained with cross-entropy loss in classification tasks. To illustrate, cross-entropy is the only loss function specifically discussed in connection with training neural networks for classification in popular references [40, 142]. It is the default for classification in widely used packages such as NLP implementation Hugging Face Transformers [135], speech classification by ESPnet [133] and image classification implemented by torchvision [84]. Yet we know of few empirical evaluations or compelling theoretical analyses to justify the predominance of cross-entropy in practice. In what follows, we use a number of modern deep learning architectures, including convolutional neural networks and Transformers, and standard datasets across the range of tasks of natural language processing, speech recognition and computer vision domains as a basis for a systematic comparison between the cross-entropy and square losses. The square loss (also known as the Brier score [7] in the classification context) is a particularly useful basis for comparison since it is nearly universally used for regression tasks and is available in all major software packages. To ensure a fair evaluation, for the square loss we use hyper-parameter settings and architectures exactly as reported in the literature for cross-entropy, with the exception of the learning rate, which needs to be increased in comparison with cross-entropy and, for problems with a large number of classes (42 or more in our experiments), loss function rescaling (see Section 2.5).

Our evaluation includes 28 separate learning tasks[1] (neural model/dataset combinations) evaluated in terms of the error rate or, equivalently, accuracy (depending on the prevalent domain conventions). We also provide some additional domain-specific evaluation metrics – F1 for NLP tasks, and Top-5 accuracy for ImageNet. Training with the square loss provides accuracy better or equal to that of cross-entropy in 22 out of 28 tasks.

These results are for averages over multiple random initalizations, results for each

[1] We note WSJ and Librispeech datasets have two separate classification tasks in terms of the evaluation metrics, based on the same learned acoustic model. We choose to count them as separate tasks.

individual initialization are similar. Furthermore, we find that training with the square loss has smaller variance with respect to the randomness of the initialization in the majority of our experiments.

Our results indicate that the models trained using the square loss are not just competitive with same models trained with cross-entropy across nearly all tasks and settings but, indeed, provide better classification results in the majority of our experiments. The performance advantage persists even when we equalize the amount of computation by choosing the number of epochs for training the square loss to be the same as the optimal (based on validation) number of epochs for cross-entropy, a setting favorable to cross-entropy.

Note that with the exception of the learning rate, we utilized hyper-parameters reported in the literature, originally optimized for the cross-entropy loss. This suggests that further improvements in performance for the square loss can potentially be obtained by hyper-parameter tuning.

Based on our results, we believe that the performance of modern architectures on a range of classification tasks may be improved by using the square loss in training. We conclude that the choice between the cross-entropy and the square loss for training needs to be an important aspect of model selection, in addition to the standard considerations of optimization methods and hyper-parameter tuning.

A historical note.

The modern ubiquity of cross-entropy loss is reminiscent of the predominance of the hinge loss in the era of the Support Vector Machines (SVM). At the time, the prevailing intuition had been that the hinge loss was preferable to the square loss for training classifiers. Yet, the empirical evidence had been decidedly mixed. In his remarkable thesis [115], Ryan Rifkin conducted an extensive empirical evaluation and concluded that *"the performance of the RLSC* [square loss] *is essentially equivalent to that of the SVM* [hinge loss] *across a wide range of problems, and the choice between the two should be based on computational tractability* *considerations"*. More recently, the experimental results in [107] show an advantage to training with the square loss over the hinge loss across the majority of the tasks, paralleling our results in this paper. We note that conceptual or historical reasons for the current prevalence of cross-entropy in training neural networks are not entirely clear.

Theoretical considerations.

The accepted justification of cross-entropy and hinge loss for classification is that they are better "surrogates" for the 0-1 classification loss than the square loss, e.g. [40], Section 8.1.2. There is little theoretical analysis supporting this point of view. To the contrary, the recent work [92] proves that in certain over-parameterized regimes, the classifiers obtained by minimizing the hinge loss and the square loss in fact the same. While the hinge loss is different from cross-entropy, these losses are closely related in certain settings [60, 122]. See [92] for a more in-depth theoretical discussion of loss functions and the related literature.

Probability interpretation of neural network output and calibration.

An argument for using the cross-entropy loss function is sometimes based on the idea that networks trained with cross-entropy are able to output probability of a new data point belonging to a given class. For linear models in the classical analysis of logistic regression, minimizing cross-entropy (logistic loss) indeed yields the maximum likelihood estimator for the model (e.g.,[44], Section 10.5). Yet, the relevance of that analysis to modern highly non-linear and often over-parameterized neural networks is questionable. For example, in [33] the authors state that *"In classification, predictive probabilities obtained at the end of the pipeline (the softmax output) are often erroneously interpreted as model confidence"*. Similarly, the work [137] asserts that *"for DNNs with conventional (also referred as 'vanilla') training to minimize the softmax cross-entropy loss, the outputs do not contain sufficient information for well-calibrated confidence estimation"*. Thus, accurate class probability estimation cannot be considered an unambiguous advantage of neural networks trained with cross-entropy. While the analysis of calibration for different loss functions is beyond the scope of this paper, we note that in many practical settings accurate classification, the primary evaluation metric of this work, takes precedence over the probability estimation.

Domain applicability.

It is interesting to note that in our experiments the square loss generally performs better on NLP and ASR tasks, while cross-entropy has a slight edge on computer vision. It is tempting to infer that the square loss is suitable for NLP and speech, while cross-entropy may be more appropriate for training vision architectures. Yet we are wary of over-interpreting the evidence. In particular, we observe that the cross-entropy has a significant performance advantage on just a single vision architecture (EfficientNet [124] trained on ImageNet). The rest of the vision results are quite similar between square loss and cross-entropy and are likely to be sensitive to the specifics of optimization and parameter tuning. Understanding whether specific loss functions are better suited for certain domain will require more in-depth experimental work.

Related work.

The choice of a loss function is an integral and essential aspect of training neural networks. Yet we are aware of few comparative analyses of loss functions and no other systematic studies of modern architectures across a range of datasets.

[63] compared the effectiveness of squared-error versus cross-entropy in estimating posterior probabilities with small neural networks, five or less nodes in each layer, and argued that cross-entropy had a performance advantage. [39] provided a comparison of cross-entropy and squared error training for a hybrid HMM/neural net model for one ASR and one handwriting recognition datasets. The authors observed that with a good initialization by pre-training, training with the squared error had better performance than the cross-entropy. [118] analyzed the convergence of mean squared error (MSE) and cross-entropy under the normalized logistic regression model (Soft-Max) setting, and indicated the MSE loss function is robust to the true model parameter values and can converge to the same parameter estimation variance of the cross-entropy loss function with half the number of gradient descent iterations. [57] compared

several different loss functions on MNIST and CIFAR-10 datasets concluding that *"depending on the application of the deep model – losses other than log loss* [cross-entropy] *are preferable"*. A recent work [16] provided a theoretical comparison of square and cross-entropy losses for training mixture models. The authors argued that the cross-entropy loss has more favorable optimization landscapes in multiclass settings. To alleviate that issue, they proposed rescaling of the loss function equivalent to choosing parameter k in Section 2.5. The authors showed that rescaling allowed the square loss to become competitive with cross-entropy on CIFAR-100, a finding that aligns with the results in our paper.

2.2 Experiments

We conducted experiments on a number of benchmark datasets for NLP, ASR and computer vision, following the standard recipes given in recent papers of each domain. The NLP datasets are MRPC, SST-2, QNLI, QQP, text-c5, text-c20, text8 and enwik8. TIMIT, WSJ and Librispeech are three standard datasets used for training ASR systems. For vision experiments, we choose MNIST, CIFAR-10 and ImageNet. To the best of our knowledge, we are the first to experimentally compare the square loss and the cross-entropy on a wide range of datasets with different size, dimensionality (number of features) and the number of classes (up to 1000 class numbers). See Appendix A.1 for references and description.

Architectures.

In what follows we explore several widely used modern neural architectures. For NLP tasks, we implement classifiers with a fine-tuned BERT [17], Transformer-XL [14], a LSTM+Attention model [10], and a LSTM+CNN model [45]. Joint CTC-Attention based model [62], triggered attention model with VGG and BLSTM modules [88], and Transformer are used for ASR tasks. Note that for the CTC-Attention based model, the original loss function is a weighted sum of the cross-entropy and the CTC loss. When training with the square loss, we only replace the cross-entropy to be the square loss, and keep the CTC loss untouched. For vision tasks, we use TCNN [3], Wide ResNet [141], Visual transformer [64], ResNet [46] and EfficientNet [124] architectures.

Experimental protocols.

For training with the cross-entropy loss, we use a standard protocol, which is to stop training after the validation accuracy does not improve for five consecutive epochs. For the square loss we use two protocols. The first one is the same as for cross-entropy. The second protocol is to train the square loss using the number of epochs selected when training the cross-entropy loss with the first protocol. The second protocol is designed to equalize the usage of computational resources between the square loss and cross-entropy and is favorable to cross-entropy.

Following the hyper-parameter settings of the architectures in the literature, we re-implement the models trained with the cross-entropy loss keeping the same architecture and hyper-parameter settings. We train the same models using the square loss, employing our two experimental protocols. The only alteration to the parameters of the network reported in the literature is adjustment of the learning rate. For datasets with a large number of labels (42 or more in our experiments) we apply loss function rescaling (see Section 2.5).

The key points for the implementation are described in Section 2.5. The implementation details and specific hyper-parameter settings are given in Appendix A.2. See Appendix A.4 for a summary of comparisons between the original results and our re-implementations. Additionally, we report the results on validation sets and training sets in Appendix A.3.

The results presented below are average results of 5 runs corresponding to 5 different random initalizations for each task. The result across initializations are given in Section 2.3.

2.2.1 NLP experiments

We conduct different classification tasks from NLP domain. The datasets information is summarized in Table 2.1.

As in [127], we report accuracy and F1 scores for MRPC and QQP datasets, and report accuracy for SST-2, QNLI, text-c5 and text-c20. Text8 and enwik8 are classification tasks which classify each text unit into different characters or subwords.

Table 2.1. NLP task statistics and descriptions

Corpus	|Train|	|Test|	#classes	Metric	Domain
MRPC [20]	3.7K	1.7K	2	acc./F1	news
SST-2 [121]	67K	1.8K	2	acc.	movie reviews
QNLI [110]	105K	5.4K	2	acc.	Wikipedia
QQP [55]	364K	391K	2	acc./F1	social QA questions
text-c5	2.1K	0.4K	5	acc.	title of conference
text-c20	28K	12K	20	acc.	stack overflow
text8 [91]	90M	5M	27	acc.	English text
enwik8 [91]	89M	4.9M	204	acc.	English Wikipedia

Table 2.2 gives the accuracy and Table 2.3 gives the F1 scores of the neural models on NLP tasks.

Table 2.2. NLP results, accuracy

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
fine-tuned BERT [17]	MRPC	**83.8**	82.1	83.6
	SST-2	**94.0**	93.9	93.9
	QNLI	**90.6**	**90.6**	**90.6**
	QQP	**88.9**	**88.9**	88.8
	text5	**80.6**	80.5	**80.6**
	text20	**85.6**	85.2	**85.6**
Transformer-XL [14]	text8	**73.2**	72.8	73.2
	enwik8	76.7	**77.5**	76.7
LSTM+Attention [10]	MRPC	**71.7**	70.9	71.5
	QNLI	**79.3**	79.0	**79.3**
	QQP	**83.4**	83.1	**83.4**
LSTM+CNN [45]	MRPC	**73.2**	69.4	72.5
	QNLI	**76.0**	**76.0**	**76.0**
	QQP	84.3	**84.4**	84.3

As can be seen in Table 2.2, in 12 out of 14 tasks using the square loss has better/equal accuracy compared with using the cross-entropy, and in terms of F1 score (see Table 2.3), 5 out of 6 tasks training with the square loss outperform training with the cross-entropy loss. Even with same epochs, i.e. with same computation cost, using the square loss has equal/better accuracy in 11 out of 14 tasks , and has higher F1 score in 5 out of 6 tasks.

Table 2.3. NLP results, F1 scores

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
BERT	MRPC	**88.1**	86.7	88.0
[17]	QQP	**70.9**	70.7	70.7
LSTM+Attention	MRPC	**80.9**	80.6	80.7
[10]	QQP	**62.6**	62.3	**62.6**
LSTM+CNN	MRPC	**81.0**	78.2	**81.0**
[45]	QQP	60.3	**60.5**	60.3

We observe the relative improvements brought by training with the square loss vary with different model architectures, and other than LSTM+CNN model on QQP dataset and Transformer-XL on enwik8, all architectures trained with the square loss have better/equal accuracy and F1 score. The performance of loss functions also varies with data size, especially for MRPC, which is a relatively small dataset, all model architectures trained with the square loss gives significantly better results than the cross-entropy.

2.2.2 Automatic Speech Recognition (ASR) experiments

We consider three datasets, TIMIT, WSJ and Librispeech, and all are ASR tasks. For Librispeech, we choose its train-clean-100 as training set, dev-clean and test-clean as validation and test set. We report phone error rate (PER) and character error rate (CER) for TIMIT, word error rate (WER) and CER for both WSJ and Librispeech. A brief description of the datasets used in our ASR experiments is given in Table 2.4[2]. Note that we only alter the training loss of the acoustic model, while keeping the language model and decoding part the same as described in the literature. The acoustic model is a classifier with the dictionary size as the class number. For TIMIT, getting PER and CER needs two different acoustic models, i.e. they are two separate

[2] We measure the data size in terms of frame numbers, i.e. data samples. As we take frame shift to be 10ms, 1 hour data $\sim$ 360k frames.

Table 2.4. ASR task statistics and descriptions

Corpus	| Train|	| Test|	#classes	Metric	Domain
TIMIT [36]	1.15M	54K	42 27	PER CER	3.2 hours (training set) telephone English
WSJ [102]	28.8M	252K	52*	WER CER	80 hours (training set) read newspapers
Librispeech [98]	36M	1M	1000*	WER CER	100 hours (training set) audio books

* This is the number of classes used for training the acoustic model.

classification tasks, 42-class classification for PER, and 27-class classification for CER. For WSJ, the size of dictionary used for acoustic model is 52. WER and CER of WSJ are calculated with one acoustic model. Hence for WSJ it is a 52-class classification task for both WER and CER. Acoustic model of Librispeech is a 1000-class classifier for both WER and CER, as we use 1000 unigram [61] based dictionary. The results are in Table 2.5.

Table 2.5. ASR results, error rate

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
Attention+CTC	TIMIT (PER)	**20.8**	**20.8**	**20.8**
[62]	TIMIT (CER)	**32.5**	33.4	**32.5**
VGG+BLSTMP	WSJ (WER)	**5.1**	5.3	**5.1**
[88]	WSJ (CER)	**2.4**	2.5	**2.4**
VGG+BLSTM	Librispeech (WER)	**9.8**	10.6	10.3
[88]	Librispeech (CER)	**9.7**	10.7	10.2
Transformer	WSJ (WER)	**5.7**	5.8	**5.7**
[133]	Librispeech (WER)	9.4	**9.2**	9.4

We see that the square loss performs better (equal for TIMIT PER result) in 7 out of 8 tasks. It is interesting to observe that the performance advantage of the square loss reported in Table 2.5 increases with dataset size. In particular, the relative advantage of the square loss (9.3% relative improvement on CER, and 7.5% on WER, respectively) is largest for the biggest dataset, Librispeech with VGG+BLSTM architecture. On WSJ with VGG+BLSTMP architecture, using the square loss has $\sim$4% relative improvement on both CER and WER, while the results on TIMIT for the square loss and cross-entropy are very similar. For Transformer architecture, the square loss slightly outperforms the cross-entropy on WSJ, while the cross-entropy is slightly better on Librispeech. The question of whether this dependence between the data size and the relative advantage of the square loss over cross-entropy is a coincidence or a recurring pattern requires further investigation.

For TIMIT and WSJ, we observed that training with both the square loss and the cross-entropy need same epochs to converge. The two training protocols for training with the square loss have same performance, and both are comparable/better than training with the cross-entropy. On Librispeech, the square loss needs more epochs, but provides better performance.

2.2.3 Computer vision experiments

For vision tasks we conduct experiments on MNIST, CIFAR-10 and ImageNet, as in Table 2.6.

Table 2.6. Vision task statistics and descriptions

Corpus	|Train|	|Test|	#classes	Metric	Domain
MNIST [72]	60K	10K	10	acc.	28×28
CIFAR-10 [68]	50K	10K	10	acc.	32×32
ImageNet [117]	∼1.28M	50K[3]	1000	acc. Top-5 acc.	224×224

As in Table 2.7, on MNIST and CIFAR-10, training with the square loss and the cross-entropy have comparable accuracy. On much larger ImageNet, with ResNet-50 architecture, the accuracy and Top-5 accuracy of using the square loss are comparable with the ones got by using the cross-entropy loss. While with EfficientNet, using the cross-entropy shows better results. The performance of different loss functions varies among different architectures. On MNIST and CIFAR-10, we use exactly the same hyper-parameters well-selected for the cross-entropy loss. For ImageNet, we adjust the learning rate and add a simple rescaling scheme (see Section 2.5), all other hyper-parameters are the same as for the cross-entropy loss. The performance of using the square loss can improve with more hyper-parameter tuning.

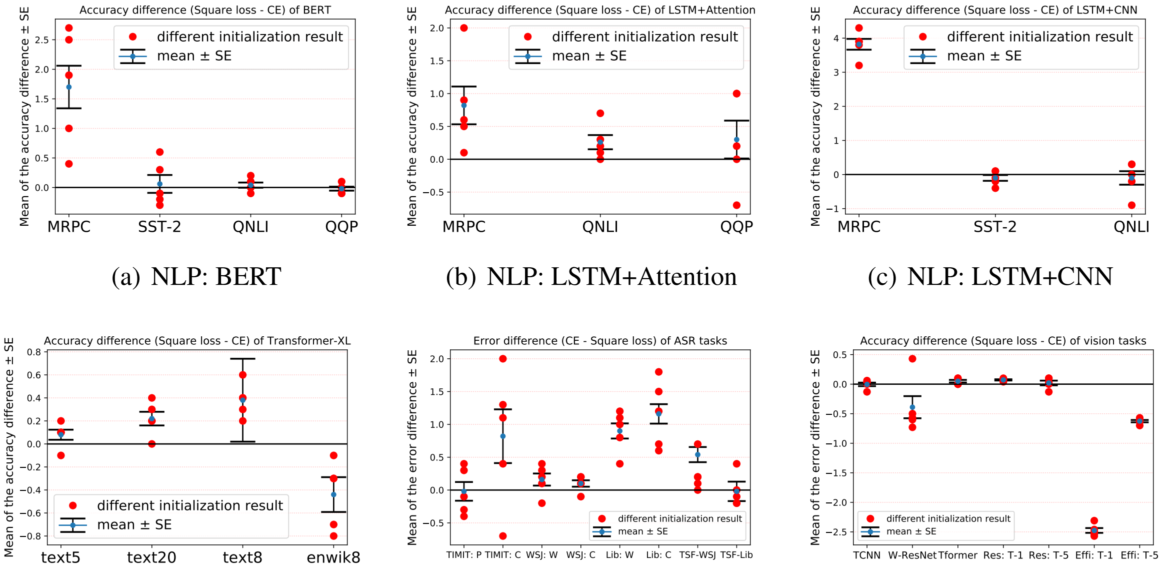
For all three datasets, training with the square loss converges as fast as training with the cross-entropy, and our two experimental protocols for the square loss result in same accuracy

Table 2.7. Vision results, accuracy

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
TCNN [3]	MNIST (acc.)	**97.7**	**97.7**	**97.7**
W-Resnet [141]	CIFAR-10 (acc.)	95.9	**96.3**	95.9
Visual transformer [64]	CIFAR-10 (acc.)	**99.3**	99.2	**99.3**
ResNet-50	ImageNet (acc.)	**76.2**	76.1	76.0
[46]	ImageNet (Top-5 acc.)	**93.0**	**93.0**	92.9
EfficientNet	ImageNet (acc.)	74.6	**77.0**	74.6
[124]	ImageNet (Top-5 acc.)	92.7	**93.3**	92.7

performance (except ImageNet with ResNet-50 model).

2.3 Performance across different initializations



(a) NLP: BERT

(b) NLP: LSTM+Attention

(c) NLP: LSTM+CNN

(d) NLP

(e) ASR: Error rate, P: PER, C: CER, W: WER, TSF: Transformer

(f) Vision: Accuracy

Figure 2.1. Difference between accuracy (or error rate) between square loss and CE for each initialization. (Square loss acc. - CE acc.) is shown for accuracy, (CE - Square loss) for error rate.

To evaluate the stability of the results with respect to the randomness of model initialization we analyze the results for each random seed initialization.

For each random seed, we calculate *the difference* between the the accuracy (or the error) of networks trained with the square loss and the cross-entropy respectively. We present the results with error bars for one standard deviation in Figure 2.1.

Absolute error and accuracy results for each run and the corresponding standard deviations are given in Appendix A.6.

Table 4.2 (Libri is short for Librispeech and I-Net is short for ImageNet) shows the standard deviation of test accuracy/error for training with the square loss and cross-entropy. Square loss has smaller variance in 21 out of 28 tasks, which indicates that training with the square loss is less sensitive to the randomness in the training process.

Table 2.8. Standard deviation of test accuracy/error. Smaller number is bolded.

Model	Dataset	Square loss	CE
BERT	MRPC	**0.484**	0.766
	SST-2	0.279	**0.173**
	QNLI	0.241	**0.205**
	QQP	**0.045**	0.063
	text5	**0.147**	0.167
	text20	0.172	**0.08**
Transformer-XL	text8	**0.174**	0.204
	enwik8	0.228	**0.102**
LSTM +Attention	MRPC	**0.484**	0.786
	QNLI	**0.210**	0.371
	QQP	0.566	**0.352**
LSTM +CNN	MRPC	**0.322**	0.383
	QNLI	**0.173**	0.286
	QQP	0.458	**0.161**
Attention +CTC	TIMIT (PER)	0.508	**0.249**
	TIMIT (CER)	**0.361**	0.873
VGG+ BLSTMP	WSJ (WER)	**0.184**	0.249
	WSJ (CER)	**0.077**	0.118
VGG+ BLSTM	Libri (WER)	**0.126**	0.257
	Libri (CER)	**0.148**	0.316
Transformer	WSJ (WER)	**0.206**	0.276
	Libri (WER)	**0.102**	0.232
TCNN	MNIST	**0.161**	0.173
W-ResNet	CIFAR-10	**0.184**	0.481
Visual Transformer	CIFAR-10	**0.063**	0.075
ResNet-50	I-Net (Top-1)	**0.032**	0.045
	I-Net (Top-5)	0.126	**0.045**
EfficientNet	I-Net (Top-1)	0.138	**0.122**
	I-Net (Top-5)	**0.089**	**0.089**

2.4 Observations during training

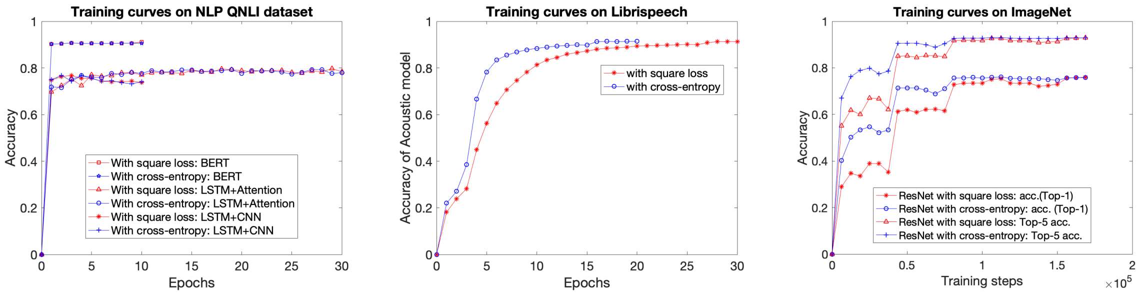
There are several interesting observations in terms of the optimization speed comparing training with the square loss and the cross-entropy loss. We give the experimental observations for the cases when the class number is small, as for our NLP tasks, which are all 2-class classification tasks, and when the class number is relatively large, as for Libripseech and ImageNet (both have 1000 classes).

We compare the convergence speed in terms of accuracy, and find that for 2-class NLP classification tasks, the training curves of training with the square loss and the cross-entropy are quite similar.

Figure 2.2 (a) gives the accuracy of three model architectures trained with the square loss and the cross-entropy along different epochs for QNLI dataset. For all three models, BERT, LSTM+Attention, and LSTM+CNN, using the square loss converges as fast as cross-entropy loss, and achieves better/comparable accuracy to training with the cross-entropy.

Convergence speed when class number is large

When the class number becomes large, as on speech dataset Librispeech and vision dataset ImageNet, training with the square loss may need more epochs to converge. Figure 2.2 (b) gives the classification accuracy of acoustic model along different epochs, and Figure 2.2 (c) gives the accuracy (Top-1) and Top-5 accuracy along different training steps of ResNet on ImageNet. Training with the square loss converges slower but reaches similar/better accuracy.



(a) NLP tasks

(b) ASR tasks

(c) Vision tasks

Figure 2.2. Training curves

2.5 Implementation

We summarize the key points of implementation in this section. Full details and the exact parameters are given in Appendix A.2. Two important pieces of the implementation are (1) no softmax for training with the square loss and (2) loss rescaling for datasets with large number of classes.

No softmax. The widely accepted pipeline for modern neural classification tasks trained with the cross-entropy loss contains the last softmax layer before calculating the loss. When training with the square loss that layer needs to be removed as it appears to impede optimization.

Loss rescaling mechanism. For datasets with a small number of classes, we do not use any additional mechanisms. For datasets with a large number of output classes (≥ 42 in our experiments) we employ loss rescaling which helps to accelerate training. Let $(\boldsymbol{x}, \boldsymbol{y})$ denote a single labeled point, where $\boldsymbol{x} \in \mathbb{R}^d$ is the feature vector, and $\boldsymbol{y} \in \mathbb{R}^C$. Here C is the number of output labels and $\boldsymbol{y} = [0, \ldots, \underbrace{1}_{c}, 0, \ldots, 0]$ is the corresponding one-hot encoding vector of the label c. We denote our model by $f : \mathbb{R}^d \to \mathbb{R}^C$.

The standard square loss for the one-hot encoded label vector can be written (at a single point) as

$$l = \frac{1}{C}\left((f_c(\boldsymbol{x}) - 1)^2 + \sum_{i=1, i \neq c}^{C} f_i(\boldsymbol{x})^2\right) \tag{2.1}$$

Table 2.9. Rescaling parameters

Dataset	#classes	k	M
MRPC	2	1	1
SST-2	2	1	1
QNLI	2	1	1
QQP	2	1	1
text-c5	5	1	1
text-c20	20	1	1
text8	27	1	1
enwik8	204	1	1
TIMIT (CER)	27	1	1
TIMIT (WER)	42	1	15
WSJ	52	1	15
Librispeech	1000	15	30
MNIST	10	1	1
CIFAR-10	10	1	1
ImageNet	1000	15	30

For a large number of classes, we use the rescaled square loss defined by two parameters, k and M, as follows:

$$l_s = \frac{1}{C}\left(k * (f_c(\boldsymbol{x}) - M)^2 + \sum_{i=1, i \neq c}^{C} f_i(\boldsymbol{x})^2\right).$$

The parameter k rescales the loss value at the true label, while M rescales the one-hot encoding (the one-hot vector is multiplied by M). Note that when $k = M = 1$, the rescaled square loss is same as the standard square loss in Eq. 2.1. The values of k and M for all experiments are given in Table 2.9. As in [16], the parameter k is used to increase the emphasis on the correct class in multiclass classification, and this paper proves how adding k can simplify the optimization landscape. We find that for very large class numbers additional parameter M further improves performance.

2.6 Summary and discussion

In this work we provided an empirical comparison of training with the cross-entropy and square loss functions for classification tasks in a range of datasets and architectures. We observe that the square loss outperforms cross-entropy across the majority of datasets and architectures, sometimes by a significant margin. No additional parameter modification except for adjusting the learning rate was necessary for most datasets. For datasets with a large number of classes (42 or more) we used additional loss rescaling to accelerate training. We note that all models used in our experiments were originally designed and tuned for training with the cross-entropy loss. We conjecture that if the neural architectures were selected and tuned for the square loss, performance would be further improved and no extra loss rescaling parameters would be necessary. Another important observation is that the final softmax layer, commonly used with cross-entropy, needs to be removed during training with the square loss.

While we could only explore a small sample of modern models and learning tasks, we believe that the scope of our experiments — ten different neural architectures and ten different datasets across three major application domains — is broad enough to be indicative of the wide spectrum of neural models and datasets. Our empirical results suggest amending best practices of deep learning to include training with square loss for classification problems on equal footing with cross-entropy or even as a preferred option. They also suggest that new theoretical analyses and intuitions need to be developed to understand the important question of training loss function selection.

2.7 Acknowledgements

Chapter 2, in full, is a reprint of Like Hui, and Mikhail Belkin. “Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks.” ICLR 2021. The

dissertation author was the primary investigator and author of this paper.

Chapter 3

Precise asymptotic of rescaled square loss

Multi-class classification is a fundamental problem in modern machine learning, typically involving cross-entropy minimization. Recently, empirical evidence by [51], showed that the *rescaled* square loss is a competitive alternative to the cross-entropy loss. However, it remains unclear how the rescaling hyperparameter R, needs to vary with the number of classes. In this paper, we provide an exact analysis for a 1-layer ReLU network in the proportional asymptotic regime for isotropic Gaussian data. Specifically, we focus on the optimal choice of R as a function of (i) the number of classes, (ii) the degree of overparameterization, and (iii) the level of label noise. Finally, we provide empirical results on real data, which supports our theoretical predictions.

3.1 Introduction

The choice of loss function is important in modern machine learning and for classification tasks in practice, the cross-entropy is usually the default loss function. However, some works [57, 51] compare the square loss with the cross-entropy for classification and conclude that the square loss can be equivalent or even superior to the cross-entropy loss. Specifically, Ryan Rifkin provided empirical evidence supporting this claim on kernel machines [115, 114]. More recently, in [51], by a wide range of experiment on various datasets and modern architectures, the authors show that the square loss achieves comparable or better test performance than the cross entropy. On the other hand, a theoretical line of work prove that the solutions of minimizing the square loss and the cross-entropy can be equivalent under some setting [92, 129]. However, as shown in [51], when class number is large, the square loss needs to do rescaling, and there is no in-depth study, especially, no thorough theoretical analysis on why and how the recaling mechanism helps. As shown in our experiments Figure 3.6, and in [51] the rescaling makes a huge difference to the original square loss and enables square loss achieve even better results compared with the cross-entropy loss. Hence, it is important to understand the correlation between rescaling parameter R and the generalization error, especially, how to set the optimal R. Also, in most theoretical analysis of classification problems, most of the works focus on 2-class classification with the logistic loss, while in practice, a large fraction of the classification learning problems are more than 2 classes. This work focuses on multi-class classification and provides precious asymptotics of 1-layer Relu network trained with rescaled square loss. This enables exact computation of the training loss and generalization error, which are key properties.

Our contribution.

We study the training dynamics and generalization behaviours of training with rescaled square loss in multi-class classification. Empirical evidence in [51] show that rescaled square loss can achieve comparable or even better test performance on a wide range of models and datasets. Here we provide a in-depth study on why and how rescaling helps with the square loss in classifications. In details:

1. In section 3.3.2, we study a 1-layer Relu network which enables theoretical analysis and show that this simplified model (no final linear classification layer) captures the generalization behaviours of scaled square loss found in [51].

2. In section 3.3.4 we give closed-form equations capturing the precious asymptotics of the generalization error and training error, especially the relation of them to rescaling parameter R and class number k. These formulations enable exact computation of those key

quantities. Our proof method relies on recent advances of approximate message passing [28].

3. In section 3.4.1 we investigate the performance of scaled square loss with different *R* and *k* in both over-parameterized setting and under-parameterized settings by giving simulation results on synthetic data.

4. In section 3.4.2, we provide empirical results on real data and observe that the learning curves and generalization behaviour are similar to the ones given by our formulations. That shows our theoretical equations are applicable to real datasets.

3.2 Related work

Multiclass classification.

There is a range of work on analyzing the impact of training loss functions for multiclass classification, including [18, 134, 6, 13, 73] which focus on the algorithms and some of them provide empirical studies on the comparison among different loss functions, see [115, 114] for kernel machines and see [48, 32, 70, 51, 16, 5, 138] for deep neural networks. Many of those work observed that comparing with the cross-entropy loss, training with the square loss achieves comparable generalization performance. Also, another line of research works on analyzing the finite sample behaviour of multiclass classification algorithm in under-parameterized setting [65, 74, 85, 75] or over-parameterized setting [129]. Specifically, Wang et al. [129] study benign overfitting in multiclass linear classification and find that the multiclass support vector machine (SVM) solution, the min-norm interpolating (MNI) solution and the one-vs-all SVM classifier all lead to classifiers that interpolate the training data and have equal accuracy. Frei et. al. [31] consider the generalization error of a two-layer neural network trained to interpolation with logistic loss by gradient descent and show under nonlinearity of the model and training dynamic, this neural networks exhibit benign overfitting. However, they only consider 2-class classification problems. [125] precisely characterize how the test error varies over different training algorithms, data distributions, problem dimensions as well as number of classes, inter/intra class correlations and class priors.

Rescaled loss functions.

[16] compared the cross-entropy and the square loss in multi-class classification and observed that square loss with a scaling parameter to emphasis the loss on the correct class has competitive performance compared with the cross-entropy. [51] conducted a wide range of classification experiments with various modern neural architectures and showed training with the square loss has comparable test performance. However, when the class number is large, the square loss needs to do a heuristical-scaling. [43] were able to reproduce those results and further analyzed the neural collapse of models trained with the square loss. They observed even faster convergence to activation collapse and better robustness than training with the cross-entropy loss. [49] provide theoretical understanding of square loss in classification under over-parameterized NTK regime, and concluded that the square loss has comparable generalization error but noticeable advantages in robustness and model calibration. [145] justify the usage of the rescaled square loss under unconstrained feature model by visualizing the optimization loss landscape around the neural collapse solutions. The authors find that tuning the rescaling hyperparameters can improve the optimization landscape which converges faster to the simplex ETF solutions. As far as we are aware, our work is the first in-depth study on the rescaled square loss for multi-class classification with non-linear model.

Approximate Message Passing.

Our key tool for analyzing the behaviour of training with the rescaled square loss is approximate message passing (AMP). It is originally proposed for the inference of compressed sensing by [21] and some follow-up works develop vector approximate messesage passing (VAMP) algorithm [119] and others prove that this algorithem can be Bayes optimal under certain settings [113]. One recent work by [80] proves exact asymptotics characterising the ERM estimator in high-dimensions for generalized linear model for multi-class classification built on techniques in approximate message-passing framework. They studied the efficiency of L_1 penalty with respect to L_2 and the phase transition on the existence of the multi-class logistic maximum likelihood estimator.

In this work, we apply the VAMP algorithm and adjust it for our 1-layer Relu network for the minimization of the rescaled square loss. The VAMP algorithm involves a set of deterministic recursive equations to compute unknown parameters, which is called state evolution. In the large system limit, i.e. when sample size n and feature dimension d both go to ∞ by a fixed ratio $\beta = n/d$, the VAMP algorithm enables (i) the exact computation of the *precise asymtotics*, such as the generalization error, by getting the fixed points of state evolution, and (ii) the estimates of unknown properties such as parameters, generalization error *in each iteration*. Specifically, the state evolution of VAMP and its fixed point solution help us show the correlation of the rescaling parameter R, class number k and level of label noise to the generalization error.

3.3 Preliminaries

We consider multiclass classification tasks and minimize the rescaled square loss for training. For generalization error, or say test accuracy, the standard 0-1 loss is used. We study the 1-layer Relu network with isotropic Gaussian data. The focus of this work is to provide a rigorous study on how to set the optimal rescaling parameter R and understand the correlation of R, k, label noise level to the generalization behaviour in both over-parameterized ($\beta = n/d < 1$) and under-parameterized ($\beta = n/d > 1$) settings and we call the VAMP algorithm for our 1-layer relu network with rescaled square loss the *1-relu VAMP*.

3.3.1 Data

Let the labeled samples be $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1,\ldots,n}$ and $\boldsymbol{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$ are i.i.d and sampled from a d-dimensional Gaussian distribution with zero mean and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$. $\boldsymbol{y}_i \in \mathbb{R}^k$ is the one-hot encoding of the real-valued label y_i and $y_i = f^*(\boldsymbol{x}_i) + \varepsilon_i$. Specifically, in the following, unless pointed out, we consider the noise-free model by default, and $y_i = \boldsymbol{x_i}\boldsymbol{w}^*$, where $\boldsymbol{w}^* \in \mathbb{R}^{d\times k}$.

The data matrix $\boldsymbol{X} := [\boldsymbol{x}_1 \boldsymbol{x}_2 ... \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$ and label matrix $\boldsymbol{Y} := [\boldsymbol{y}_1 \boldsymbol{y}_2 ... \boldsymbol{y}_n]^T \in \mathbb{R}^{n \times k}$.

Following the setting in [25], we assume that covariance matrix $\boldsymbol{\Sigma}$ has an eigenvalue decomposition,

$$\boldsymbol{\Sigma} = \frac{1}{d} \boldsymbol{V}^T \operatorname{diag}(\boldsymbol{s}_{tr}^2) \boldsymbol{V} \tag{3.1}$$

where $\boldsymbol{s}^2$ are the eigenvalues of $\boldsymbol{\Sigma}$ and $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ is the eigenvectors, which is an orthogonal matrix. With Eq. (3.1), we can write $\boldsymbol{X} = \boldsymbol{U} \operatorname{diag}(\boldsymbol{s}_{tr}) \boldsymbol{V}$, where $\boldsymbol{U} \in \mathbb{R}^{n \times d}$ are sampled i.i.d from $\mathcal{N}(0, \frac{1}{d})$ and its SVD is:

$$\boldsymbol{U} = \boldsymbol{U}_2 \boldsymbol{S} \boldsymbol{U}_1, \ \boldsymbol{S} := \begin{bmatrix} \operatorname{diag}(\boldsymbol{s}) & \boldsymbol{0} \\ \boldsymbol{0} & * \end{bmatrix} \tag{3.2}$$

where $\boldsymbol{U}_1 \in \mathbb{R}^{n \times n}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{d \times d}$ are orthogonal and $\boldsymbol{S} \in \mathbb{R}^{n \times d}$ has non-zero entries $\boldsymbol{s}$ only on the principal diagonal and $\boldsymbol{s} \in \mathbb{R}^{min\{n,d\}}$ are the singular values of $\boldsymbol{U}$. The matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are Harr-distributed on the group of orthogonal matrices.

Test data.

We assume the test data is also Gaussian, and the covariance matrix of test data is:

$$\boldsymbol{\Sigma}_{ts} = \frac{1}{d} \boldsymbol{V}^T \operatorname{diag}(\boldsymbol{s}_{ts}^2) \boldsymbol{V}. \tag{3.3}$$

Then for one test sample $\boldsymbol{x}_{ts}$,

$$\boldsymbol{x}_{ts}^T = \boldsymbol{u} \operatorname{diag}(\boldsymbol{s}_{ts}) \boldsymbol{V}, \tag{3.4}$$

where $\boldsymbol{u} \in \mathbb{R}^d \sim \mathcal{N}(0, 1/d)$. $\boldsymbol{s}_{ts}$ and $\boldsymbol{V}$ are the eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{\Sigma}_{ts}$. Note here following the setting in [25] we assume that the eigenvectors of the training and test samples are the same.

3.3.2 1-layer relu network

In this work we consider a network with only one-hidden layer neural network, that is, the output of relu activation is the prediction of label Y and we directly calculate the loss between the two. Let the parameter of the only layer be $\boldsymbol{w}$, which is in $\mathbb{R}^{p\times k}$, then our model $f:\mathbb{R}^d\to\mathbb{R}^k$ (at a single sample point $\boldsymbol{x}$) is formulated as follows:

$$f(\boldsymbol{w};\boldsymbol{x}) = (\boldsymbol{x}\boldsymbol{w})_+, \tag{3.5}$$

where $(\boldsymbol{x}\boldsymbol{w})_+$ is the output of relu activation function. Note that we do not include a linear classification layer as we find that this one-hidden layer network already capture the optimization and generalization pattern of training with re-scaled square loss. See empirical evidence in Section 3.4. Our 1-layer Relu network enables us to provide the most concise results with VAMP algorithm. Our idea still works for networks with the classification layer and can also extend to general deep networks.

We first consider the scaled square loss as in [51], which has two scaling parameters, and its format is:

$$l_s = \frac{1}{k}\left(M*(f_c(\boldsymbol{w};\boldsymbol{x})-R)^2 + \sum_{j=1,j\neq c}^{k} f_j(\boldsymbol{w};\boldsymbol{x})^2\right), \tag{3.6}$$

where M re-scales the loss at true class c, while R re-scales the one-hot encoding. Our extensive empirical results show that only one parameter R is enough to get similar training dynamic and comparable generalization performance with both M and R. [145] show analysis only with large R is simple and the global optimization conditions remain the same with both M and R. Also, the authors show that with large R leads to a "better" optimization landscape similar to that of the CE loss. Hence, in the following, we consider the scaled square loss with only one scaling parameter R. It remains to solve the following minimization problem:

$$l(\boldsymbol{w}) = \min_{\boldsymbol{w}} \|R\boldsymbol{Y}-(\boldsymbol{X}\boldsymbol{w})_+\|^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{3.7}$$

Note that the nonlinear Relu activation is essential for the rescaling of $\boldsymbol{Y}$ to make a difference, as for linear models such as logistic regression or kernel machines, $\boldsymbol{w}$ is linear to $\boldsymbol{Y}$ and R only change $\boldsymbol{w}$ by R times and does not make a difference for the classification error.

3.3.3 Large system limit

We consider a setting where the sample number n and feature dimension d go to infinity with a fixed ratio β, i.e:

$$\lim_{n\to\infty} \frac{n}{d} = \beta$$

and $\beta \in (0, \infty)$. This large system limit setting is necessary for the analysis of the AMP based algorithms [113]. When $\beta > 1$, it corresponds to under-parameterized setting, while when $\beta < 1$, it corresponds to over-parameterized setting. Our VAMP based analysis of the generalization behaviour caused by the rescaling of the square loss holds for both over-parameterized and under-parameterized settings.

Under the large system limit assumption, the limiting behaviour of the i-th entry of $\boldsymbol{s}$ definied in Eq. (3.2) is

$$\lim_{n\to\infty} \{\boldsymbol{s}_i\} \overset{\text{PL(2)}}{=} s_i \tag{3.8}$$

where $s_i \geq 0$ is a non-negative random variable given by the SVD of matrix $\boldsymbol{U}$ as in Eq. (3.2). It satisfies the Marcencko-Pastur (MP) law and the exact computation (not bound) of the generalization behaviour comes from this MP law which considers all eigenvalues distribution of the matrix.

3.3.4 Main results

Theorem 1. *Consider the learning problem give in Eq. 3.7, which minimize the re-scaled square loss with the 1-layer relu network model. Apply the VAMP algorithm to optimize the parameters w, then there exists constants $\kappa, \tau, \gamma^+, \gamma^-$, such that*

1. *the parameters estimation $\hat{\boldsymbol{w}}$ empirically converges to the solution of proximal operator $\hat{\hat{\boldsymbol{w}}}_i = h(\boldsymbol{Q}, \gamma^-)$, which is*

$$\hat{\hat{\boldsymbol{w}}} = \frac{\boldsymbol{S}\boldsymbol{Q}}{\boldsymbol{S}^2 + \lambda/\gamma^-} \tag{3.9}$$

where $\boldsymbol{Q} = \boldsymbol{J}^* + \mathcal{N}(0, \tau)$. Note that $\kappa, \tau, \gamma^+, \gamma^-$ depends on re-scaling parameter R and *class number k.*

2. *The asymptotic generalization error is a function of k, R, τ, u, and specifically,*

$$\varepsilon_g(k,R,\tau,u) = \mathbb{P}((\boldsymbol{u}\boldsymbol{s}_{ts}\hat{\tilde{\boldsymbol{w}}}_c)_+ < (\boldsymbol{u}\boldsymbol{s}_{ts}\hat{\tilde{\boldsymbol{w}}}_{j\neq c})_+) \tag{3.10}$$

where c is the true label of sample $\boldsymbol{x}_{ts}$.

3.3.5 State evolution of 1-relu VAMP and its fixed point solution

Now we are ready to go to the details of our 1-relu VAMP algorithm for the optimization of Eq. (3.7).

Recursive iterations of state evolution

Let $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{w} \in \mathbb{R}^{n\times k}$, and $\boldsymbol{Q}$, $\boldsymbol{p}$ be the estimates of $\boldsymbol{w}$ and $\boldsymbol{z}$, correspondingly. Then in each iteration of the VAMP algorithm, the functions $h(\cdot)$ and $g(\cdot)$ that produce $\boldsymbol{Q}$ and $\boldsymbol{p}$ are essentially proximal-type operators which are defined as follows:

$$h(\boldsymbol{Q},\boldsymbol{H}^-) = \arg\min_{\boldsymbol{w}} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{Q}\|^2_{\boldsymbol{H}^-} \tag{3.11}$$

$$g(\boldsymbol{p},\boldsymbol{G}^+) = \arg\min_{z} \|R\boldsymbol{Y} - (\boldsymbol{z})_+\|^2 + \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{p}\|^2_{\boldsymbol{G}^+} \tag{3.12}$$

Let $\boldsymbol{G}^+ = \gamma^+\mathbb{I}$ and $\boldsymbol{G}^- = \gamma^-\mathbb{I}$, $\boldsymbol{H}^+ = \eta^+\mathbb{I}$, $\boldsymbol{H}^- = \eta^-\mathbb{I}$, then the VAMP becomes:

$$h(\boldsymbol{Q},\gamma^-) = \arg\min_{\boldsymbol{w}} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\gamma^-}{2}\|\boldsymbol{w}-\boldsymbol{Q}\|^2 \tag{3.13}$$

$$g(\boldsymbol{p},\gamma^+) = \arg\min_{\boldsymbol{z}} \|R\boldsymbol{Y} - (\boldsymbol{z})_+\|^2 + \frac{\gamma^+}{2}\|\boldsymbol{z}-\boldsymbol{p}\|^2 \tag{3.14}$$

Lemma 1. *Let $\boldsymbol{J} = \boldsymbol{S}\boldsymbol{V}^T\boldsymbol{w}$, then $\boldsymbol{Z} = \boldsymbol{V}\boldsymbol{J}, \tilde{\boldsymbol{w}} = \boldsymbol{V}^T\boldsymbol{w} \in \mathbb{R}^{d\times k}$. $h(\boldsymbol{Q},\gamma^-) = \arg\min_{\boldsymbol{J}} \frac{\gamma^-}{2}\|\boldsymbol{J}-\boldsymbol{Q}\|^2 + \frac{\lambda}{2}\|\tilde{\boldsymbol{w}}\|^2$ s.t.$\boldsymbol{J} = \boldsymbol{S}\tilde{\boldsymbol{w}}$, and*

$$\hat{\tilde{w}}_i = \arg\min_{\tilde{w}_i} \frac{\gamma^-}{2}\|s_i\hat{\tilde{w}}_i - Q_i\|^2 + \frac{\lambda}{2}\|\tilde{w}_i\|^2 \; s.t.\hat{J}_i = s_i\hat{\tilde{w}}_i \tag{3.15}$$

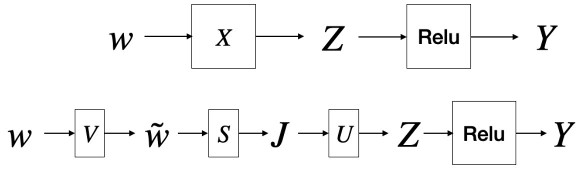


Figure 3.1. Flow of parameter $\boldsymbol{w}$ to model output Y

then, $\hat{\tilde{w}}_i = \frac{\gamma^- s_i Q_i}{\gamma^- s_i^2 + \lambda}, \hat{J}_i = \frac{\gamma^- s_i^2 Q_i}{\gamma^- s_i^2 + \lambda}$.

$\eta^- = \gamma^+ / \frac{\partial g}{\partial p}, \eta^+ = \gamma^- / \frac{\partial h}{\partial Q}$ and

$$\boldsymbol{p} = \boldsymbol{V} \frac{\hat{J}\eta^+ - Q\gamma^-}{\eta^+ - \gamma^-} \tag{3.16}$$

$$\boldsymbol{Q} = \boldsymbol{V}^\top \frac{\hat{Z}\eta^- - p\gamma^+}{\eta^- - \gamma^+} \tag{3.17}$$

This makes $\boldsymbol{p} \approx \boldsymbol{Z}^* + \mathscr{N}(0, \kappa^2)$ and $\boldsymbol{Q} \approx \boldsymbol{J}^* + \mathscr{N}(0, \tau^2)$ and the VAMP algorithm gives the following:

$$\eta^+ = \gamma^- / \frac{\partial h}{\partial Q} \tag{3.18a}$$

$$\gamma^+ = \eta^+ - \gamma^- \tag{3.18b}$$

$$\eta^- = \gamma^+ / \frac{\partial g}{\partial p} \tag{3.18c}$$

$$\gamma^- = \eta^- - \gamma^+ \tag{3.18d}$$

$$\tau^2 = \mathbb{E}\|\frac{\hat{z} - \frac{\partial g}{\partial p} p_i}{1 - \frac{\partial g}{\partial p}} - z_i^*\|^2 \tag{3.18e}$$

$$\kappa^2 = \mathbb{E}\|\frac{\hat{J}_i - \frac{\partial h}{\partial Q} Q_i}{1 - \frac{\partial h}{\partial Q}} - J_i^*\|^2 \tag{3.18f}$$

Let t denote the iterations of VAMP algorithm and $\hat{\boldsymbol{z}}$ be the fix point of $\boldsymbol{z}$. When $t \to \infty$, the solution of $\boldsymbol{z}$ found by the VAMP state evaluation converge to $\hat{\boldsymbol{z}}$.

Lemma 2.

$$\frac{\partial g}{\partial p_{ij}} = \begin{cases} \frac{-\mathit{erf}\left(\frac{-p_\star}{\sqrt{2(\sigma * + \kappa^2)}}\right)}{2k(1+\gamma)} + \frac{2\gamma+1}{2(1+\gamma)} & i = j \\ 0 & i \neq j \end{cases} \tag{3.19}$$

$$\frac{\partial h}{\partial Q_{ij}} = 1 - T(\lambda/\gamma^-) \tag{3.20}$$

The proof is given in Appendix B.2.

Lemma 3. *Let* $u = \lambda/\gamma^-, \tilde{w}_i^* = \mathcal{N}(0, \sigma^2), p_\star = R(1 - \sqrt{1 + 1/\gamma^+})$*, the error function is* $\mathit{erf}(\frac{-p_\star}{\sqrt{2(\sigma *^2 + \kappa^2)}}) := \mathscr{E}_{R,k}(\gamma^+, \kappa)$*, then with fixed known quantities* $\lambda, \sigma_\star^2$ *(variance of* Z^**),* σ^2*,* $\beta = \frac{n}{p}, k, R,$ $\kappa, \tau, \gamma^+, \gamma^-$ *are the solutions of following equations.*

$$\gamma^+ = \frac{\mathscr{T}(u)\lambda/u}{1 - \mathscr{T}(u)} \tag{3.21a}$$

$$\mathscr{T}(u) = \frac{-\mathscr{E}_{R,k}(\gamma^+, \kappa) + k(2\gamma^+ + 1)}{2k(1 + \gamma^+)} \tag{3.21b}$$

$$\kappa^2 = \frac{\tau^2(T_4(u) - (1 - \mathscr{T}(u))^2) + \sigma^2 u^2 \mathscr{T}_2(u)}{\mathscr{T}(u)^2} \tag{3.21c}$$

$$\tau^2 = \frac{\frac{1}{2}\kappa^2(k^2 + k\mathscr{E}_{R,k}(\gamma^+, \kappa) - 2\mathscr{E}_{R,k}(\gamma^+, \kappa))}{(2k\gamma^+ + k - \mathscr{E}_{R,k}(\gamma^+, \kappa))^2} \tag{3.21d}$$

$$+ \frac{2(1 + \mathscr{E}_{R,k}(\gamma^+, \kappa)^2)R^2 + 2(k + \mathscr{E}_{R,k}(\gamma^+, \kappa)^2)\sigma_\star^2}{(2k\gamma^+ + k - \mathscr{E}_{R,k}(\gamma^+, \kappa))^2} \tag{3.21e}$$

where

1. $\mathscr{T}(u) = \frac{-(1-\beta+u) + \sqrt{(1-\beta+u)^2 + 4\beta u}}{2\beta}$

2. $\mathscr{T}_2(u) = \mathscr{T}'(u) = \frac{1}{2\beta}(-1 + \frac{1+\beta+u}{\sqrt{(1-\beta+u)^2 + 4\beta u}})$

3. $\mathscr{T}_4(u) = 1 - T(u) - uT'(u) = \frac{(1+\beta) - \sqrt{(1-\beta+u)^2 + 4\beta u}}{2\beta} - \frac{u(1+\beta+u)}{2\beta\sqrt{(1-\beta+u)^2 + 4\beta u}}$

We give the derivation of Eq. (3.21) in the Appendix B.2.

Lemma 4. *Let $\hat{z}_c$ be the prediction of y_c and $y_c = 1$, where c is the true class index of a sample $\boldsymbol{x}$. Let $\hat{z}$ denote the prediction corresponding to 0 in the one-hot encoding $\mathbf{y}$. Then*

$$\hat{z}_c = \begin{cases} \frac{1}{1+\gamma^+}R + \frac{\gamma^+}{1+\gamma^+}p & p > p_\star \\ p & \textit{otherwise} \end{cases} \tag{3.22}$$

$$\hat{z} = \begin{cases} \frac{\gamma^+}{1+\gamma^+}p & p > 0 \\ p & \textit{otherwise} \end{cases} \tag{3.23}$$

where p is real valued scalar corresponds to the i-th row (sample index) and j-th (label index) column entry of $\boldsymbol{p}$ and $p_\star = R(1-\sqrt{1+1/\gamma^+})$.

See Appendix B.1 for the proof.

3.4 Numerical Results

3.4.1 Simulation results on synthetic data

We generate the training samples and test data as described in Section 3.3.1. For each case of the synthetic data, the test accuracy is the simulation results of Theorem 1. With a given k, R, β, λ, we first get the the solution of τ and u by solving Eq. (3.21) using the fsolve package of Scipy. Then we plug in the value of τ and u into Eq. 3.9 and Eq. 3.10 to get the test accuracy. We find that large R increases the test accuracy by a large scale in most cases. All results are taken as the average of 10 runs with different random seeds.

In the left figure of Figure 3.2 we vary the class number k and fix $n = 1500, d = 7500$ and $\lambda = 10^{-5}$ and get the test accuracy as a function of rescaling parameter R. We see that the test accuracy get large when R increase and becomes flat even with a larger R. This holds for different class number k. In the right figure od Figure 3.2, we fix $k = 60, n = 1500, \lambda = 10^{-5}$ and vary β, i.e. n/d, which measures the sample complexity, essentially is varying feature dimension $d = n/\beta$. We get the test accuracy as a function of R, and observe that in both over-parameterized

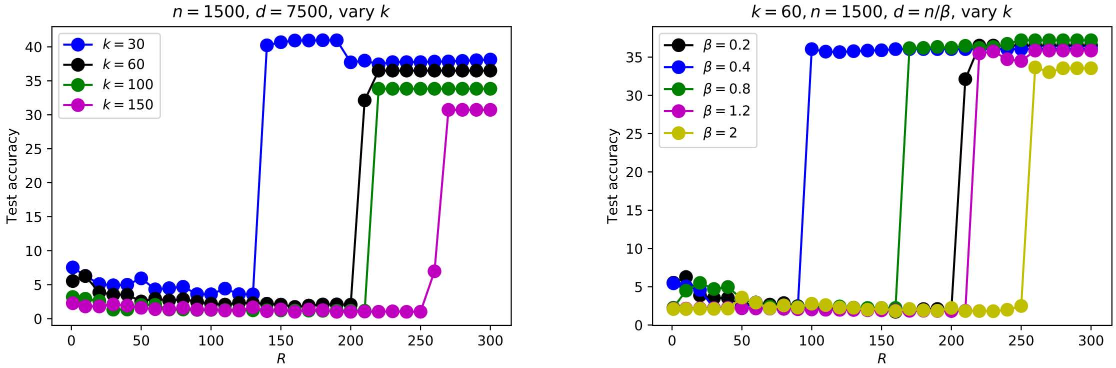


Figure 3.2. *Left:* The test accuracy with different rescaling parameter R for dataset with different class number k. *Right:* The test accuracy with different rescaling parameter R for dataset with different sample complexity $\beta = n/d$.

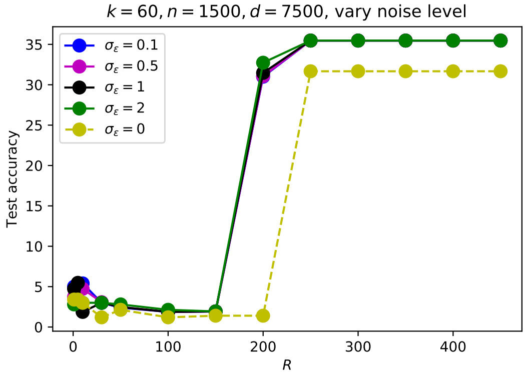


Figure 3.3. The test accuracy with different rescaling parameter R for dataset with different level of noise σ_ε is the variance of the noise. Note that when $\sigma_\varepsilon = 0$ means no label noise.

setting, i.e. $\beta < 1$ case and under-parameterized setting $\beta = 1.2$ and $\beta = 2$, the test accuracy increases with a larger R until some constant. In results of both Figure 3.2, we see that the generalization performance increases by a large scale with an optimal R which is larger than 1.

In Figure 3.3 we show the test accuracy as a function of R under the setting with different level of label noise. That is, for a training sample $\boldsymbol{x}_i$, its label $y_i = \arg\max_j (\boldsymbol{x_i w}^* + \varepsilon_i)$ and ε_i is randomized from a Gaussian distribution $\mathcal{N}(0, \sigma_\varepsilon)$. We observe that increasing R consistently increases the test accuracy by a large margin under different levels of label noise, compared with $R = 1$. Interestingly, the highest test accuracy with an optimal R is larger under label noise case than the one without label noise.

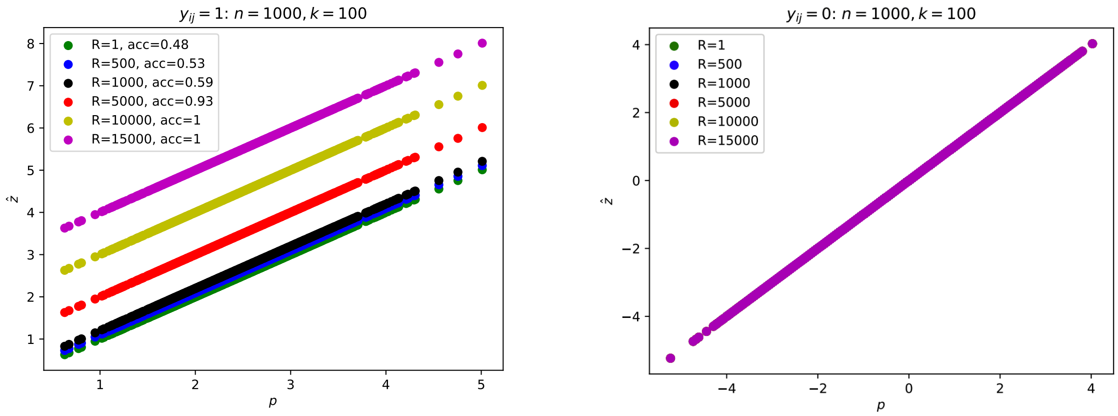


Figure 3.4. The prediction $\hat{z}$ of $y_{ij} = 1$ and $y_{ij} = 0$ for the i-th sample.

To visually see how rescaling helps, we plot $\hat{z}_c$ and $\hat{z}$, which are prediction of corresponding to $y_c = 1$ and $y_i = 0$. In Figure 3.4, the left one plots $\hat{z}_c$ (corresponds to $y_{ij} = 1$) as a function of p, which is the input of the proximal operator in terms of $\boldsymbol{z}$, i.e. $g(\boldsymbol{p}, \gamma^+) = \arg\min_{\boldsymbol{z}} \|R\boldsymbol{Y} - (\boldsymbol{z})_+\|^2 + \frac{\gamma^+}{2}\|\boldsymbol{z} - \boldsymbol{p}\|^2$. The right one of Figure 3.4 gives the average of all $\hat{z}_{j \neq c}$ (corresponds to $y_{ij} = 0$). Here we only consider a single sample, hence $\hat{z}_j$ and the corresponding p is a scalar. We see that $\hat{z}$ is linear with p, that is, $\hat{z}_j = p + b_j$ for all $j = 1, ..., k$, while with a larger R, b_c increases while $b_{j \neq c}$ stays to be 0. That makes $\hat{z}_c$ win and increase the accuracy.

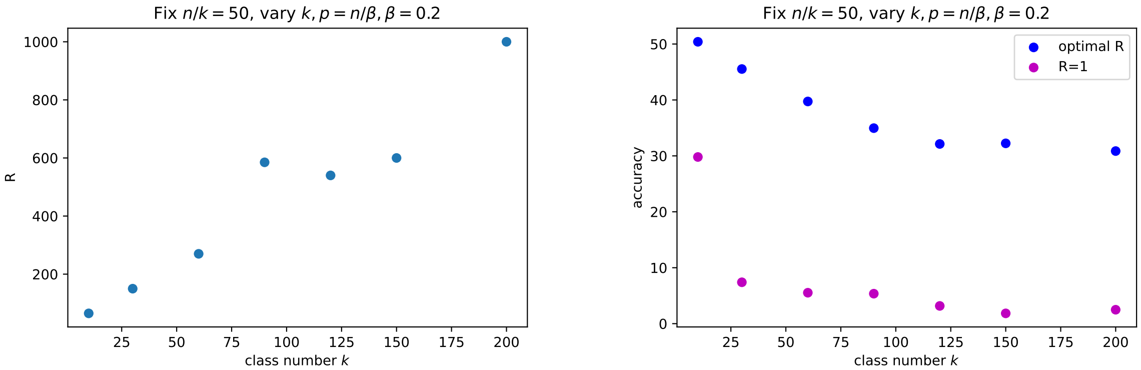


Figure 3.5. Optimal R for different class number k

In Figure 3.5, we fix $n/k = 50, \beta = 0.2$ and vary class number k. We show the optimal R for different class number k, and also give the accuracy of the optimal R and when $R = 1$. We can see that mostly $R > 1$ and the optimal R gives much better test accuracy for all class number cases.

3.4.2 Results on real data

In this section, we show how rescaling of the square loss helps increasing the test performance on real datasets. Inspired by the empirical results in [51] which show the (rescaled) square loss are competitive to the cross-entropy loss in training modern networks, we do experiments on real data and see that rescaling mechnism improves the original square loss by a large scale.

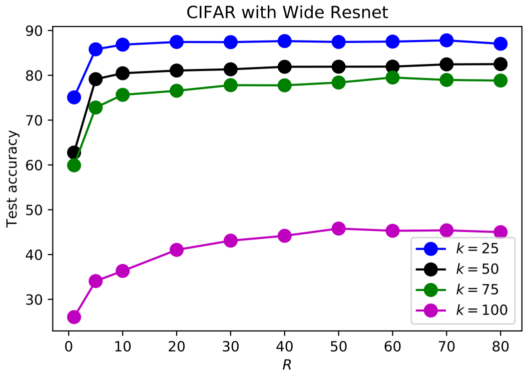


Figure 3.6. The test accuracy of CIFAR-100. Note that for $k = 100$ case, we choose a subset with 5000 training samples to speed up.

We do experiments on CIFAR-100, and we randomly select a subset from the full CIFAR-100 to get different class number k. The results are shown in Figure 3.6, the test accuracy has more than 15% improvement in all cases with different class numbers. Our simulation results in Section 3.4.1 captures the correlation of test accuracy and rescaling parameter R observed in real dataset. As this work aims at providing a theoretical framework to understand the rescaling mechanism of the square loss, we do not provide a lot of empirical results on real data.

3.5 Acknowledgements

Chapter 3, in full, is a preprint of Like Hui, Parthe Pandit, Mikhail Belkin, “Precise asymptotics of Rescaled square loss for Multiclass Classification”. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Cut your Losses with Squentropy

Nearly all practical neural models for classification are trained using cross-entropy loss. Yet this ubiquitous choice is supported by little theoretical or empirical evidence. Recent work [51] suggests that training using the (rescaled) square loss is often superior in terms of the classification accuracy. In this paper we propose the "squentropy" loss, which is the sum of two terms: the cross-entropy loss and the average square loss over the incorrect classes. We provide an extensive set of experiments on multi-class classification problems showing that the squentropy loss outperforms both the pure cross entropy and rescaled square losses in terms of the classification accuracy. We also demonstrate that it provides significantly better model calibration than either of these alternative losses and, furthermore, has less variance with respect to the random initialization. Additionally, in contrast to the square loss, squentropy loss can typically be trained using exactly the same optimization parameters, including the learning rate, as the standard cross-entropy loss, making it a true "plug-and-play" replacement. Finally, unlike the rescaled square loss, multiclass squentropy contains no parameters that need to be adjusted.

4.1 Introduction

As with the choice of an optimization algorithm, the choice of loss function is an indispensable ingredient in training neural network models. Yet, while there is extensive theoretical and empirical research into optimization and regularization methods for training deep neural networks [123], far less is known about the selection of loss functions. In recent years, cross-entropy loss has been predominant in training for multi-class classification with modern neural architectures. There is surprisingly little theoretical or empirical evidence in support of this choice. To the contrary, an extensive set of experiments with neural architectures conducted in [51] indicated that training with the (rescaled) square loss produces similar or better classification accuracy than cross entropy on most classification tasks. Still, the rescaled square loss proposed in that work requires additional parameters (which must be tuned) when the number of classes is large. Further, the optimization learning rate for the square loss is typically different from that of cross entropy, which precludes the use of square loss as an out-of-the-box replacement.

In this work we propose the "squentropy" loss function for multi-class classification. Squentropy is the sum of two terms: the standard cross-entropy loss and the average square loss over the incorrect classes. Unlike the rescaled square loss, squentropy has no adjustable parameters. Moreover, in most cases, we can simply use the optimal hyperparameters for cross-entropy loss without any additional tuning, making it a true "plug-and-play" replacement for cross-entropy loss.

To show the effectiveness of squentropy, we provide comprehensive experimental results over a broad range of benchmarks with different neural architectures and data from NLP, speech, and computer vision. In 24 out of 34 tasks, squentropy has the best (or tied for best) classification accuracy, in comparison with cross entropy and the rescaled square loss. Furthermore, squentropy has consistently improved *calibration*, an important measure of how the output values of the neural network match the underlying probability of the labels [41]. Specifically, in 26 out of 32 tasks for which calibration results can be computed, squentropy is better calibrated than either alternative. We also show results on 121 tabular datasets from [30]. Compared with cross entropy, squentropy has better test accuracy on 94 out of 121 tasks, and better calibration on 83 datasets. Finally, we show that squentropy is less sensitive to the randomness of the initialization than either of the two alternative losses.

Our empirical evidence suggests that in most settings, squentropy should be the first choice of loss function for multi-class classification via neural networks.

4.2 The squentropy loss function

The problem we consider here is supervised multi-class classification. We focus on the loss functions for training neural classifiers on this task.

Let $D = (\boldsymbol{x}_i, y_i)_{i=1}^n$ denote the dataset sampled from a joint distribution $\mathscr{D}(\mathscr{X}, \mathscr{Y})$. For each sample i, $\boldsymbol{x}_i \in \mathscr{X}$ is the input and $y_i \in \mathscr{Y} = \{1, 2, \ldots, C\}$ is the true class label. The one-hot encoding label used for training is $\boldsymbol{e}_{y_i} = [0, \ldots, \underbrace{1}_{y_i}, 0, \ldots, 0]^T \in \mathbb{R}^C$. Let $f(\boldsymbol{x}_i) \in \mathbb{R}^C$ denote the logits (output of last linear layer) of a neural network of input $\boldsymbol{x}_i$, with components $f_j(\boldsymbol{x}_i)$, $j = 1, 2, \ldots, C$. Let $p_{i,j} = e^{f_j(\boldsymbol{x}_i)} / \sum_{j=1}^C e^{f_j(\boldsymbol{x}_i)}$ denote the predicted probability of $\boldsymbol{x}_i$ to be in class j. Then the squentropy loss function on a single sample $\boldsymbol{x}_i$ is defined as follows:

$$l_{\text{sqen}}(\boldsymbol{x}_i, y_i) = -\log p_{i,y_i}(\boldsymbol{x}_i) + \frac{1}{C-1} \sum_{j=1, j \neq y_i}^{C} f_j(\boldsymbol{x}_i)^2. \tag{4.1}$$

The first term $-\log p_{i,y_i}(\boldsymbol{x}_i)$ is simply cross-entropy loss. The second term is the square loss averaged over the incorrect ($j \neq y_i$) classes.

The cross-entropy loss is minimized when $f_{y_i}(\boldsymbol{x}_i) \to \infty$ while $f_j(\boldsymbol{x}_i) \to -\infty$ or at least stays finite for $j \neq y_i$. By encouraging all incorrect logits to go to a specific point, namely 0, it is possible that squentropy yields a more “stable” set of logits — the potential for the incorrect logits to behave chaotically is taken away. In other words, the square loss term plays the role of a regularizer. We discuss this point further in Section 4.4.2.

Dissecting squentropy.

Cross entropy acts as an effective penalty on the prediction error made for the true class y_i, as it has high loss and large gradient when p_{i,y_i} is close to zero, leading to effective steps in a gradient-based optimization scheme. The “signal” coming from the gradient for the incorrect classes is weaker, so such optimization schemes may be less effective in driving the probabilities for these classes to zero. Squentropy can be viewed as a modification of the rescaled square loss [51], in which cross entropy replaces the term $t(f_{y_i}(\boldsymbol{x}_i) - M)^2$ corresponding to the true class, which depends on two parameters t, M that must be tuned. This use of cross entropy dispenses with the additional parameters yet provides an adequate "signal" for the gradient for a term that captures loss on the "true" class.

The second term in (4.1) pushes all logits $f_j(\boldsymbol{x}_i)$ corresponds to false classes $j \neq y_i$ to 0. Cross entropy attains a loss close to zero on term i by sending $f_{y_i}(\boldsymbol{x}_i) \to \infty$ and/or $f_j(\boldsymbol{x}_i) \to -\infty$ for all $j \neq y_i$. By contrast, squentropy "anchors" the incorrect logits at zero (via the second term) while driving $f_{y_i}(\boldsymbol{x}_i) \to \infty$ (via the first term). Then the predicted probability of true class $p_{i,y_i}(\boldsymbol{x}_i)$ will be close to $\frac{e^{f_{y_i}(\boldsymbol{x}_i)}}{e^{f_{y_i}(\boldsymbol{x}_i)}+C-1}$ for squentropy, which possibly approaches 1 more slowly than for cross entropy. When the training process is terminated, the probabilities $p_{i,y_i}(\boldsymbol{x}_i)$ tend to be less clustered near 1 for squentropy than for cross entropy. Confidence in the true class thus tends to be slightly lower in squentropy. We see the same tendency toward lower confidence in the *test* data, thus helping calibration.

In calibration literature, various post-processing methods, such as Platt scaling [104] and temperature scaling [41], also improves calibration by reducing p_{i,y_i} below 1, while other methods such as label smoothing [90, 79] and focal loss [89] achieve similar reduction on the predicted probability. While all these methods require additional hyperparameters, squentropy does not. We conjecture that calibration of squentropy can be further improved by combining it with these techniques.

Relationship to neural collapse.

Another line of work that motivates our choice of loss function is the concept of neural collapse [100]. Results and observations for neural collapse interpose a linear transformation between the outputs of the network (the transformed features $f_j(\boldsymbol{x}_i)$) and the loss function. They show broadly that the features collapse to a class average and that, under a cross-entropy loss, the final linear transformation maps them to rays that point in the direction of the corners of the simplex in $\mathbb{R}^C$. (A modified version of this claim is proved for square loss in [43].) Our model is missing the interposing linear transformation, but these observations suggest roughly that cross entropy should drive the true logits $f_{y_i}(\boldsymbol{x}_i)$ to ∞ while the incorrect logits $f_j(\boldsymbol{x}_i)$ for $j \neq y_i$ tend to drift toward $-\infty$, as discussed above. As noted earlier, the square loss term in our squentropy loss function encourages $f_j(\boldsymbol{x}_i)$ for $j \neq y_i$ to be driven to zero instead — a more well defined limit and one that may be achieved without blowing up the weights in the neural network (or by increasing them at a slower rate). In this sense, as mentioned above, the squared loss term is a kind of regularizer.

Confidence calibration.

We use the expected calibration error (ECE) [93] to evaluate confidence calibration performance. It is defined as $\mathbb{E}_p[|\mathbb{P}(\hat{y} = y|p) - p|]$, where p and y correspond to the estimated probability and true label of a test sample $\boldsymbol{x}$. $\hat{y}$ is the predicted label given by $\underset{j}{\operatorname{argmax}}\, p_j$. It captures the expected difference between the accuracy $\mathbb{P}(\hat{y} = y|p)$ and the estimated model confidence p.

Because we only have finite samples in practice, and because we do not have access to the true confidences p_{true} for the test set (only the labels y), we need to replace this definition with an *approximate* ECE. This quantity is calculated by dividing the interval $[0,1]$ of probability predictions into K equally-spaced bins with the k-th bin interval to be $(\frac{k-1}{K}, \frac{k}{K}]$. Let B_k denote the set of test samples $(\boldsymbol{x}_i, \hat{y}_i)$ for which the confidence p_{i,y_i} predicted by the model lies in bin k. (The probabilities $p_{i,j}$ are obtained from a softmax on the exponentials of the logits $f_j(\boldsymbol{x}_i)$.) The accuracy of this bin is defined to be $\operatorname{acc}(B_k) = \frac{1}{|B_k|}\sum_{i \in B_k} \mathbf{1}(\hat{y}_i = y_i)$, where y_i is the true label for the test sample $\boldsymbol{x}_i$ and $\hat{y}_i$ is the model prediction for this item (the one for which $p_{i,j}$ are maximized over $j = 1, 2, \ldots, C$). The confidence for bin k is defined empirically as

$\mathrm{conf}(B_k) = \frac{1}{|B_k|}\sum_{i \in B_k} p_{i,y_i}$. We then use the following definition of ECE:

$$\mathrm{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{n} \left|\mathrm{acc}(B_k) - \mathrm{conf}(B_k)\right|. \tag{4.2}$$

This quantity is small when the frequency of correct classification over the test set matches the probability of the predicted label.

4.3 Experiments

In this paper we consider three loss functions, our proposed squentropy, cross entropy and the (rescaled) square loss from [51]. The latter is formulated as follows:

$$l_s(\boldsymbol{x}_i, y_i) = \frac{1}{C}\left(t * (f_{y_i}(\boldsymbol{x}_i) - M)^2 + \sum_{j=1, j \neq y_i}^{C} f_j(\boldsymbol{x}_i)^2\right), \tag{4.3}$$

where t and M are positive parameters. ($t = M = 1$ yields standard square loss.) We will point out those entries in which values $t > 1$ or $M > 1$ were used; for the others, we set $t = M = 1$. Note that following [51], the square loss is directly applied to the logits, with no softmax layer in training.

We conduct extensive experiments on various datasets. These include a wide range of well-known benchmarks across NLP, speech, and vision with different neural architectures — more than 30 tasks altogether. In addition, we evaluate the loss functions on 121 tabular datasets [30]. In the majority of our experiments, training with squentropy gives best test performance and also consistently better calibration results.

Training scheme.

In most of experiments we train with squentropy with hyperparameter settings that are optimal for cross entropy, given in [51]. This choice favors cross entropy. This choice also means that switching to squentropy requires a change of just one line of code. Additional gains in performance of squentropy might result from additional tuning, at the cost of more computation in the hyperparameter tuning process.

Datasets.

We test on a wide range of well-known benchmarks from NLP, speech and computer vision. NLP datasets include MRPC, SST-2, QNLI, QQP, text8, enwik8, text5, and text20. Speech datasets include TIMIT, WSJ, and Librispeech. MNIST, CIFAR-10, STL-10 [12], CIFAR-100, SVHN [96], and ImageNet are vision tasks. See Appendix A of [51] for details of most of those datasets. (The exceptions are SVHN, STL-10, and CIFAR-100, which we describe in Appendix C.1 of this paper). The 121 tabular datasets are from [30] and they are mostly small datasets — 90 of them have ≤ 5000 samples. The feature dimension is small (mostly < 50) and most datasets are class-imbalanced.

Architectures and hyperparameter settings.

We choose various modern neural architectures, including simple fully-connected networks, convolutional networks (TCNN[3], Resnet-18, VGG, Resnet-50 [46], EfficientNet[124]), LSTM-based networks [11] (LSTM+CNN, LSTM+Attention, BLSTM), and Transformers [126] (fine-tuned BERT, Transformer-XL, Transformer, Visual transformer). See Table 4.1 for detailed references. We follow the hyperparameter settings given in Appendix C.2 of [51] for the cross-entropy loss and the square loss (other than SVHN, STL-10, and CIFAR-100), and use the algorithmic parameters settings of the cross entropy for squentropy in most cases. The exceptions are SVHN and STL-10, where squentropy and square loss have a smaller learning rate (0.1 for cross entropy while 0.02 for squentropy and square loss). More details about hyperparameter settings of SVHN, STL-10, CIFAR-100 are in Appendix C.2.

Metrics.

For NLP, vision and 121 tabular datasets, we report accuracy as the metric for test performance. For speech dataset, we conduct the automatic speech recognition (ASR) tasks and report test set error rates which are standard metrics for ASR. Precisely, for TIMIT, we report phone error rate (PER) and character error rate (CER). For WSJ and Librispeech, we report CER and word error rate (WER). ECE is the metric to measure the calibration results for all datasets. For speech datasets, we report calibration results for the acoustic modeling part. See Table 4.1 shows the results of NLP, speech and vision datasets. Figure 4.2 show results of 121 tabular datasets. In addition, we provide reliability diagrams [15, 97] to visualize the confidence and accuracy of each interval and see details in Section 4.3.2.

Remarks on Table 4.1.

For the results of square loss, we use rescaled square loss with $t > 1$ or $M > 1$ for TIMIT(PER) ($t = 1, M = 15$), WSJ ($t = 1, M = 15$), Librispeech ($t = 15, M = 30$), CIFAR-10 and CIFAR-100 ($t = 1, M = 10$), and ImageNet ($t = 15, M = 30$). All others are the standard square loss. Note that WSJ (WER) and WSJ (CER) share the same ECE number as they share one acoustic model. (Similarly for Librispeech.) Additionally, since ECE numbers are not available for Top-5 accuracy, the corresponding entries (ImageNet, Top-5 acc.) are marked as "N/A".

For the empirical results reported in Table 4.1, we discuss generalization / test performance in Section 4.3.1 and calibration results in Section 4.3.2. Results for 121 tabular datasets are reported in Section 4.3.3. We report the *average* accuracy/error rate (for test performance) and *average* ECE (for model calibration) of *5 runs* with different random initializations for all experiments. We report the standard derivation of this collection of runs in Section 4.3.4.

4.3.1 Empirical results on test performance

See Table 4.1 for its test accuracy. In Figure 4.1, the Confidence histogram gives the portion of samples in each confidence interval, and the reliability diagrams show the accuracy as a function of the confidence. The ECE numbers are percentages as in Table 4.1. In the left of Figure 4.1 is for Squentropy, and in the middle left is cross entropy. While in the middle right is for rescaled square loss, and in the right of Figure 4.1 is standard square loss. We see that models trained with squentropy are better calibrated, while cross entropy suffers from overconfidence

Table 4.1. Test performance (perf(%): accuracy for NLP&Vision, error rate for speech data) and calibration: ECE(%).

Domain	Model	Task	Squentropy		Cross-entropy		Square loss	
			perf	ECE	perf	ECE	perf	ECE
NLP	fine-tuned BERT [17]	MRPC	**84.0**	**7.9**	82.1	13.1	83.8	14.0
		SST-2	**94.2**	7.0	93.9	**6.7**	94.0	19.8
		QNLI	**91.0**	7.3	90.6	7.4	90.6	**4.2**
		QQP	**89.0**	**2.2**	88.9	5.8	88.9	2.8
		text5	**85.2**	**12.4**	84.5	14.9	84.6	46.7
		text20	**81.2**	**10.5**	80.8	16.2	80.8	69.2
	Transformer-XL [14]	text8	71.5	**3.9**	**72.8**	5.8	73.2	57.6
		enwik8	77.0	**4.8**	**77.5**	9.3	76.7	64.5
		enwik8 (subset)	**48.9**	**10.7**	48.6	18.9	47.3	70.6
	LSTM+Attention [11]	MRPC	71.4	**3.2**	70.9	7.1	**71.7**	3.5
		QNLI	**79.3**	**7.2**	79.0	7.6	**79.3**	13.0
		QQP	**83.5**	**2.4**	83.1	3.2	**83.4**	16.5
	LSTM+CNN [45]	MRPC	70.5	**5.2**	69.4	6.3	**73.2**	16.3
		QNLI	**76.0**	4.1	**76.0**	**2.3**	**76.0**	20.5
		QQP	**84.5**	**5.1**	84.4	7.2	84.3	24.6
Speech	Attention+CTC [62]	TIMIT (PER)	**19.6**	**0.7**	20.0	3.1	20.0	2.8
		TIMIT (CER)	**32.1**	**1.6**	33.4	3.3	32.5	4.3
	VGG+BLSTMP [88]	WSJ (WER)	5.5	**3.2**	5.3	5.0	**5.1**	5.3
		WSJ (CER)	2.9	**3.2**	2.5	5.0	**2.4**	5.3
	VGG+BLSTM [88]	Librispeech (WER)	**7.6**	7.1	8.2	**2.7**	8.0	7.9
		Librispeech (CER)	**9.7**	7.1	10.6	**2.7**	**9.7**	7.9
	Transformer [133]	WSJ (WER)	**3.9**	**2.1**	4.2	4.3	4.0	4.4
		Librispeech (WER)	**9.1**	**4.2**	9.2	4.9	9.4	5.1
Vision	TCNN [3]	MNIST	**97.8**	**1.4**	97.7	1.6	97.7	75.0
	Resnet-18 [46]	CIFAR-10	**85.5**	**8.9**	84.7	10.0	84.6	13.4
		STL-10	67.7	**21.2**	**68.9**	26.1	65.4	40.3
	W-Resnet [141]	CIFAR-100	**77.5**	**10.9**	76.7	17.9	76.5	12.7
		CIFAR-100 (subset)	**43.5**	**18.8**	41.5	40.3	41.0	23.8
	Visual transformer	CIFAR-10	**99.3**	**1.9**	99.2	3.8	**99.3**	7.2
	VGG	SVHN	93.0	**4.8**	**93.7**	5.7	92.5	65.4
	Resnet-50 [46]	ImageNet (acc.)	**76.3**	**6.3**	76.1	6.7	76.2	8.2
		ImageNet (Top-5 acc.)	**93.2**	N/A	93.0	N/A	93.0	N/A
	EfficientNet [124]	ImageNet (acc.)	76.4	6.8	**77.0**	**5.6**	74.6	7.9
		ImageNet (Top-5 acc.)	93.0	N/A	**93.3**	N/A	92.7	N/A

and the standard square loss is highly underconfident.

Our results show that squentropy has better test performance than cross entropy and square loss in the majority of our experiments. The perf(%) numbers in Table 4.1 show the test accuracy of benchmarks of the NLP and vision tasks, and error rate for the speech tasks.

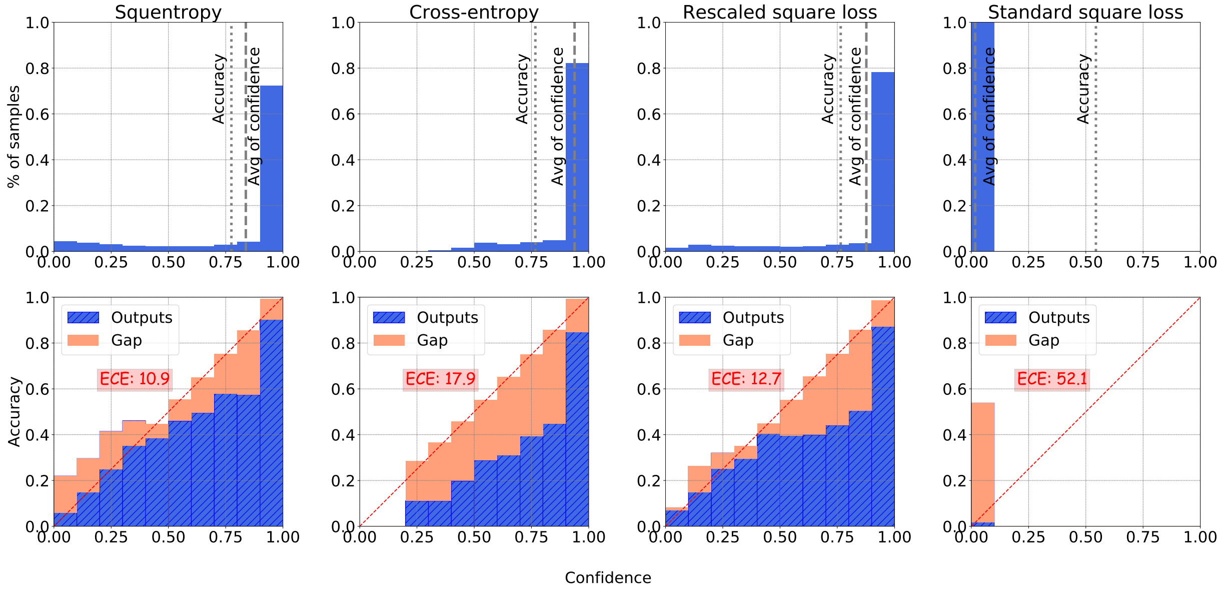


Figure 4.1. Confidence histograms (top) and reliability diagrams (bottom) for a Wide Resnet on CIFAR-100.

Squentropy behaves the best in *24 out of 34* tasks. We also report the numbers for *subsets* of enwik8 and CIFAR-100. Compared with full datasets of these collections, squentropy seems to gain more when the datasets are small.

Applicability and significance.

Table 4.1 shows improvements for squentropy across a wide range distributions from the NLP, speech, and vision domains. On the other hand, the improvement on one single task often is not significant, and for some datasets, squentropy's performance is worse. One reason may be our choice to use the optimal hyperparameter values for cross entropy in squentropy. Further tuning of these hyperparameters may yield significant improvements.

4.3.2 Empirical results on calibration

In this section we show model calibration results, measured with ECE of the models given in Table 4.1. The ECE numbers for NLP, speech, and vision tasks are also shown in Table 4.1.

Squentropy consistently improves calibration.

As can be seen in and Table 4.1, in 26 out of 32 tasks, the calibration error (ECE) of models trained with squentropy is smaller than for cross entropy and square loss, even in those cases in which squentropy had slightly worse test performance, such as WSJ, STL10, and SVHN.

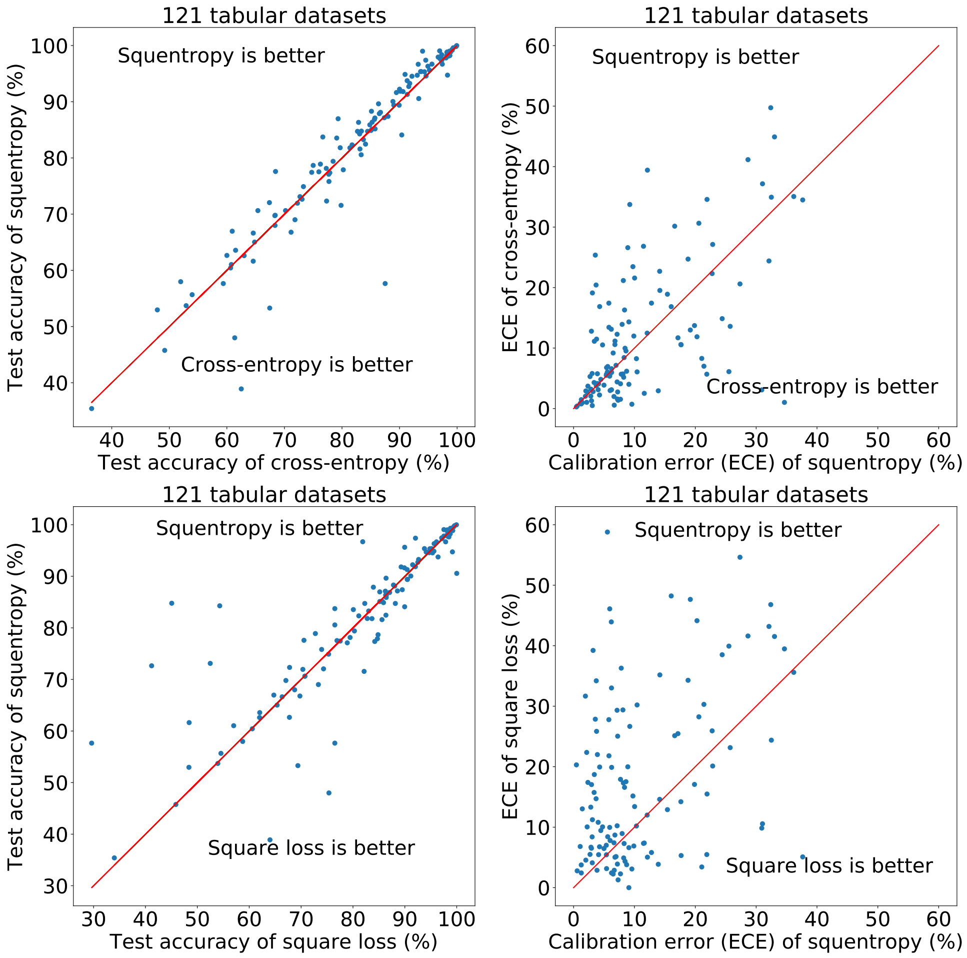
Besides using ECE to measure model calibration, we also provide a popular form of visual representation of model calibration: reliability diagrams [15, 97], which show accuracy as a function of confidence as a bar chart. If the model is perfectly calibrated, i.e. $\mathbb{P}(\hat{y}_i = y_i | p_i) = p_i$, the diagram should show all bars aligning with the identity function. If most of the bars lie below the identity function, the model is overconfident as the confidence is mostly larger than corresponding accuracy. When most bars lie above the identity function that means the model is underconfident as confidence is smaller than accuracy. For a given bin k, the difference between $\mathrm{acc}(B_k)$ and $\mathrm{conf}(B_k)$ represents the calibration *gap* (orange bars in reliability diagrams – e.g. the bottom row of Figure 4.1).

In Figure 4.1 we plot the confidence histogram (top) and the reliability diagrams (bottom) of Wide Resnets on CIFAR-100, trained with four different loss functions: squentropy, cross entropy, rescaled square loss (with $t = 1, M = 10$), and standard square loss ($t = 1, M = 1$). The confidence histogram gives the percentage of samples in each confidence interval, while the reliability diagrams show the test accuracy as a function of confidence.

In the reliability diagrams of Figure 4.1 bottom, the orange bars, which represent the confidence *gap*, start from the top of the blue (accuracy) bar. We show $\mathrm{conf}(B_k) - \mathrm{acc}(B_k)$ for all intervals in all reliability diagram plots. Note that for intervals where confidence is smaller than accuracy, the orange bars go down from the top of the blue bars, such as the one in the right bottom of Figure 4.1. More reliability diagrams for other tasks are given in Appendix C.3.

Squentropy vs. cross entropy.

If we compare the diagrams of squentropy and cross entropy, the bars for squentropy are closer to the identity function; cross entropy apparently yields more overconfident models. The gap for squentropy is also smaller than cross entropy in most confidence intervals.



Loss functions	Squentropy	Cross entropy	Square loss
Avg accuracy	**82.5%**	82.1%	81.0%
Avg ECE	**10.9%**	11.8%	17.5%

Figure 4.2. Test accuracy and model calibration of 121 tabular datasets from [30] trained with a 3 layer (64-128-64) fully connected network. The results for each dataset are averaged over 5 runs with different random initializations.

Standard square loss leads to underconfidence.

We also plot the reliability diagrams for training with the standard square loss on the right ones of Figure 4.1. We see that it is highly underconfident as the confidence is smaller than 0.1 (exact number is 0.017) for all samples. Note that the square loss is directly applied to the logits $f_j(x_i)$, and the logits are driven to the one-hot vector $\boldsymbol{e}_{y_i}$, then the *probabilities* $p_{i,j}$ formed from these logits are not going to be close to the one-hot vector. The "max" probability (confidence) will instead be close to $\frac{e}{e+(C-1)}$, which is small when C is large.

Rescaling helps with calibration.

The second-from-right bottom diagram in Figure 4.1 shows the results of training with the rescaled square loss ($t = 1, M = 10$) on CIFAR-100. This minimization problem drives the logits of true class closer to M, making the max probability approach $\frac{e^M}{e^M+(C-1)}$ - a much larger value than for standard square loss, leading to better calibration. However, squentropy can avoid extra rescaling hyperparameters while achieving even smaller values of ECE.

4.3.3 Additional results on 121 Tabular datasets

Additional results for 121 small, low dimensional, and class-imbalanced tabular dataset, obtained with 3-layer fully-connected networks, are shown in Figure 4.2. For all these cases, we use SGD optimizer with weight decay parameter $5 * 10^{-4}$ and run 400 epochs with learning rate 0.01. The “square loss” function used here is in fact rescaled version with parameters $t = 1$ and $M = 5$.

Figure 4.2 shows that for most datasets, in the left of Figure 4.2 shows test accuracy and larger is better. In the right of Figure 4.2 shows the calibration error ECE and smaller is better. The top figures plot the results of squentropy and cross entropy, while the bottom figures plot the results of squentropy and the (rescaled) square loss. Test accuracy/ECE for each dataset are tabulated in Appendix C.4. squentropy has slightly better test accuracy and significantly smaller ECE than cross entropy or square loss. Squentropy has the best test accuracy in 71 out of 121 tasks and best calibration in 60 tasks. If only compare with cross entropy, squentropy is better in 94 tasks on accuracy, and is better on calibration in 83 tasks. Test accuracy and ECE for each dataset in this collection are tabulated in Appendix C.4.

In the results of Figure 4.3, we fix all random seeds to be the same for all cases and hence the test set is exactly the same. (Thus, we display legends only in the bottom-row figures). Color coding indicates the calculated probability of class label to be 1, according to the scale on th

Table 4.2. Standard deviation of test accuracy/error. Smaller number is bolded. CE is short for cross-entropy.

Model	Dataset	Squentropy	CE	Square loss
fine-tuned BERT	MRPC	**0.285**	0.766	0.484
	SST-2	**0.144**	0.173	0.279
	QNLI	**0.189**	0.205	0.241
	QQP	0.050	0.063	**0.045**
	text5	**0.132**	0.167	0.147
	text20	0.131	**0.08**	0.172
Transformer-XL	text8	**0.149**	0.204	0.174
	enwik8	0.156	**0.102**	0.228
LSTM +Attention	MRPC	**0.315**	0.786	0.484
	QNLI	**0.198**	0.371	0.210
	QQP	0.408	**0.352**	0.566
LSTM +CNN	MRPC	**0.289**	0.383	0.322
	QNLI	**0.154**	0.286	0.173
	QQP	0.279	**0.161**	0.458
Attention +CTC	TIMIT (PER)	0.332	**0.249**	0.508
	TIMIT (CER)	**0.232**	0.873	0.361
VGG+ BLSTMP	WSJ (WER)	**0.147**	0.249	0.184
	WSJ (CER)	0.082	0.118	**0.077**
VGG+ BLSTM	Libri (WER)	**0.117**	0.257	0.126
	Libri (CER)	**0.125**	0.316	0.148
Transformer	WSJ (WER)	**0.186**	0.276	0.206
	Libri (WER)	0.168	0.232	**0.102**
TCNN	MNIST	**0.151**	0.173	0.161
Resnet-18	CIFAR-10	**0.147**	0.452	0.174
	STL-10	0.413	0.376	**0.230**
W-ResNet	CIFAR-100	**0.164**	0.433	0.181
Visual Transformer	CIFAR-10	0.070	0.075	**0.063**
VGG	SVHN	0.283	**0.246**	0.307
Resnet-50	I-Net (Top-1)	**0.029**	0.045	0.032
	I-Net (Top-5)	0.098	**0.045**	0.126
EfficientNet	I-Net (Top-1)	**0.099**	0.122	0.138
	I-Net (Top-5)	0.092	**0.089**	**0.089**

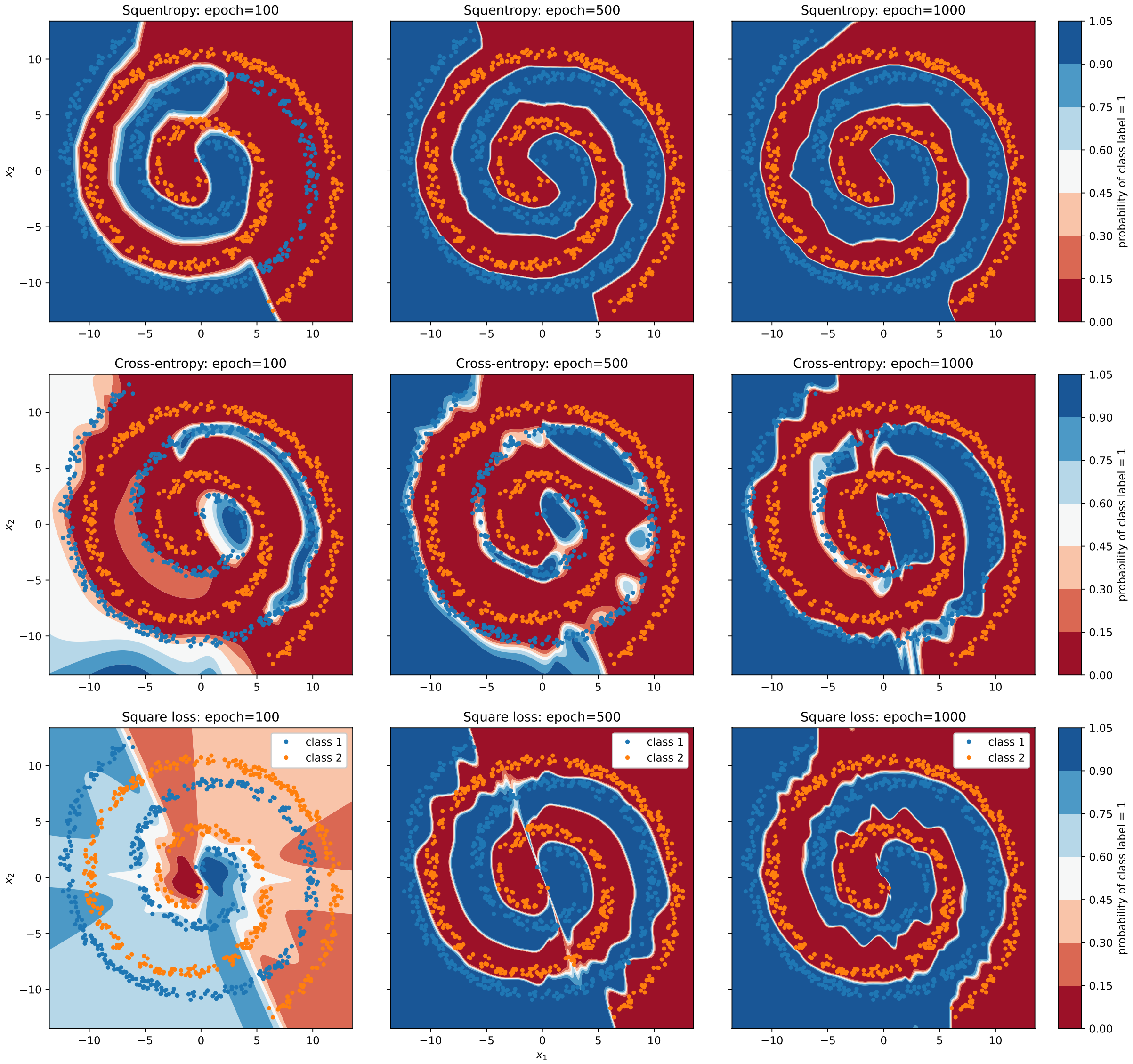


Figure 4.3. Decision boundary along different epochs for test samples.

eright. The white line between red and blue areas indicates the decision boundary. We train a 3-layer fully connected network with 12 units in each layer, for a 2-class spiral data set in $\mathbb{R}^2$. There are 1000 samples for training and 500 samples for test, and we train for 1000 epochs, yielding a training accuracy of 100% for all loss functions. Test accuracies are squentropy: 99.9%, cross entropy: 99.7%, square loss: 99.8%. *Top:* squentropy. *Middle:* cross entropy. *Bottom:* square loss. Columns show results after 100, 500, and 1000 epochs, respectively.

4.3.4 Robustness to initialization

To evaluate the stability of the model trained with the loss functions considered in this paper, we report the standard deviation of the accuracy/error rate with respect to the randomness in initialization of weights for NLP, speech, and vision tasks. Standard deviation is over 5 runs with different random initializations; see Table 4.2 for results. The standard derivation of squentropy is smaller in the majority of the tasks considered, so results are comparatively insensitive to model initialization.

4.4 Observations

As mentioned previously, we conjecture that the square term of squentropy acts as an implicit regularizer and in this section we provide some observations in support of this conjecture. We discuss the decision boundary learnt by a fully-connected network on a 2-class spiral data problem (the "Swiss roll") in Section 4.4.1, and remark on the weight norm of the last linear layer of several networks in Section 4.4.2.

4.4.1 Predicted probabilities and decision boundary

Using a simple synthetic setting, we observe that the decision boundary learned with squentropy appears to be smoother than that for cross entropy and the square loss. We illustrate this point with a 2-class classification task with spiral data and a 3-layer fully-connected network with parameter θ. This setup enables visual observations. Given a sample $\boldsymbol{x}_i \in \mathbb{R}^2$ and labels $y_i \in \{1,2\}$, and the one hot encoding $\boldsymbol{y}_i = [0,1]$ or $\boldsymbol{y}_i = [1,0]$, we solve for weights θ to define functions $f_1(\boldsymbol{x}_i)$ and $f_2(\boldsymbol{x}_i)$ corresponding to the two classes. For any $\boldsymbol{x}_i$, we then predict a probability of $\boldsymbol{x}_i$ being classified as class 1 as follows: $p(\boldsymbol{x}_i) := e^{f_1(\boldsymbol{x}_i)}/(e^{f_1(\boldsymbol{x}_i)} + e^{f_2(\boldsymbol{x}_i)})$. Samples are assigned to class 1 if $f_{i,1} > f_{i,2}$ and to class 2 otherwise. The decision boundary is the set of points for which $\{\boldsymbol{x} \,|\, f_1(\boldsymbol{x}) = f_2(\boldsymbol{x})\}$ or $\{\boldsymbol{x}|p(\boldsymbol{x}_i) = 1/2\}$.

We see from Figure 4.3 that the decision boundary obtained with squentropy is smoother than those learnt with both cross entropy and square loss. This appears to be true throughout the training process, on this simple example. Meanwhile, the margin (distance from training points to the decision boundary) is also larger for squentropy in many regions. Together, the large margin and smooth decision boundary imply immunity to perturbations and could be one of the reasons for the improved generalization resulting from the use of squentropy [24].

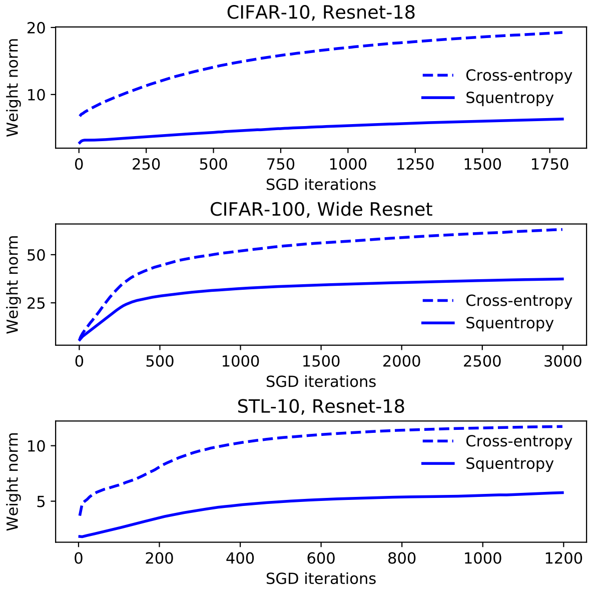


Figure 4.4. Weight norm along training.

4.4.2 Weight norm

Neural classifiers trained with cross-entropy loss suffer from overconfidence, causing miscalibration of the model [41]. Our calibration results in Figure 4.1 and Section C.3 show evidence of this phenomenon. As can be seen in the confidence histogram of cross entropy — the $(1,2)$ figure in Figure 4.1 — the average confidence $p_{y_i}(\boldsymbol{x}_i)$ for the predicted label in cross entropy is close to 1. This fact suggests that the logits $f_{y_i}(\boldsymbol{x}_i)$ of true class are close to ∞, while the logits of the incorrect classes approach $-\infty$. Such limits are possible only when the weights of last linear layer have large norm. To quote [89], "*cross-entropy loss thus inherently induces this tendency of weight magnification in neural network optimisation.*"

[41] comment that weight decay, which corresponds to adding a penalty term to the loss consisting of the sum of squares of the weights, can produce appreciably better calibration while having a minimal effect on test error; see the rightmost diagram in Figure 2 of [41]. In [89, 79], the authors point out how focal loss proposed in [77] improves calibration by encouraging the predicted distribution to have higher entropy, thus implicitly regularizing the weights. Figure C.1 of [89] compares weight norm and final logit values between cross entropy and the focal loss, showing that the latter are significantly smaller.

We perform a similar experiment, showing in Figure 4.4. We train a Resnet-18 on CIFAR-10 (calibration error, ECE: Squentropy: 8.9%, cross entropy: 10.0%) and STL-10 (ECE: Squentropy: 21.2%, cross entropy: 26.1%), a wide Resnet on CIFAR-100 (ECE: Squentropy: 10.9%, cross entropy: 17.9%), and show the norm of the last linear layer's weights. These are the same experiments as given in Table 4.1. The weight norm of the final-layer weights for three examples from Table 4.1 as a function of training steps. We observe that the weight norm for the model trained with squentropy is much smaller than the norms for the same set of weights in the model trained with cross entropy, along the whole training process.

4.5 Rescaled squentropy

Consider the following rescaled version of squentropy:

$$l_{\text{sqen}}(\boldsymbol{x}_i, y_i) = -\log p_{i,y_i}(\boldsymbol{x}_i) + \frac{\alpha}{C-1} \sum_{j=1, j\neq y_i}^{C} f_j(\boldsymbol{x}_i)^2, \tag{4.4}$$

which introduces a positive factor α into the second term of (4.1). Here $\alpha = 0$ corresponds to standard cross-entropy loss while $\alpha = 1$ yields the squentropy loss (4.1). Limited computational experiments show that $\alpha < 1$ the squentropy gives even better results for some tasks in Table 4.3, with significant improvements for such examples as TIMIT (CER), STL-10 and CIFAR-100. Mostly $\alpha = 0.1$ for the scaled squentropy results in Table 4.3.

Table 4.3. Test accuracy/error rate, and scaled sqen is short for rescaled squentropy. CE is short for cross-entropy.

Task	Scaled sqen	Squentropy	CE	Square loss
text5	**85.3**	85.2	84.5	84.6
text20	**81.5**	81.2	80.8	80.8
TIMIT(PER)	**19.0**	19.6	20.0	20.0
TIMIT(CER)	**29.6**	32.1	33.4	32.5
WSJ(WER)	5.3	5.5	5.3	**5.1**
WSJ(CER)	2.6	2.9	2.5	**2.4**
Librispeech(WER)	7.8	**7.6**	8.2	8.0
Librispeech(CER)	10.0	**9.7**	10.6	**9.7**
CIFAR-10	**86.0**	85.5	84.7	84.6
STL-10	**69.5**	67.7	68.9	65.4
CIFAR-100	**78.7**	77.5	76.7	76.5
SVHN	**93.8**	93.0	93.7	92.5
ImageNet (Resnet-50)	76.2	**76.3**	76.1	76.2
ImageNet (EfficientNet)	76.5	76.4	**77.0**	74.6

4.6 Summary, thoughts, future investigations

As with the selection of an optimization procedure, the choice of the loss function is an ineluctable aspect of training all modern neural networks. Yet the machine learning community has paid little attention to understanding the properties of loss functions. There is little justification, theoretical or empirical, for the predominance of cross-entropy loss in practice. Recent work by Hui & Belkin [51] showed that the square loss, which is universally used in regression, can perform at least as well as cross entropy in classification. Other works have made similar observations: [115, 118, 108, 16]. While several alternative loss functions, such as the focal loss [77], have been considered in the literature with good results, none have been adopted widely. Even the hinge loss, the former leader in the popularity contest for classification losses, is barely used outside the context of Support Vector Machines.

In this work we demonstrate that a simple hybrid loss function can achieve better accuracy and better calibration than the standard cross entropy on a significant majority of a broad range of classification tasks. Our squentropy loss function has no tunable parameters. Moreover, most of our experiments were conducted in a true "plug-and-play" setting using the same algorithmic parameters in the optimization process as for training with the standard cross-entropy loss. Performance of squentropy can undoubtedly be further improved by tuning the optimization parameters. Furthermore, various calibration techniques can potentially be applied with squentropy in the same way they are used with cross entropy.

Thus, from a practical point of view, squentropy currently appears to be the natural first choice to train neural models.

By no means does it imply that we know of fundamental reasons or compelling intuition indicating that squentropy is the last word on the choice of loss functions for classification. One of the main goals of this work is to encourage both practitioners and theoreticians to investigate the properties of loss functions, an important but largely overlooked aspect of modern Machine Learning.

4.7 Acknowledgements

Chapter 4, in full, is a reprint of Like Hui, Mikhail Belkin, and Stephen Wright. “Cut your Losses with Squentropy.” ICML, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Limitation of Neural Collapse on Understanding Generalization in Deep Learning

The recent work of [101] presented an intriguing “Neural Collapse” phenomenon, showing a structural property of interpolating classifiers in the late stage of training. This opened a rich area of exploration studying this phenomenon. Our motivation is to study how far understanding Neural Collapse can take us in understanding deep learning. First, we investigate its role in generalization. We refine the Neural Collapse conjecture into two separate conjectures: collapse on the train set (an optimization property) and collapse on the test distribution (a generalization property). We find that while Neural Collapse often occurs on the train set, it does not occur on the test set. We thus conclude that Neural Collapse is primarily an optimization phenomenon, with as-yet-unclear connections to generalization. Second, we investigate the role of Neural Collapse in representation learning. We show simple, realistic experiments where more collapse leads to *worse* last-layer features, as measured by transfer-performance on a downstream task. This suggests that Neural Collapse is not always desirable for representation learning, as previously claimed. Our work thus clarifies the phenomenon of Neural Collapse, via more precise definitions that motivate controlled experiments.

5.1 Introduction

In science, and in deep learning, novel empirical observations often catalyze deeper scientific understanding [69]. When faced with a new or surprising experiment, we can then try to understand the phenomenon more precisely: How universal is the behavior? In what settings does it hold? Can we describe it quantitatively?

What does it teach us more generally? This overall roadmap for understanding —from observations to quantitative conjectures & laws— has a long history of success in the natural sciences, and has also enjoyed recent successes in deep learning.

The recent "Neural Collapse" work of [101] initiated another instance of such a research program in understanding deep learning. Their work presented a new experimental observation, along with a partial characterization. At a high level, Neural Collapse conjectures several structural properties of deep neural networks when trained past the point of 0 classification error on the train set. Their most relevant conjecture to generalization— is "variability collapse (NC1)." Variability collapse proposes, informally, that when a deep network is trained on a k-way classification task, the last-layer representations converge to k discrete points. This is apriori surprising, since this internal structure is in no way required to achieve low train loss and high test performance: there exist networks with identical decision boundaries which do not satisfy collapse. However, our standard training methods (Stochastic Gradient Descent and variants) on standard architectures and datasets empirically seem to satisfy some form of collapse, as demonstrated in [101]. This work has since inspired many follow-up works investigating this phenomenon, both theoretically and empirically.

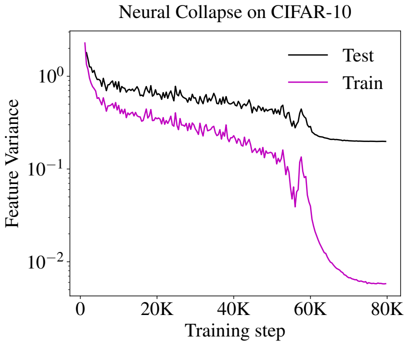


Figure 5.1. Failure of Test Collapse. Neural Collapse for ResNet18 on CIFAR-10. Collapse appears to occurs on the train set, but not on test.

A motivating factor in this research program is the belief that Neural Collapse is not an isolated phenomenon, but rather is deeply connected to other important and unsolved aspects of deep learning— in particular *generalization*. The problem of generalization, informally, is the study of why a model trained on a finite set of samples has good performance on out-of-sample inputs. Although this is not apriori related to Neural Collapse, the original work proposes that collapse “confers important benefits, including better generalization performance, better robustness, and better interpretability.” And it is stated as a hypothesis that “the benefits of the interpolatory regime of overparametrized networks are directly related to Neural Collapse” [101]. This postulated connection between Neural Collapse and generalization is implicit in many of the follow-up works as well, and motivates studying collapse as a phenomenon.

However, the nature of the connection between Neural Collapse and generalization remains muddled. Some works argue they are closely related [101], while others cast some doubt [23, 146, 4]. There are at least two reasons for this confusion in the literature: First, it is often not clear whether Neural Collapse refers to a phenomenon on the train set, or on the test set. The behaviors most relevant to generalization occur on the test set, and yet most experiments and theorems consider only the train set. Second, the Neural Collapse conjectures do not precisely specify the role of the sample size, and thus it is not always clear how to connect to generalization— where sample size is fundamental. This ambiguity is especially problematic because some natural ways to extend the Neural Collapse conjecture to the test set turn out to be impossible to satisfy, as we will describe.

Our Contributions.

We clarify ambiguities in the original Neural Collapse (NC) conjectures, which allows us to investigate which forms of NC are possible to achieve, both in theory and in practice. Specifically:

1. We propose more precise versions of the Neural Collapse conjectures (“variability collapse”), stating different versions for the train set and the test set, with both “strong” and “weak” forms. **(section 5.2)**

2. We discuss the theoretical feasibility of these different conjectures. As we will see, strong test collapse is extremely unlikely, while weak test collapse is in principle possible but does not occur in practice. **(section 5.2.1)**

3. We empirically confirm the finding of [101], that train-collapse occurs in many realistic settings. However, we find that test-collapse does not occur. **(section 5.3)**

4. We show several settings where increasing train-collapse is anti-correlated with test performance, in both on-distribution and transfer-learning settings. This demonstrates that train-time neural collapse is not always desirable— and indeed, can be counterproductive— for some kinds of generalization. **(section 5.4**).

We thus conclude that Neural Collapse is primarily an *optimization* phenomenon, and its connections to generalization require further investigation.

5.1.1 Related Works

The Neural Collapse phenomenon was originally presented in [101], and led to a series of follow-up works investigating and extending it. Many of the subsequent works develop simplified models in which Neural Collapse on the train set can be theoretically proven and understood. For example, [27] develops a “layer-peeled” model of training, and explores neural collapse in class-imbalanced settings. [86] proposes an alternate simplification, an “unconstrained features” model, in which train collapse also occurs. [22] and [146] also investigate the train collapse under unconstrained features model. Several works [105, 106, 112, 42] examine the Neural collapse with the square loss under different settings. Specifically, [105, 106] give theory which predicts the properties of neural collapse for homogeneous, weight-normalized networks. [112] proves that quasi-interpolating solutions obtained by gradient descent in the presence of weight decay have Neural collapse properties. [42] proposes a generic decomposition of the MSE loss which, under certain assumptions, results in a simplified dynamical description (the “central path”) which exhibits neural collapse on the train set. [81] extend theoretical analysis of neural collapse to the cross-entropy loss (while previous works mainly considered MSE loss). They prove neural collapse on the train set in the "unconstrained features" setting. [26] reformulate the last-layers of networks to convex formulations and give an explanation of Neural Collapse properties. [59, 58] proposes unconstrained layer-peeled model which captures the properties of Neural Collapse and prove that gradient flow on this model converges to critical points exhibiting neural collapse in its global minimizer. [140] and [103] observe that fixing the parameters (i.e. no back-propagation) in the final classifier as a simplex Equiangular Tight Frame (ETF) does not basically reduce the performance on the test set.

[78] investigate the impact of a margin parameter added to softmax/cosine softmax loss in the setting of few-shot learning and they show that this margin parameter controls the degree of NC. They also show that a higher margin parameter (larger intra-class variance) leads to higher accuracy on the validation set, but lower accuracy on unseen classes at training time. [38] study NC in the context of meta-learning, and the authors find that higher NC is better. They add a regularizer to the loss to increase NC appears to improve transfer learning results. [19] state that supervision collapse is an obstacle to learning good representations for few-shot learning. The definition of supervision collapse is the representations "represent only an image's (training-set) class, and discard information that might help with out-of-distribution classes", which is similar with the NC1 (feature collapse) definition. [116] propose a strategy to reduce NC, and show that doing so improves the performance of deep metric learning. [67] examine a variety of loss functions and find that loss functions that produce greater NC on the ImageNet training set sometimes get higher validation accuracy but transfer worse.

However, all the above papers present results for Neural Collapse on the train set, with an exception of [42], which gives a preliminary experiment on the test set collapse (see Figure 12 on page 20). They observe that "the rate of collapse is much slower on the test data compared to that on the train data", which agrees with our observations. [112] has a discussion on neural collapse and generalization and argues that neural collapse is not related to good generalization as "Neural Collapse is a property of the dynamics independently of the size of the margin which provides an upper bound on the expected error”. However, there has been no in-depth study on test data collapse and generalization.

One of the few papers focusing on collapse at test time is [35]. Their work focuses on neural collapse for transfer learning, under a particular assumption: classes in the source and target tasks are selected randomly from the same class distributions. In contrast to their work, our transfer-learning experiment does not obey the assumptions on source/target task required by [35], since we consider a source task which is a “class-superset” of the target task. Another key difference is that [35] use a notion of “collapse” which only requires collapse to occur in the limit of infinite train size. However, we consider “collapse” to occur if it occurs at finite train size. This finite-sample definition follows the original framework of [101], and is essential to a meaningful definition of collapse. We elaborate on this important point in Section 5.2. At first glance, the conclusion in the transfer learning setting arrived at by [35] contradicts to ours results. One reason for this can be the class number in their pretraining is larger than the downstream tasks, while we are considering a “super-class” setting where the downstream task has finer labels and larger class number. We pre-train on large datasets while fine-tune on tasks with limited data. This is a standard transfer learning setting where transfer learning is particularly useful in practice.

[146] and [87] provide empirical evidence that neural collapse can happen for training data with random labels. However, the presence of neural collapse on training data cannot indicate whether the network generalizes or not. [29] points out that negligence of valuable intra-class semantic difference is the reason for worse transferability of existing supervised pre-training methods, compared with the powerful transferability of self-supervised pre-training. They propose a new supervised pre-training method based on Leave-One-Out K-Nearest-Neighbor to preserve part of intra-class difference, i.e. to have less neural collapse. Extensive empirical studies show their method leads to better transferring to downstream tasks. Their conclusion agrees with ours. Inspired by the property of nearest-class center decision rule, [34] proposes “minimal NCC-depth” to capture the relationship of neural collapse and generalization, as they

also observe no clear relation between training data collapse and generalization. Note that the class-distance normalized variance (CNDV) definition used in this paper comes from [34] is from [35] and it is essentially the same as the definition in [101]. In contrast, our NC1 definitions explicitly consider the dependency on train set size, which is more precise than the CNDV definition and the original definition of [101]. Also, we extend the NC1 definition to the test set, with both "strong" and "weak" forms. For the empirical findings, on the difference between train set and test set collapse, the observations in Figure 1 of [34] is similar to ours, i.e. test-collapse seems to occur to a much less extent than train-collapse. Our main contribution is making this observation mathematically precise, by considering the asymptotic limit as a function of train samples (which has not been done in prior work as far as we are aware).

5.1.2 Notation

Let $\mathscr{X}$ be the input space, and $\mathscr{Y}$ be the label space. We consider multi-class classification problems, where $\mathscr{Y} = [k]$ for some $k \in \mathbb{N}$. Let $\mathscr{D}$ be the target distribution over $\mathscr{X} \times \mathscr{Y}$. *Training procedures*[1] are functions which map a train set $S \in (\mathscr{X} \times \mathscr{Y})^n$ and an iteration count $t \in \mathbb{N}$ and to a model f. In this work, we will always consider Stochastic-Gradient-Descent (SGD)-based training procedures, where t is the number of SGD steps. For a fixed train set S of size n, let f_S^t denote the model output by the training procedure after t iterations. So $\mathsf{Train} : (S,t) \mapsto f_S^t$, where Train denotes the training procedure. For a given model f_S^t, let the *last-hidden-layer feature map* be denoted $h_S^t : \mathscr{X} \to \mathbb{R}^d$. This is the feature-map induced by the trained model, as a map from inputs into $\mathbb{R}^d$.

5.2 Defining Neural Collapse

We first define two kinds of Neural Collapse: on the train set, and on the test set. Our definitions naturally extend the definitions in [101], but are more precise since we explicitly

[1] We can consider randomized training procedures by allowing an additional random string as input. We omit this randomness throughout, for notational clarity.

include the train/test distinction, and the dependency on training iterations t and train samples n. This is essential to describe the relevant asymptotic limits in the "collapse".

Throughout this work, we focus only on the first conjecture from [101]: "NC1 (Variability Collapse)." NC1 captures the within-class variance and it is the most relevant one to generalization. Also, the subsequent Simplex ETF conjecture is particularly meaningful only if NC1 is true, that is features cannot collapse to a simplex ETF if the variability does not "collapse" at all. When we refer to "neural collapse" in this work, we specifically are referring to "variability collapse." We first define collapse on the train set, which follows closely the definition in [101].

Definition 1 (Train-Collapse)**.** For a particular train set S, we say a training procedure T exhibits *Train-Collapse on S* if there exists some distinct $\mu_1, \mu_2, \ldots, \mu_k \in \boldsymbol{R}^d$ such that

$$\forall (x_i, y_i) \in S: \quad \lim_{t \to \infty} h_S^t(x_i) = \mu_{y_i}$$

That is, the trained network converges to representations such that all train points of class k get embedded to a single point μ_k (called the "class means" in [101]). The conjecture below then states conditions under which Train-Collapse occurs. This conjecture is meant to capture the original NC1 conjecture of [101], which was demonstrated empirically across many settings.

Conjecture 1 (Train-Collapse Conjecture, informal)**.** For all train sets S containing at least two distinct labels, and all training procedures T corresponding to SGD on "natural" sufficiently-deep and sufficiently-large neural network architectures: *T exhibits Train-Collapse on S.*

Crucially, we state Conjecture 1 for train sets of *all sizes*. This dependency on train set size is implicit, but omitted from [101] — it will become especially important when we discuss generalization, and this makes the biggest difference from the CDNV definition given by [35], which assume infinite train size. This behavior is called a "collapse" because regardless of the train set size, any big-enough network that enables neural collapse converge to this discrete limiting structure. We replicated this finding in most of our experiments. However, for completeness we acknowledge that this conjecture does not hold fully universally, and there are subtleties in practice[2]. Nevertheless, we believe the NC1 conjecture captures the right qualitative behavior in many realistic settings.

We also acknowledge that Conjecture 1, while more precise than the conjectures in [101], is still not fully formal. For example, it only applies to "natural" architectures and not all architectures, and does not quantify what "sufficiently large" means. In our experiments, we also apply weight decay, batch normalization (BN), tune different learning rates for each model. [111] shows that neural collapse does not necessarily happen when training without weight decay and without biases. [26] study the connection between NC and BN. Also, [34] shows that depth also matters for NC to happen. This restriction to "natural" architectures is a known obstacle to formalism in deep learning theory (e.g. [94]) and is necessary to avoid pathologies such as [1]. Nevertheless, our definitions take a step towards greater formalism, and this precision will be useful in understanding connections to generalization. Refining our definitions and conjectures further is an area for future work.

The notion of train-collapse described above (and in [101]) is an *optimization* notion: it involves only behavior of a model on its train set, and not behavior at test time. Thus, it is a priori unclear whether this notion is related to generalization aspects of models. To explore this, we first extend the definition of Neural Collapse to the test set, and then investigate whether this test-collapse occurs in practice. The most immediate way to formulate test collapse is to use the exact same formulation and quantifier on sample size n with Train-Collapse. We call this similar formulation with Train-Collapse Strong Test-collapse.

Definition 2 (Strong Test-Collapse)**.** A training procedure T exhibits *Strong Test-Collapse* on distribution $\mathscr{D}$ if for all sample sizes $n \in \mathbb{N}$, the following holds with probability 1 over sampling

[2] For example, we found in some settings training variability does not collapse to negligible value, such as CIFAR-10 and STL-10 dataset with VGG architectures (see Figure 5.3) . In some preliminary experiments we also found that adding stochasticity (such as dropout noise) often accelerated collapse, which is consistent with the theoretical model in [101].

$S \sim \mathscr{D}^n$: there exists some distinct $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ such that

$$\text{with prob 1 over } (x,y) \sim \mathscr{D}: \quad \lim_{t \to \infty} h_S^t(x) = \mu_{y^*(x)}$$

where $y^*(x) := \underset{y}{\operatorname{argmax}}\ p_{\mathscr{D}}(y \mid x)$ is the Bayes-optimal classification under distribution $\mathscr{D}$.

Strong Test-Collapse requires that test points x map to their "correct" embedding point μ_i, where i is the Bayes-optimal class for x. However, unless n is large enough that we are able to learn the Bayes-optimal classifier exactly, Strong Test-Collapse will not occur. Since this natural extension from Train-Collapse is hard to happen, we define a "weak" version of test set collapse which is likely to happen. It requires only that test points embed as *one of k* discrete points $\mu_1, \mu_2, \ldots \mu_k$, without requiring that all points of class i map to μ_i.

Definition 3 (Weak Test-Collapse)**.** A training procedure T exhibits *Weak Test-Collapse* on distribution $\mathscr{D}$ if for all sample sizes $n \in \mathbb{N}$, the following holds with probability 1 over sampling $S \sim \mathscr{D}^n$: there exists some distinct $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ such that

$$\text{with prob 1 over } (x,y) \sim \mathscr{D}: \quad \lim_{t \to \infty} h_S^t(x) \in \{\mu_i\}_{i \in [k]}$$

There are several important differences between the notions of test-collapse and train-collapse. First, for test-collapse we require that the train set S is not arbitrary, but sampled from some distribution $\mathscr{D}$. And we check for limiting behavior with respect to *new* samples from $\mathscr{D}$, as opposed to train samples from S. However, both train and test collapse require the collapse to occur *for all finite sample sizes n*, letting only time $t \to \infty$. This is the meaningful asymptotic, since taking limit of samples $n \to \infty$ would obscure almost all aspects of learning, which is most interesting at finite-sample sizes.

With the dependency on train set size, which is crucial when discussing generalization, our definitions are a natural extension of definitions given in [101] and [35], and they are a step forward to evaluate the correlation of neural collapse and generalization.

5.2.1 Remarks on Feasibility

With the above definitions, we can see that strong test-collapse is too strong a property to apply in realistic settings. We discuss this infeasibility here, and then corroborate this with experiments in the following section.

Infeasibility of Strong Test-Collapse.

First, note that both train-collapse and test-collapse definitions require that collapse occurs for all train set sizes $n \in \mathbb{N}$. This property is easy to satisfy for train-collapse, but is an extremely strong property for test-collapse. In particular, the "strong" form of test collapse (Definition 2) is too strong to hold in practice: it implies that a Bayes-optimal classifier can be extracted from the trained model features, even if the model is trained on only e.g. $n = 10$ samples. Even with large but finite n, it's hard to learn Bayes-optimal classifier exactly (and it is unlikely to happen in most realistic settings). This is because, according to Definition 2, the representation must map test inputs to their "correct" cluster, and thus the correct label can be extracted from the cluster identity.

However, the "weak" form of NC1-test (Definition 3) still has hope of holding, since it does not imply learning a Bayes-optimal classifier. Nevertheless, note that even the "weak" form is a fairly strong condition for neural networks: it implies that trained networks (on *any* size train set) learn feature-maps h such that the push-forward $h_*(\mathscr{D})$ is a discrete measure. Mapping the continuous measure $\mathscr{D}$ to a discrete measure is a strong property, and one that is unlikely to hold for standard neural networks.

Feasibility of Weak-Collapse.

While weak-collapse is unlikely to hold for neural networks trained with SGD, the definition itself is non-vacuous: there exist learning methods which are "reasonable" (asymptotically consistent) and exhibit weak test-collapse. To see this, consider the following modified training procedure: first, train a neural network as usual to get a network $f : \mathscr{X} \to \mathscr{Y}$. Then, construct another network f' such that the *last-layer representation* of f' is a one-hot encoding of the *classification decision* of f. That is, the representation $h'(x) \in \mathbb{R}^k$ satisfies $h'(x) := \vec{e}_{f(x)}$ where $\{\vec{e}_i\}$ are standard basis vectors. This can be constructed by, for example, adding post-processing layers to f. Now, the training procedure which outputs f' will satisfy weak test-collapse of its representations, since its representations are always one of the k standard basis vectors by construction.

Desirability of Neural Collapse for Generalization.

Armed with these definitions, we can now consider whether train or test collapse are necessary or sufficient for on-distribution generalization. First, neither train nor test collapse are strictly necessary for good generalization: As discussed, it is possible to construct models with identically good generalization performance, but which satisfy neither train nor test collapse. There are even natural, non-contrived examples of this: models trained for less than one epoch (the "Ideal World" in the terminology of [95]) will not exhibit train collapse, because they are not trained to fit their train set. And yet, as demonstrated in [95], they can match the performance of interpolating models. This "one epoch" regime is also relevant in practice, where models are trained on massive data sources such as internet scrapes, often for less than one epoch [8, 109, 66].

Further, neither train collapse (definition 1) nor weak test-collapse (definition 3) are sufficient for generalization. It is possible to construct models which satisfy train collapse perfectly, but which are random functions at test time. Likewise, it is possible to construct models which satisfy weak test-collapse, but have random classification decisions.

Strong test-collapse (definition 2) *is* sufficient for good test performance, since it implies that test inputs map to the "correct" cluster in representation-space. However, as we discussed, strong test-collapse is infeasible, and impossible in practice.

5.3 Experiments: Train and Test Collapse

Here we complement our theoretical discussion by measuring both train and test collapse in realistic settings, following the experiments of [101]. We find that train-collapse occurs in many settings, while test-collapse (both strong and weak) does not. We also show the dependency on the train set size: larger train sets lead to *stronger* test collapse, but *weaker* train collapse. Note that we say stronger collapse or more collapse when the feature variance is smaller. This further highlights the importance of distinguishing between the two forms of collapse, since they can be anti-correlated. Also, stronger train set collapse can lead to worse test performance, which means that stronger collapse on train set itself is not correlated with better generalization.

5.3.1 Measuring Collapse

It is not possible to measure collapse strictly according to definitions 1 to 3, since they involve a $t \to \infty$ limit. Instead, we follow exactly the experimental procedure of [101], and measure approximations which capture the "degree of collapse." We restate their procedure here for convenience. Measuring collapse require finding the vectors $\mu_1, \mu_2, \dots \mu_k \in \mathbb{R}^d$, which embeddings collapse to. The choice of these vectors depends on the setting, as below.

Train Collapse. For the train set, μ_i is defined as the train class-means:

$$\hat{\mu}_i := \mathop{\mathbb{E}}_{(x,y)\in S}[h_S^T(x) \mid y = i]$$

where T is the maximum train time in the experiment. Define the global mean as $\hat{\mu} := \sum_i \hat{\mu}_i / |\mathscr{Y}|$. Then, the "degree of train collapse" is measured as:

$$\mathsf{TrainVariance}(t) := \frac{\mathbb{E}_{(x,y)\in S}[||h_S^t(x) - \hat{\mu}_y||^2]}{\mathbb{E}_i[||\hat{\mu}_i - \hat{\mu}||^2]}$$

Smaller values of this quantity indicate more "collapse." The numerator here is the "within-class variance" and it is normalized by the "between-class variance", in the terminology of [101]. This definition follows the experimental measurements in [101].

Strong Test Collapse. For test collapse, μ_i is defined as the test class-means:

$$\overline{\mu}_i := \underset{(x,y)\sim\mathscr{D}}{\mathbb{E}}[h_S^T(x) \mid y = i]$$

The global mean is $\overline{\mu} := \sum_i \overline{\mu}_i / |\mathscr{Y}|$. Then, the "degree of strong test collapse" is measured as:

$$\mathsf{StrongTestVariance}(t) := \frac{\mathbb{E}_{(x,y)\sim\mathscr{D}}[||h_S^t(x) - \overline{\mu}_y||^2]}{\mathbb{E}_i[||\overline{\mu}_i - \overline{\mu}||^2]}$$

Weak Test Collapse. For weak test-collapse (definition 3), we do not require that representations collapse to their *class means*, but simply to some μ_i. Thus, we define $\{\widetilde{\mu}_i\}$ as the result of k-means clustering on the following set of vectors: $\{h_S^T(x)\}_{x\in\text{TestSet}}$. The global mean is $\widetilde{\mu} := \sum_i \widetilde{\mu}_i / |\mathscr{Y}|$. And the "degree of weak test collapse" is measured as:

$$\mathsf{WeakTestVariance}(t) := \frac{\mathbb{E}_{(x,y)\sim\mathscr{D}}[\underset{i\in[k]}{\text{argmin}}\ ||h_S^t(x) - \widetilde{\mu}_i||^2]}{\mathbb{E}_i[||\widetilde{\mu}_i - \widetilde{\mu}||^2]}$$

5.3.2 Experimental Results

Setup. We consider image classification tasks with MNIST, FashionMNIST, CIFAR-10, SVHN and STL-10 datasets. We train Resnet, DenseNet and VGG networks with stochastic gradient descent (SGD) to minimize the cross-entropy loss. All tasks were trained on a single GPU with batch size 128 and 80000 SGD iterations. See Appendix D.1 for more details and references about the datasets, architectures and training mechanisms.

In the following of this section we show that the test collapse does not occur with experiments on a wide range of datasets and model architecture combinations. We show that train collapse and test collapse can be anti-correlated and more train collapse can lead to worse test performance. Considering the dependency on train set size is fundamental to generalization.

Failure of Test Collapse.

In figure 5.2, we train a single model (ResNet-18 on CIFAR-10) and measure TrainVariance, WeakTestVariance, and StrongTestVariance as a function of train time t. That is, we measure the degree of train and test collapse over increasing time. We see that train collapse appears to occur, while test variance does not decrease to negligible value. In particular, there is a "generalization gap" in the Train vs. Test Variances: the TrainVariance appears to converge to 0 as $t \to \infty$, while TestVariance (both weak and strong) do not. For the remainder of the experimental results, we plot only "strong" test collapse, since we generally observe that both strong and weak collapse have similar behavior.

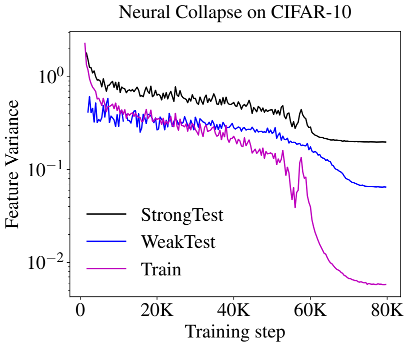


Figure 5.2. Neural Collapse on CIFAR-10. Collapse occurs on the train set, but not on the test set (neither Strong nor Weak).

In figure 5.3, we train different models on various datasets and measure TrainVariance and StrongTestVariance as a function of train time t. We train all models to get 0 training error and continue training to achieve close to 0 training loss[3]. We see Strong Test-Collapse does not occur on all settings, and has a large gap with Train Collapse. Again, the results show that Neural Collapse is mainly an optimization phenomenon and not a generalization one: test set does not collapse to negligible value in any setting, together with our theoretical argument of infeasibility of Strong Test-Collapse in 5.2.1, we claim a failure of test collapse.

The numbers up the dots are corresponding test accuracy of different train set size (N). We observe that train and test collapse are anti-correlated and small train variance has worse test accuracy. That is neural collapse can hurt generalization. *Left* of Figure 5.4 is ResNet18 trained on subsets of CIFAR-10. *Right* of Figure 5.4 is for VGG11 trained on subsets of FashionMNIST.

[3] We use "close to 0" to mean when the loss is below 10^{-5}.

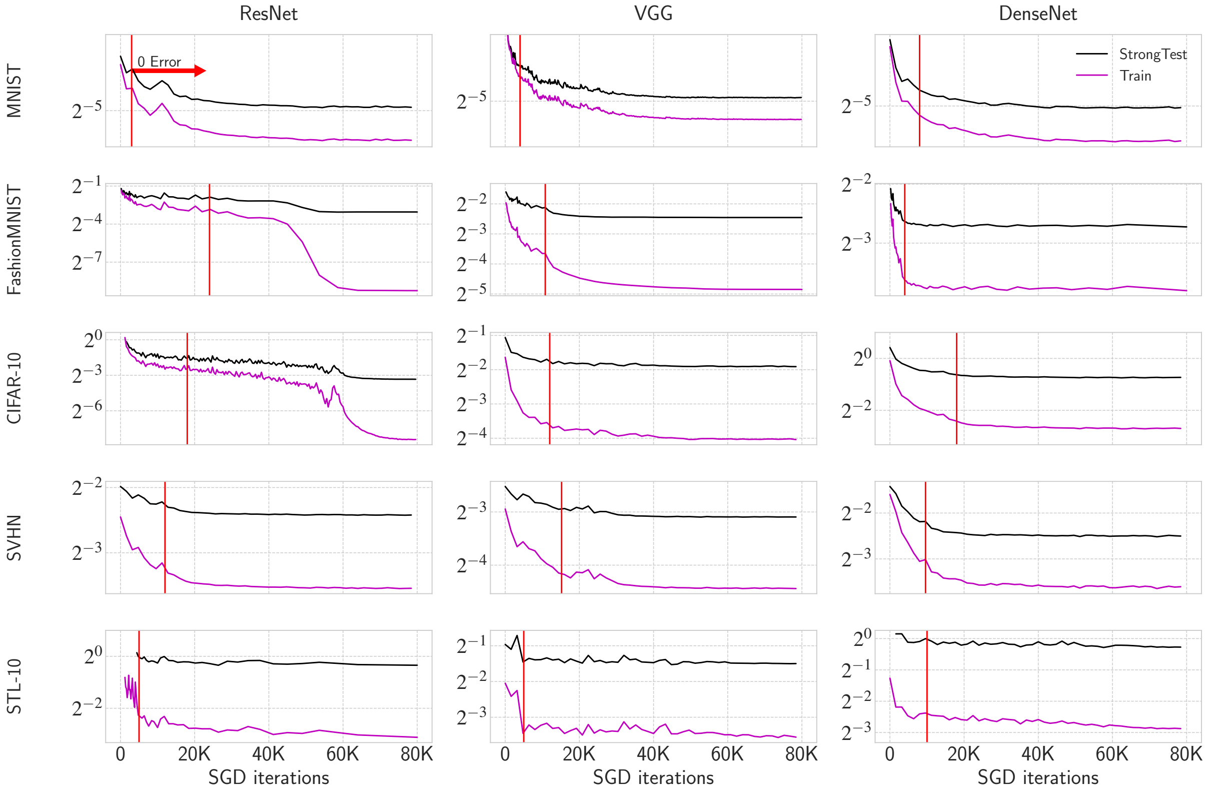


Figure 5.3. Failure of Test Collapse. Training and test variance vs. SGD iterations, for various dataset and architecture combinations. All test sets (black line) do not collapse to negligible variance, and have much less collapse than the train sets (purple line).

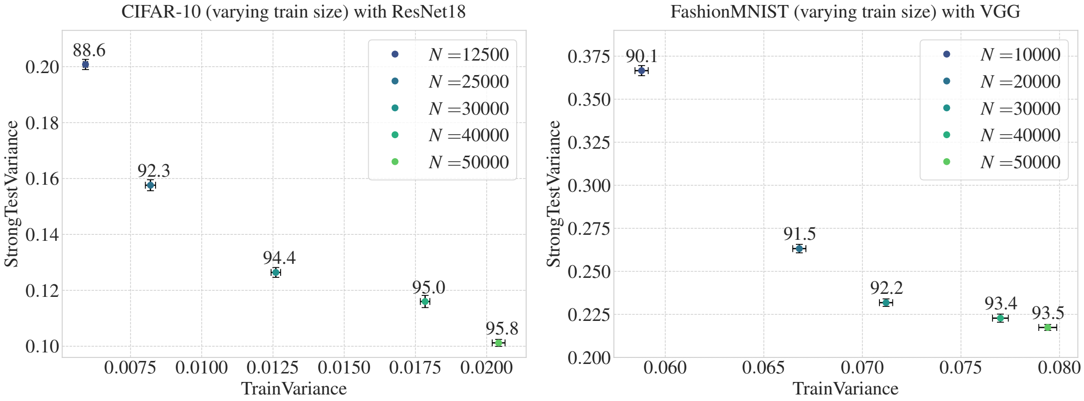


Figure 5.4. Train vs. Test Anti-Correlation.

Train vs. Test Anti-Correlation.

In figure 5.4 we train a ResNet18 on CIFAR-10, and vary the size of the train set from $N = 12500, 25000, 30000, 40000$ to $N = 50000$. We also report results on training a VGG11 network with batch normalization on different subsets of FashionMNIST. For each train set size N, we have 5 runs with different random seeds (the subset of each run is different because of random selection), and report the average of the variance and plot the error bar.

We train all models past the point of 0 training loss and stop training when the training loss decreases to 10^{-6} in each run. figure 5.4 plots the train collapse (TrainVariance) compared to the test collapse (StrongTestVaraince) at the end of training, for different train set sizes. We also report the corresponding test accuracy right up the dots in figure 5.4. We find that as the train set size increases, the train variation increases (less train collapse), however, the test accuracy gets higher and the test variation decreases (more test collapse). This illustrates that test and train collapse are not always correlated, and thus it is important to distinguish between the two: "better" optimization behavior (train set collapse) accompanies worse generalization behavior. Also, by considering the dependency on train set size, we see that stronger train set collapse itself does not imply good generalization. If you look at the train variance in x-axis and the accuracies up in the dots in Figure 5.4, it shown that solutions that does not exhibit train collapse (larger train variance) actually have good generalization (better test accuracy). Also, this observation matches the claim in [95] which say "1-epoch" CIFAR models have similar performance as multi-epoch models but the "1-epoch" models do not exhibit train collapse.

One limitation of this experiment is that we evaluate collapse at finite train time, and not at $t = \infty$. Indeed, at $t = \infty$ we expect the train variation to be identically 0 for all data sizes (by the definition of collapse), but the test variation to decay with larger data sizes. This situation is analogous to measuring train/test error itself for overparameterized models: for large enough models, train error will always be 0, but test error will decay with the data size. This experiment thus highlights the importance of measuring both train & test quantities, and the subtlety involved in measuring collapse at finite time.

We also acknowledge that in this experiment, increasing the size of the train set is correlated with both better test collapse, and better generalization. However, we caution that this should not be seen as evidence that test collapse is mechanistically related to generalization. First, because the test variance does not truly “collapse”, it just reduces, as already discussed. And second, because this reduction in test variance is in some sense necessary for any model with improved test error— since high test variance would produce noisy classification decisions. Thus, the correlation of test variance and generalization in this experiment should not be surprising. We are cautious to make the claim on the correlation of test set collapse and generalization, and we think this needs more careful study. We think it’s better to leave this question in a separate work, as this work mainly focuses on the correlation of train set collapse and generalization and most previous works on neural collapse focus on the train set.

5.4 Collapsed Features Transfer Worse

In the previous section, we showed that train-time collapse can be anti-correlated with generalization performance, when measuring generalization on-distribution. We now explore generalization on other distributions, by considering transfer learning. We consider the standard transfer-learning setting, where models are usually pre-trained on massive datasets and then fine-tuned for tasks on small datasets. Now we investigate generalization on downstream tasks, to understand the role of Neural Collapse in *transfer-learning* and *representation learning*.

5.4.1 Test Collapse implies Bad Representations

We first observe that, using our definition of test collapse, a model which has fully test-collapsed will have representations that are bad for most downstream tasks. To see this, consider the following example. Suppose we have a distribution $\mathscr{D}$ with ten types of images (as in CIFAR-10), but we group them into two superclasses, such as “animals” and “objects.” We then train a classifier on this binary problem (e.g. CIFAR-10 images with these binary labels). Let the feature map of the fully-trained model (that is, the limiting model as $t \to \infty$) be denoted

h. If this model exhibits even weak test collapse, then there exist vectors $\{\mu_1, \mu_2\}$ such that the representations satisfy:

$$\Pr_{x \sim \mathscr{D}}[h(x) \in \{\mu_1, \mu_2\}] = 1. \tag{5.1}$$

That is, the representations will by definition "collapse": every input $x \sim \mathscr{D}$ will map to exactly one of two points μ_1, μ_2. This property is clearly undesirable for representation learning. For example, suppose we use these representations for learning on a related task: the original 10-way classification problem. It is clear that no classifier using the fixed representations (linear probing scheme) from h can achieve more than 20% test accuracy on the original 10-way task: each group of 5 classes will collapse to a single point after passing through h (by equation (5.1)), and will become impossible to disambiguate among these 5 classes. This example is formalized in the lemma below.

Lemma 5. *Let $(x,y) \sim \mathscr{D}$ be any target distribution defining a balanced $2k$-wise classification task. Let $\mathscr{D}_2$ be the pretraining distribution, defined by super-classing $\mathscr{D}$ into a balanced binary classification task. That is, the distribution of $\mathscr{D}_2$ is given by $(x, F(y))$ for $(x,y) \sim \mathscr{D}$ and some balanced partition $F : [2k] \to \{0,1\}$.*

Now let $h : \mathscr{X} \to \mathbb{R}^d$ be pretrained representations that are fully test-collapsed with respect to $\mathscr{D}_2$, with $\mathsf{WeakTestVariance}$ exactly 0.

Then, all classifiers that input only representations $h(x)$ have test accuracy at most $1/k$ on $\mathscr{D}$. Formally, for all functions $f : \mathbb{R}^d \to \{0,1,..,2k-1\}$, the test accuracy

$$\Pr_{(x,y) \sim \mathscr{D}}[f(h(x)) = y] \leq \frac{1}{k}.$$

As Weak Test-collapse is a weaker condition than Strong Test-collapse, Lemma 5 also holds for Strong Test-Collapse. The proof of Lemma 5 is straightfoward, and included for completeness in Appendix D.3. In our experimental results below, we show the variance of Strong Test-collapse. Even if we do not fix the pre-trained representations and fine-tune all the parameters (fine-tuning scheme), as shown in our experimental results 5.4.2, fully collapsed features (less within-class variance) lead to worse down-stream task performance. The results in [29] show similar observations with ours and their proposed method to enlarge within-class variance actually improves test performance. This shows that test collapse is undesirable for even an extremely simple transfer learning task (where we transfer to the same distribution, with finer label structure). In the following sections, we will demonstrate this result experimentally, even for classifiers which have not fully collapsed.

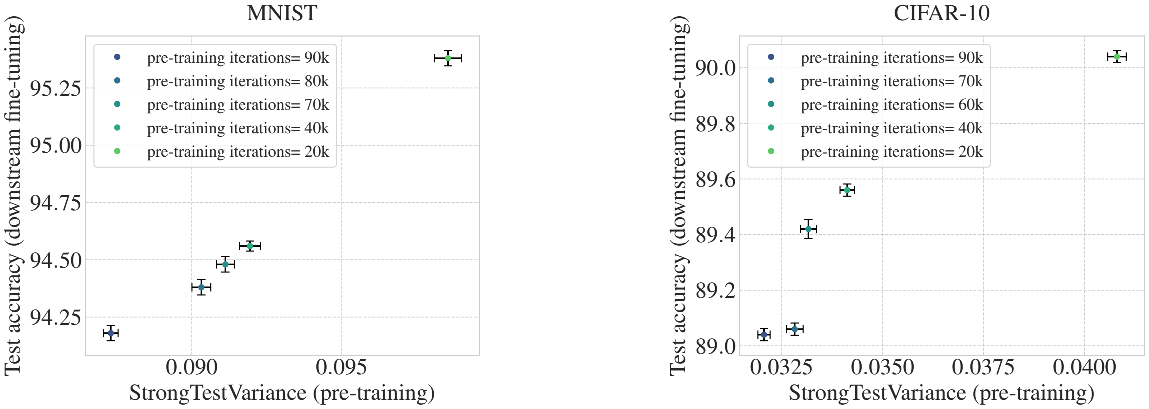


Figure 5.5. Collapsed Features Transfer Worse.

We save different checkpoints during pre-training, and use them to initialize the downstream models. We fine-tune all the parameters of the model. In Figure 5.5 the x-axis shows the StrongTestVariance of those checkpoints on the pre-training test set, and y-axis shows the test accuracy after fine-tuning on downstream tasks. We find that stronger test collapse (i.e. lower variance) is correlated with lower downstream test accuracy. *Left* of Figure 5.5 is for MNIST with a 3 hidden layer fully-connected network. *Right* of of Figure 5.5 is for CIFAR-10 with a standard Resnet18.

5.4.2 Experiments

There are many relevant settings in transfer learning, especially in practice. For example, in practice, we often pre-train on a "generic" task with massive datasets, and then fine-tune on a specific task with limited data. This specific task may involve finer-grained labels than the generic task, which parallels our experimental setup. We train a 3 hidden layer fully-connected networks with 1024 units per layer on MNIST, and a standard Resnet18 on CIFAR-10. For pre-training, we use a subset of the train set and perform 2-class classification (via super-classing). For *fine-tuning*, we use the weights pre-trained as initialization of the weights other than the last classification layer, and do standard (10-class for MNIST, and 8-class for CIFAR-10) classification with a much smaller held-out subset. We do not report results with linear probing, as it gives much worse transfer-performance than *fine-tuning* scheme. See more details in Appendix D.2.

Here we show transfer learning results on MNIST and CIFAR-10. To see the correlation between neural collapse (on test set) and generalization, we plot the degree of test set collapse during pre-training and test performance in down-stream tasks. We report the average of 5 runs with different random seeds, and give the error bars, as illustrated in figure 5.5. We see that for both MNIST and CIFAR-10, the checkpoints with more Test Collapse gives worse transfer-performance on downstream tasks. That is, in these settings more Test Collapse actually leads to learning worse features. This demonstrates that neural collapse does not always lead to good representation learning— when the class number in pre-training is less than the number of downstream tasks. collapse actually harms representation quality.

5.5 Conclusion

We show that Neural Collapse is primarily an optimization phenomenon, and does not always correlate with better generalization. We propose more precise definitions— "strong" and "weak" Neural Collapse for both the train set and the test set— which disentangle generalization and optimization behaviors. We believe these more precise definitions aid in clarifying the literature around neural collapse, and will help guide further study. By investigating the train and test collapse on various dataset and architectures, we show that while train collapse reliably occurs in many settings, test collapse does not. Our theoretical formulations and empirical observations suggest that while neural collapse continues to be an intriguing phenomenon and a promising optimization research program, its relevance to generalization requires further study.

5.6 Acknowledgements

Chapter 5, in full, is a reprint of Like Hui, Mikhail Belkin, and Preetum Nakkiran. “Limitations of neural collapse for under- standing generalization in deep learning.” arXiv preprint arXiv:2202.08384 (2022). The dissertation author was the primary investigator and author of this paper.

Chapter 6

Kernel Machines in Speech Enhancement

We apply a fast kernel method for mask-based single-channel speech enhancement. Specifically, our method solves a kernel regression problem associated to a non-smooth kernel function (exponential power kernel) with a highly efficient iterative method (EigenPro). Due to the simplicity of this method, its hyper-parameters such as kernel bandwidth can be automatically and efficiently selected using line search with subsamples of training data. We observe an empirical correlation between the regression loss (mean square error) and regular metrics for speech enhancement. This observation justifies our training target and motivates us to achieve lower regression loss by training separate kernel model per frequency subband. We compare our method with the state-of-the-art deep neural networks on mask-based HINT and TIMIT. Experimental results show that our kernel method consistently outperforms deep neural networks while requiring less training time.

6.1 Introduction

Speech enhancement aims at reducing noise from speech and the challenging problem of this task has received significant attention in research and applications. In recent years the dominant methodology for addressing single-channel speech enhancement has been based on neural networks of different architectures [128, 144]. Deep Neural Networks (DNNs) present an attractive learning paradigm due to their empirical success on a range of problems and efficient optimization.

In this paper, we demonstrate that modern large-scale kernel machines are a powerful alternative to DNNs, capable of matching and surpassing their performance while utilizing less computational resources in training. Specifically, we take the approach to speech enhancement based on the Ideal Binary Mask (IBM) and Ideal Ratio Mask (IRM) methodology. The first application of DNNs to this problem was presented in [131], which used a DNN-SVM (support vector machine) system to solve the classification problem corresponding to estimating the IBM. [130] compared different training targets including IRM. [139] proposed a regression-based approach to estimate speech log power spectrum. Recently, [132] applies recurrent neural networks to similar mask-based tasks and [99] applies convolutional networks to the spectrum-based tasks.

Kernel-based shallow models (which can be interpreted as two-layer neural networks with a fixed first layer), were also proposed to deal with speech tasks. In particular, [50] gave a kernel ridge regression method, which matched DNN on TIMIT. Inspired by this work, [9] applied an efficient one-vs-one kernel ridge regression for speech recognition. [82] developed kernel acoustic models for speech recognition.

Notably, these approaches require large computational resources to achieve performance comparable to neural networks.

In our opinion, the computational cost of scaling to larger data has been a major factor limiting the success of these methods. In this work we apply a recently developed highly efficient kernel optimization method EigenPro [83], which allows kernel machines to handle large datasets.

We conduct experiments on standard datasets using mask-based training target. Our results show that, with EigenPro iteration, kernel methods can consistently outperform the performance of DNN in terms of the target mean square error (MSE) as well as the commonly used speech quality evaluation metrics including perceptual evaluation of speech quality (PESQ) and short-time objective intelligibility (STOI).

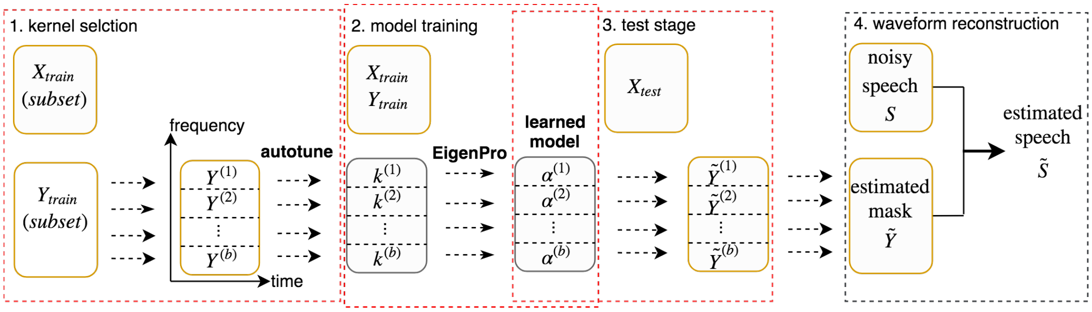


Figure 6.1. Kernel-based speech enhancement framework

The contributions of our paper are as follows:

1. Using modern kernel algorithms we show performance on mask-based speech enhancement surpassing that of neural networks and requiring less training time.
2. To achieve the best performance, we use exponential power kernel, which, to the best of our knowledge, has not been used for regression or classification tasks.
3. The simplicity of our approach allows us to develop a nearly automatic hyper-parameter selection procedure based on target speech frequency channels.

The rest of the paper is organized as follows. Section 6.2 introduces our proposed kernel-based speech enhancement system: kernel machines, exponential power kernel, automatic hyper-parameter selection for subband adaptive kernels. Experimental results and time complexity comparisons are discussed in Section 6.3. Section 6.4 gives the conclusion.

6.2 Kernel-Based Speech Enhancement

6.2.1 Kernel Machines

The standard kernel methods for classification/regression denote a function f that minimizes the discrepancy between $f(\boldsymbol{x}_j)$ and y_j, given labeled samples $(\boldsymbol{x}_j, y_j)_{j=1,\ldots,n}$ where $\boldsymbol{x}_j \in \mathbb{R}^d$ is a feature vector and $y_j \in \mathbb{R}$ is its label.

Specifically, the space of f is a Reproducing Kernel Hilbert Space $\mathbb{H}$ associated to a positive-definite kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We typically seek a function $f^* \in \mathbb{H}$ for the following optimization problem:

$$f^\star = \text{ argmin } f(\boldsymbol{x}_j) = y_j, j = 1, 2, ..., n \, \|f\|_{\mathbb{H}}, \tag{6.1}$$

According to the Representer Theorem [120], f^* has the form

$$f(\boldsymbol{x}) = \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}, \boldsymbol{x}_j), \tag{6.2}$$

To compute f^* is equivalent to solve the linear system,

$$K\boldsymbol{\alpha} = (y_1, \cdots, y_n)^T, \tag{6.3}$$

where the kernel matrix K has entry $[K]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{\alpha} \triangleq (\alpha_1, \cdots, \alpha_n)^T$ is the representation of f under basis $\{k(\cdot, \boldsymbol{x}_1), \cdots, k(\cdot, \boldsymbol{x}_n)\}$.

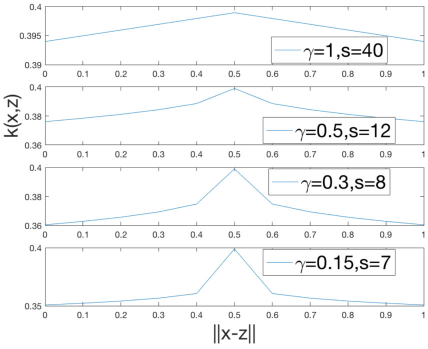
6.2.2 Exponential Power Kernel

We use an exponential power kernel of the form

$$k_{\gamma,\sigma}(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^\gamma}{\sigma}) \tag{6.4}$$

for our kernel machine, where σ is the kernel bandwidth and γ is often called shape parameter.

[37] shows that the exponential power kernel is positive definite, hence a valid reproducing kernel. This kernel also covers a large family of reproducing kernels including Gaussian kernel ($\gamma = 2$) and Laplacian kernel ($\gamma = 1$).



We observe that in many noise settings of speech enhancement, the best performance is achieved using this kernel with shape parameter $\gamma \leq 1$, which is highly non-smooth. In the right side figure, we plot this kernel function with parameters that we use in our experiments. We have not seen any application of this kernel (with $\gamma < 1$) in supervised learning literature.

6.2.3 Automatic Subbands Adaptive Kernels

Algorithm 1. Automatic hyper-parameter selection[1]

```
Input: $\boldsymbol{D}_{train}, \boldsymbol{D}_{val}$: training and validation data, $\Gamma$: a set of $\gamma$ for the exponential power kernel,
$\sigma_l, \sigma_h$: smallest and largest bandwidth
Output: selected kernel parameters $\gamma_{opt}, s_{opt}$ for $D_{\text{train}}$
procedure autotune($\boldsymbol{D}_{train}, \boldsymbol{D}_{val}, \Gamma, \sigma_l, \sigma_h$)
    define subprocedure cross-validate($\gamma, \sigma$) as: train one kernel model with $k_{\gamma,\sigma}$ on $\boldsymbol{D}_{train}$
using EigenPro iteration, return its loss on $D_{val}$.
    for $\gamma$ in $\Gamma$ do
        $\sigma_\gamma$ = search(cross-validate($\gamma, \cdot$), $\sigma_l, \sigma_h$)
    $\gamma_{opt}, \sigma_{opt} \leftarrow \text{argmin}_{\gamma \in \Gamma, \sigma_\gamma}$cross-validate($\gamma, \sigma_\gamma$)
     return $\gamma_{opt}, \sigma_{opt}$
procedure search($f, \sigma_l, \sigma_h$)
    if ($\sigma_h - \sigma_l \leq 2$) then
        return $\sigma_l$
    select $\sigma_{m1}, \sigma_{m2} \in (\sigma_l, \sigma_h)$
    compute $f(\sigma_l), f(\sigma_{m1}), f(\sigma_{m2}), f(\sigma_h)$
     min{$f(\sigma_l), f(\sigma_{m1}), f(\sigma_{m2}), f(\sigma_h)$} $f(\sigma_l)$: return search($f, \sigma_l, \sigma_{m1}$) $f(\sigma_{m1})$: return
search($f, \sigma_l, \sigma_{m2}$) $f(\sigma_{m2})$: return search($f, \sigma_{m1}, \sigma_h$) $f(\sigma_h)$: return search($f, \sigma_{m2}, \sigma_h$)
```

As empirically shown in Section 6.3.3, we see that models with lower MSE at every frequency channel consistently outperform other models in STOI. This motivates us to achieve lower MSE for all frequency channels by tuning kernel parameters for each of them. In practice, we split the band of frequency channels into several blocks , which we call *subband*s.

We propose a simple kernel-based framework as depicted in Fig. 6.1 to achieve automatic parameter selection and fast training for each subband. For i-th subband, the framework learns one model $f^{(i)}$ related to an exponential power kernel $k^{(i)}$ with parameters automatically tuned

[1] We apply memoization technique for computing cross-validate$(\cdot, \cdot)$. We first attempt to set σ_{m1}, σ_{m2} as a value that is already used in (σ_l, σ_h), then we choose them to split (σ_l, σ_h) into three parts as equal as possible.

Table 6.1. Kernel & DNN on TIMIT: (MSE: lowest is best, STOI and PESQ: highest is best. Best results bolded.)

Noise Type	Metrics	5 dB			0 dB			-5 dB		
		Kernel	DNN	Noisy	Kernel	DNN	Noisy	Kernel	DNN	Noisy
Engine	MSE ($\cdot 10^{-2}$)	**1.10**	1.41	-	**1.34**	1.86	-	**1.17**	1.82	-
	STOI	**0.91**	0.90	0.80	**0.86**	0.85	0.68	**0.80**	0.77	0.57
	PESQ	**2.77**	**2.77**	1.97	**2.51**	2.45	1.66	**2.19**	2.16	1.41
Babble	MSE ($\cdot 10^{-2}$)	**3.34**	3.49	-	**4.18**	4.37	-	**4.94**	5.43	-
	STOI	**0.86**	**0.86**	0.77	**0.77**	**0.77**	0.66	**0.64**	**0.64**	0.55
	PESQ	**2.54**	2.52	2.08	**2.12**	2.10	1.73	**1.70**	1.61	1.42
SSN	MSE ($\cdot 10^{-2}$)	**1.35**	1.53	-	**1.48**	1.67	-	**1.60**	1.76	-
	STOI	**0.88**	**0.88**	0.81	**0.82**	**0.82**	0.69	**0.74**	**0.74**	0.57
	PESQ	**2.68**	2.66	2.05	**2.36**	2.32	1.75	**2.03**	2.00	1.48
Oproom	MSE ($\cdot 10^{-2}$)	**1.44**	1.85	-	**1.34**	1.86	-	**1.17**	1.82	-
	STOI	**0.88**	**0.88**	0.79	**0.84**	0.83	0.70	**0.79**	0.76	0.59
	PESQ	**2.80**	2.79	2.16	**2.50**	2.47	1.78	**2.23**	2.12	1.40
Factory1	MSE ($\cdot 10^{-2}$)	**2.51**	2.53	-	**2.52**	2.55	-	**2.71**	2.77	-
	STOI	**0.86**	**0.86**	0.77	0.78	**0.79**	0.65	**0.68**	**0.68**	0.54
	PESQ	**2.56**	2.51	1.99	2.20	**2.23**	1.62	**1.79**	1.77	1.29

for this subband,

$$f^{(i)}(\boldsymbol{x}) = \sum_{j=1}^{n} \alpha_j^{(i)} k^{(i)}(\boldsymbol{x}, \boldsymbol{x}_j). \tag{6.5}$$

Our framework starts by splitting the training targets Y_{train} into subband targets $Y^{(1)}, \cdots, Y^{(b)}$. For training data related to the i-th subband $(X_{train}, Y^{(i)})$, we perform fast and automatic kernel parameter selection using *autotune* (Algorithm 1) on its subsamples, which selects one exponential power kernel $k^{(i)}$ for this subband. We then train a kernel model on $(X_{train}, Y^{(i)})$ with kernel $k^{(i)}$ using EigenPro iteration proposed in [83]. It learns an approximate solution $\alpha^{(i)}$ (or $f^{(i)}$) for the optimization problem (6.1). Our final kernel machine is then formed by $\{f^{(1)}, \cdots, f^{(b)}\}$.

For any unseen data $\boldsymbol{x}$, our kernel machine first computes estimated mask $f^{(i)}(\boldsymbol{x})$ for each subband. Then it combines the results of $\{f^{(1)}(\boldsymbol{x}), \cdots, f^{(b)}(\boldsymbol{x})\}$ to obtain the estimated mask for all frequency channels. Applying this mask to the noisy speech produces the estimated clean speech.

6.3 Experimental Results

We use kernel machines with 4 subbands (block of frequencies) for speech enhancement. For fair comparison, we train both kernel machines and DNNs from scratch using the same features and targets. We halt the training for any model when error on validation set stops decreasing. Experiments are run on a server with 128GB main memory, two Intel Xeon(R) E5-2620 CPUs, and one GTX Titan Xp (Pascal) GPU.

6.3.1 Regression Task

We compare kernel machines and DNNs on a speech enhancement task described in [130] which is based on TIMIT corpus [36] and uses real-valued masks (IRM). We follow the description in [130] for data preprocessing and DNN construction/training. We consider five background noises: SSN, babble, a factory noise (factory1), a destroyer engine room (engine), and an operation room noise (oproom). Every noise is mixed to speech at $-5, 0, 5$dB Signal-Noise-Ratio (SNR).

Table 6.1 reports the MSE, STOI, and PESQ on test set for kernel machines and DNNs. We also present the STOI and PESQ of the noisy speech without enhancement. For all noise settings, we see that kernel machines consistently produce better MSE, in many cases significantly lower than that from DNNs, which is also the training objective for both models. We also see that STOI and PESQ of kernel machines are consistently better than or comparable to that from DNNs with only one exception (Factory1 0dB).

6.3.2 Classification Task

We train kernel machines and DNNs for a speech enhancement task in [47] which is based on HINT dataset and adopts binary masks (IBM) as targets. We follow the same procedure described in [47] to preprocess the data and construct/train DNNs. Specifically, we use two background noises, SSN and multi-talker babble. SSN is mixed to speech at -2, -5, -8dB SNR, and babble is mixed to speech at $0, -2, -5$dB SNR. As our kernel machine is designed for regression task, we use a threshold 0.5 to map its real-value prediction to binary target $\{0, 1\}$.

Table 6.2. Kernel & DNN on HINT

Metrics	Model	Babble			SSN		
		0dB	-2dB	-5dB	-2dB	-5dB	-8dB
Acc	DNN	0.90	0.91	0.90	0.91	**0.91**	**0.92**
	Kernel	**0.92**	**0.92**	**0.91**	**0.92**	0.90	0.89
STOI	DNN	0.83	0.80	0.76	0.79	**0.76**	**0.74**
	Kernel	**0.86**	**0.83**	**0.78**	**0.81**	0.75	0.71

In Table 6.2, we compare the classification accuracy (Acc) and STOI of kernel machine and DNNs under different noise settings. We see that our kernel machines outperform DNNs on noise settings with babble and perform worse than DNN on noise settings with SSN. In all, the proposed kernel machines match the performance of DNNs on this classification task.

6.3.3 Single Kernel and Subband Adaptive Kernels

We start by analyzing the performance of kernel machines that use a single kernel for all frequency channels on the regression task in Section 6.3.1. The training of such kernel machine (1 subband) is significantly faster than that of our default kernel machine (4 subbands). Remarkably, its performance is also quite competitive. It consistently outperforms DNNs in MSE in all noise settings. In 8 out of 15 noise settings, it produces STOI the same as that from the kernel machine with 4 subbands (it also produces nearly same PESQ).

Table 6.3. Comparison of kernel machines with 1 subband and 4 subbands

Noise setting	Metrics	Kernel (1 subband)	Kernel (4 subbands)	DNN
SSN 0dB	MSE	1.60	**1.48**	1.67
	STOI	0.81	**0.82**	**0.82**
	PESQ	2.35	**2.36**	2.32
SSN -5dB	MSE	1.67	**1.60**	1.76
	STOI	0.73	**0.74**	**0.74**
	PESQ	2.01	**2.03**	2.00
Factory1 -5dB	MSE	2.76	**2.71**	2.77
	STOI	0.67	**0.68**	**0.68**
	PESQ	1.78	**1.79**	1.77

However, in other noise settings, kernel (1 subband) has smaller training loss (MSE) than DNNs, but no better STOI (we show three cases in Table 6.3) [76, 143]. To improve desired metrics (STOI/PESQ), we first compare the MSE of every frequency channel of DNNs and kernel machines.

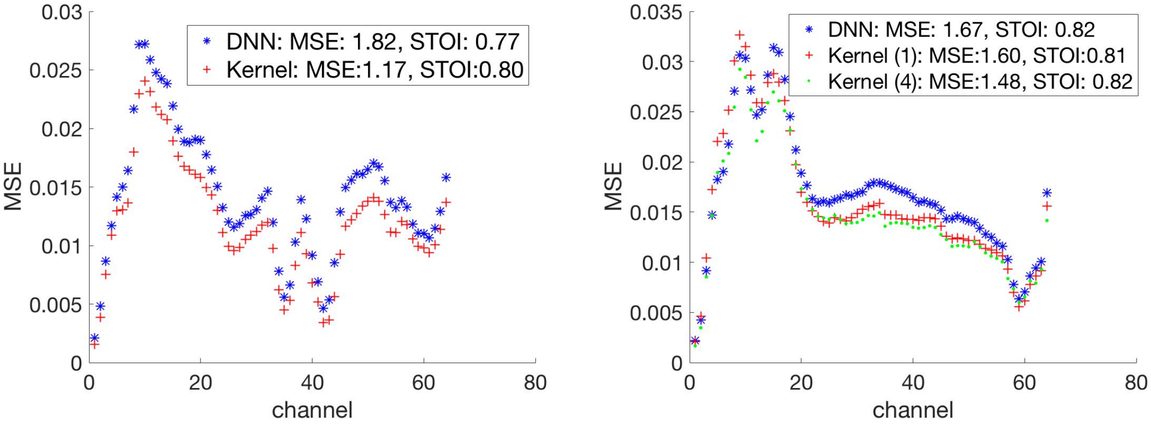


Figure 6.2. MSE along per frequency channel

As shown in the right figure of Fig. 6.2, which is for engine -5dB, for cases that kernels have much smaller overall MSE and smaller MSE on each frequency channel, kernels also achieve better STOI. For cases like SSN 0dB, as shown in the left figure of Fig. 6.2, even though single kernel (1 subband) has smaller overall MSE, its STOI is not as good as DNNs. Multiple kernels (4 subbands) decrease MSE further and also achieve better STOI. This shows that having smaller MSE along all frequency channels leads to better STOI. This reveals a correlation between MSE and STOI/PESQ associated with frequency channels.

6.3.4 Time Complexity

Table 6.4. Running time/epochs of Kernel & DNN

Dataset	Time (minutes)			Epochs	
	Kernel		DNN	Kernel	DNN
	1 subband	4 subbands			
HINT	0.8	3.2	6.6	10	50
TIMIT	18	65	124	5	93

In Table 6.4, we compare the training time of DNNs and kernel machine on both HINT and TIMIT. Note that the training of kernel machines in all experiments typically completes in no more than 10 epochs, significantly less than the number of epochs required for DNNs. Furthermore, the training time of kernel machines is also less than that of DNNs. Notably, training kernel machine with 1 subband takes much less time than DNNs.

6.4 Conclusion and Discussion

In this paper, we have shown that kernel machines using exponential power kernels show strong performance on speech enhancement problems. Notably, our method needs no parameter tuning for optimization and employs nearly automatic tuning for kernel hyper-parameter selection. Moreover, we show that the training time and computational requirement of our method are comparable or less than those needed to train neural networks. We expect that this highly efficient kernel method will be useful for other problems in speech and signal processing.

6.5 Acknowledgements

Chapter 6, in full, is a reprint of Like Hui, Siyuan Ma, and Mikhail Belkin. "Kernel Machines Beat Deep Neural Networks on Mask-based Single-channel Speech Enhancement", Interspeech 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Conclusion

Training objective is one key component in machine learning systems. The primary objective of machine learning is to optimize the model's parameters to minimize the loss function. The loss function quantifies the discrepancy between the predicted outputs of the model and the true targets. By minimizing the loss function, the model learns to make more accurate predictions and improve its performance. During the training phase, the loss function guides the learning process by measuring the model's performance. By backpropagating the gradients of the loss function, the model's parameters are updated through techniques like gradient descent. The loss function acts as a feedback signal, indicating how the model should adjust its internal parameters to reduce prediction errors.

In terms of evaluation and comparison, the loss function provides a quantitative measure of how well the model is performing. It allows for the comparison of different models or variations of a model by assessing their respective loss values. Models with lower loss values generally indicate better performance. Thus, the choice of an appropriate loss function is critical for evaluating and selecting the most suitable model for a given task. The loss function also provides feedback on model performance, helping identify areas where the model struggles or makes frequent errors. By analyzing the loss function and its gradients, researchers and practitioners can gain insights into the model's weaknesses and make targeted improvements, such as adjusting the architecture, modifying the training process, or collecting additional data.

In summary, the loss function is a fundamental component of machine learning systems. It drives the optimization process, guides the learning of the model, enables evaluation and comparison, supports regularization, aligns with task-specific objectives, and provides feedback for continuous model improvement.

Other than the training objective, model selection is another important component of machine learning systems, and understanding kernel machines is important for understanding deep models. Kernel machines, such as support vector machines (SVMs) with nonlinear kernels, allow for nonlinear transformations of the input data. Deep models also incorporate nonlinear transformations through activation functions. Understanding kernel machines helps grasp the concept of mapping data to higher-dimensional feature spaces, which is a fundamental aspect of deep models.

Kernel machines offer flexibility in capturing complex patterns and relationships in the data. Deep models, with their multiple layers and non-linear activations, are designed to learn hierarchical representations of data. Understanding kernel machines helps appreciate the ability of deep models to learn intricate structures and extract high-level features from raw data.

Kernel machines provide interpretable representations through support vectors, which are data points influencing the decision boundary. Deep models, while not as interpretable, can learn high-level representations that capture important patterns in the data. Understanding kernel machines helps grasp the notion of learning meaningful representations, which is a central aspect of deep models.

In summary, understanding kernel machines is valuable in understanding deep models as it provides insights into nonlinear transformations, flexibility in capturing complex patterns, regularization strategies, interpretable representations, architectural choices, and model selection. It enhances the overall comprehension of how deep models operate and can aid in effectively designing and analyzing deep learning architectures.

7.1 Contributions

In this thesis we give several surprising empirical phenomenon in deep learning and kernel machines. First, in [51] we do a systematic evaluation of the square loss on training modern deep classifiers and find that the square loss actually gives even better results than the widely used cross-entropy loss in the majority of our experiments. We also provide precise asymptotics of the rescaled square loss in multi-class classification.

Secondly, we propose a new loss function, which we call squentropy for multi-class classification [53]. With a wide range of experiments across NLP, speech, vision and also 121 tabular datasets, we show that the proposed squentropy loss gives better generalization and also significantly better calibration results.

The neural collapse phenomenon proposed in [101] leads to a hot research area and we also dive into it, specifically we investigate the correlation of neural collapse to generalization [52]. We provide precise definitions of training set collapse and test set collapse with both a strong and a weak version. More importantly, we show that more neural collapse does not always give better generalization in both on-distribution setting and also transfer learning setting. The conclusion is keeping some variance in the features learnt by the neural networks actually gives best test performance.

Finally, we introduce the work on kernel machines [54], where we show kernel machines with exponential power kernel and fast iteration method can achieve better test performance in speech enhancement tasks. Also, kernel machines require less computation resources and consume less time in training.

We believe our plenty empirical results would provide evidence for the effectiveness of the square loss and squentropy loss in classification. Also for the relation of neural collapse to generalization and the kernel machines in applications. Meanwhile, the empirical protocols and methodology used in those projects can also be applied to many other empirical works.

7.2 Future work

The training objective and also the model selection are fundamental questions in machine learning, and there are many interesting directions that can be done in those directions for future work.

In theory, it is important to comprehend the underlying reasons behind the improved generalization and calibration results achieved by squentropy. Several conjectures have been proposed in [53] to shed light on this matter, such as the presence of a smoother decision boundary with a wider margin and a reduction in weight norm. However, there is a lack of rigorous theoretical analysis to support these ideas. Furthermore, the correlation between neural collapse and generalization is primarily confirmed through experiments [52], and in depth theoretical analysis in this aspect is in need as well.

In practice, there is a desire to integrate squentropy into commonly used toolkits like PyTorch and scikit-learn. This integration would facilitate the adoption and implementation of squentropy, making it more accessible for individuals to experiment with.

Appendix A

A.1 Datasets and tasks

Below we provide a summary of datasets used in the experiments.

NLP tasks

- **MRPC** (Microsoft Research Paraphrase Corpus) [20] is a corpus of sentence pairs extracted from online news sources. Human annotation indicates whether the sentences in the pair are semantically equivalent. We report accuracy and F1 score.

- **SST-2** (The Stanford Sentiment Treebank) [121] is a task to determine the sentiment of a given sentence. This corpus contains sentences from movie reviews and their sentiment given by human annotations. We use only sentence-level labels, and predict positive or negative sentiment.

- **QNLI** is a converted dataset from the Stanford Question Answering Dataset [110] which consists of question-paragraph pairs. As in [127], this task is to predict whether the context sentence selected from the paragraph contains the answer to the question.

- **QQP** (Quora Question Pairs dataset) [55] contains question pairs from the question-answering website Quora. Similar to MRPC, this task is to determine whether a pair of questions are semantically equivalent. We report accuracy and F1 score.

- **text-c5** categorizes research papers to the most suitable conference. The dataset consists of 2507 short research paper titles, largely technology related and there are 5 categorizes.

- **text-c20** is the stack-overflow-data which can be found at https://www.kaggle.com/stackoverflow/stackoverflow. It is a 20-class classification task, which classifies stack overflow questions into one of the 20 tags.

- **text8** [91] originally is a language modeling task. We consider it as a classification task with the goal to classify each token of the input sentence into one of the 27 different characters.

- **enwik8** [91] is also interpreted as a classification task.

ASR tasks

- **TIMIT** [36] consists of speech from American English speakers, along with the corresponding phonemical and lexical transcription. It is widely used for acoustic-phonetic classification and ASR tasks. Its training set, validation set and test set are 3.2 hours, 0.15 hours, 0.15 hours long, respectively.

- **WSJ** (Wall Street Journal corpus) [102] contains read articles from the Wall Street Journal newspaper. Its training, validation and test set are 80 hours, 1.1 hours and 0.7 hours long, respectively.

- **Librispeech** [98] is a large-scale (1000 hours in total) corpus of 16 kHz English speech derived from audiobooks. We choose the subset train-clean-100 (100 hours) as our training data, dev-clean (2.8 hours) as our validation set and test-clean (2.8 hours) as our test set.

Vision tasks

- **MNIST** [72] contains $60,000$ training images and $10,000$ testing 28×28 pixel images of hand-written digits. It is a 10-class image classification task.

- **CIFAR-10** [68] consists of $50,000$ 32×32 pixel training images and $10,000$ 32×32 pixel test images in 10 different classes. It is a balanced dataset with $6,000$ images of each class.
- **ImageNet** [117] is an image dataset with 1000 classes, and about 1.28 million images as training set. The sizes of its validation and test set are $50,000$ and $10,000$, respectively. All images we use are in 224×224 pixels.

A.2 Hyper-parameter settings

We give the implementation toolkits and specific hyper-parameter settings to help reproduce our results, and list the epochs needed for training with the square loss and the cross-entropy (CE) loss. The data processing is following the standard methods. For NLP tasks, it is the same as in [127], and for ASR tasks, it is the same as in [133]. For vision tasks, we are following the default ones given in the implementation of the corresponding papers.

A.2.1 Hyper-parameters for NLP tasks

The implementation of BERT is based on the PyTorch toolkit [135]. The specific script we run is https://github.com/huggingface/transformers/blob/master/examples/text-classification/run_glue.py, and we use the bert-base-cased model for fine-tuning. LSTM+Attention and LSTM+CNN are implemented based on the toolkit released by [71]. The specific hyper-parameters used in the experiments are in Table A.1. As there are many hyper-parameters, we only list the key ones, and all other parameters are the default in the scripts.

A.2.2 Hyper-parameters for ASR tasks

The implementation of ASR tasks is based on the ESPnet [133] toolkit, and the specific code we use is the run.sh script under the base folder of each task, which is https://github.com/espnet/espnet/tree/master/egs/?/asr1, where ’?’ can be ’timit’, ’wsj’, and ’librispeech’. The specific hyper-parameters are following the ones in the configuration file of each task, which

Table A.1. Hyper-parameters for NLP tasks

Model	Task	Batch size	max_seq length	Learning rate w/		Epochs training w/	
				square loss	CE	square loss	CE
BERT	MRPC	32	128	5e-5	2e-5	5	3
	SST-2	32	128	2e-5	2e-5	3	3
	QNLI	32	128	2e-5	2e-5	3	3
	QQP	32	128	2e-5	2e-5	3	3
	text5	32	128	2e-5	2e-5	3	3
	text20	32	128	2e-5	2e-5	3	3
Transformer-XL	text8	8	70	2.5e-4	2.5e-4	400000♮	400000♮
	enwik8	8	70	2.5e-5	2.5e-4	400000♮	400000♮
LSTM+Attention	MRPC	64	80	2e-4	1e-4	25	20
	QNLI	32	$sent\_len^*$	1e-4	1e-4	20	20
	QQP	64	120	1e-4	1e-4	30	30
LSTM+CNN	MRPC	64	80	2e-4	1e-4	20	20
	QNLI	32	$sent\_len^*$	8e-5	1e-4	20	20
	QQP	32	120	1e-3	1e-3	20	20

* The max sequence length equals the max sentence length of the training set. ♮ training steps.

is under the base folder. We list the files which give the hyper-parameter settings for acoustic model training in Table A.2.

Table A.2. Hyper-parameters for ASR tasks

Model	Task	Hyper-parameters	Epochs training w/	
			square loss	CE
Attention+CTC	TIMIT	conf/train.yaml♮	20	20
VGG+BLSTMP	WSJ*	conf/tuning/train_rnn.yaml	15	15
VGG+BLSTM	Librispeech	conf/tuning/train_rnn.yaml◇	30	20
Transformer	WSJ	conf/tuning/train_pytorch_transformer.yaml	100	100
Transformer	Librispeech	conf/tuning/train_pytorch_transformer.yaml	120	100

* For WSJ, we use the language model given by https://drive.google.com/open?id=1Az-4H25uwnEFa4lENc-EKiPaWXaijcJp. ♮ We set mtlalpha=0.3, batch-size=30. ◇ We set elayers=4, as we use 100 hours training data.

A.2.3 Hyper-parameters for vision tasks

The implementation of these models are based on the open source toolkits. For TCNN and EfficientNet, we use the open source implementation given by [3] and [124], respectively. For Wide ResNet, we are based on the open source PyTorch implementation https://github.com/xternalz/WideResNet-pytorch (W-ResNet). For ResNet-50, our experiments are based on the

Tensorflow toolkit https://github.com/tensorflow/tpu/tree/master/models/official/resnet (ResNet) implemented on TPU. The hyper-parameter settings for our vision experiments are in Table C.1.

Table A.3. Hyper-parameters for vision tasks

Model	Task	Hyper-parameters	Epochs training w/	
			square loss	CE
TCNN	MNIST♮	the default in [3]	20	20
Wide-ResNet	CIFAR-10	the default in W-ResNet, except wide-factor=20	200	200
Visual Transformer	CIFAR-10	the default in [64]	200	200
ResNet-50	ImageNet	the default in ResNet, for square loss, learning rate=0.3	168885*	112590*
EfficientNet	ImageNet	the default in EfficientNet-B0 of [124]	218949*	218949*

♮ We are doing the permuted MNIST task as in [3].
* We give the training steps as in the original implementations.

A.3 Experimental results on validation and training sets

Table A.4. NLP results on validation set, accuracy

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
BERT [17]	MRPC	**85.3**	85.0	**85.3**
	SST-2	91.2	**91.5**	91.2
	QNLI	**90.8**	90.7	**90.8**
	QQP	**90.8**	90.7	90.6
	text5	**80.8**	80.6	**80.8**
	text20	**85.9**	85.4	**85.9**
Transformer-XL [14]	text8	**73.4**	72.9	**73.4**
	enwik8	77.0	**77.8**	77.0
LSTM+Attention [10]	MRPC	**76.5**	74.8	75.3
	QNLI	**79.7**	**79.7**	**79.7**
	QQP	**86.0**	85.5	**86.0**
LSTM+CNN [45]	MRPC	**76.0**	73.3	**76.0**
	QNLI	**76.8**	**76.8**	**76.8**
	QQP	84.0	**85.3**	84.0

We report the results for validation set of NLP tasks in Table A.4 for accuracy and Table A.5 for F1 scores.

Table A.5. NLP results on validation set, F1 scores

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
BERT	MRPC	89.5	**89.6**	89.5
[17]	QQP	**87.5**	87.4	87.4
LSTM+Attention	MRPC	**83.7**	83.3	83.5
[10]	QQP	**82.1**	81.7	**82.1**
LSTM+CNN	MRPC	**82.6**	81.4	**82.6**
[45]	QQP	77.4	**80.2**	77.4

The validation set results of the ASR tasks are in Table A.6.

Table A.6. ASR results on validation set, error rate

Model	Task	train with square loss (%)	train with cross-entropy (%)	square loss w/ same epochs as CE (%)
Attention+CTC	TIMIT (PER)	**18.1**	18.3	**18.1**
[62]	TIMIT (CER)	**30.4**	31.4	**30.4**
VGG+BLSTMP	WSJ (WER)	**8.5**	8.8	**8.5**
[88]	WSJ (CER)	**3.9**	4.0	**3.9**
VGG+BLSTM	Librispeech (WER)	**9.3**	10.7	9.9
[88]	Librispeech (CER)	**9.4**	11.1	10.2
Transformer	WSJ (WER)	**9.1**	9.3	**9.1**
[133]	Librispeech (WER)	9.7	**9.1**	9.7

We report the training result for NLP tasks in Table A.7 for accuracy and F1 score in Table A.8. The training results for ASR tasks and vision tasks are in Table A.9 and Table A.10, respectively.

A.4 Our results compared with the original work

We list our results for the models trained with the cross-entropy (CE) loss and compare them to the results reported in the literature or the toolkits in Table A.11. As we observe, our results are comparable to the original reported results.

The models marked with 'N/A' in Table A.11 do not have comparable results reported in the literature. Specifically, LSTM+Attention and LSTM+CNN models for NLP tasks are implemented based on the toolkit released by [71], where they did not show results on MRPC

Table A.7. NLP results on training and test set, accuracy

Model	Task	train with square loss (%)		train with cross-entropy (%)		square loss w/ same epochs as CE (%)	
		Train	Test	Train	Test	Train	Test
BERT [17]	MRPC	99.7	83.8	99.9	82.1	99.6	83.6
	SST-2	98.6	94.0	99.2	93.9	98.6	93.9
	QNLI	98.0	90.6	97.5	90.6	98.0	90.6
	QQP	96.2	88.9	98.0	88.9	96.2	88.8
	text5	96.7	80.6	96.3	80.5	96.7	80.6
	text20	95.6	85.6	94.9	85.2	95.6	85.6
Transformer-XL [14]	text8	90.5	73.2	90.1	72.8	90.5	73.2
	enwik8	90.7	76.7	91.8	77.5	90.7	76.7
LSTM+Attention [10]	MRPC	94.6	71.7	84.9	70.9	93.2	71.5
	QNLI	87.7	79.3	90.8	79.0	87.7	79.3
	QQP	93.7	83.4	91.5	83.1	93.7	83.4
LSTM+CNN [45]	MRPC	98.3	73.2	92.5	69.4	98.3	72.5
	QNLI	92.8	76.0	90.7	76.0	92.8	76.0
	QQP	91.3	84.3	95.7	84.4	91.3	84.3

Table A.8. NLP results on training and test set, F1 scores

Model	Task	train with square loss (%)		train with cross-entropy (%)		square loss w/ same epochs as CE (%)	
		Train	Test	Train	Test	Train	Test
BERT [17]	MRPC	99.8	88.1	99.9	86.7	99.7	88.0
	QQP	94.5	70.9	97.2	70.7	94.5	70.7
LSTM+Attention [10]	MRPC	96.1	80.9	89.5	80.6	94.7	80.7
	QQP	91.9	62.6	89.2	62.3	91.9	62.6
LSTM+CNN [45]	MRPC	98.8	81.0	94.5	78.2	98.8	81.0
	QQP	88.0	60.3	94.2	60.5	88.0	60.3

and QNLI. The QQP results are not comparable with ours as they were using a different test set, while we are using the standard test set same as in [127]. The VGG+BLSTM model for Librispeech dataset is based on ESPnet toolkit [133]. Due to computational resources limitations, we only use train-clean-100 (100 hours) as training data and 1000 unigram based dictionary for acoustic model training, while they use 1000 hours of training data with at least 2000 unigram dictionary.

Table A.9. ASR results on training and test set, error rate

Model	Task	train with square loss (%)		train with cross-entropy (%)		square loss w/ same epochs as CE (%)	
		Train	Test	Train	Test	Train	Test
Attention+CTC	TIMIT (PER)	0.9	20.8	4.8	20.8	0.9	20.8
[62]	TIMIT (CER)	4.5	32.5	11.6	33.4	4.5	32.5
VGG+BLSTMP	WSJ (WER)*	0.7	5.1	0.3	5.3	0.7	5.1
[88]	WSJ (CER)*	0.3	2.4	0.1	2.5	0.3	2.4
VGG+BLSTM	Librispeech (WER)*	0.8	9.8	0.4	10.6	0.8	10.3
[88]	Librispeech (CER)*	0.6	9.7	0.3	10.7	0.6	10.2
Transformer	WSJ (WER)*	0.7	5.7	0.5	5.8	0.7	5.7
[133]	Librispeech (WER)*	0.9	9.4	1.2	9.2	0.9	9.4

* For WSJ and Librispeech, we take 10% of the training set for the evaluation of the training error rate.

Table A.10. Vision results on training and test set, accuracy

Model	Task	train with square loss (%)		train with cross-entropy (%)		square loss w/ same epochs as CE (%)	
		Train	Test	Train	Test	Train	Test
TCNN [3]	MNIST (acc.)	98.3	97.7	99.5	97.7	98.3	97.7
W-Resnet [141]	CIFAR-10 (acc.)	100.0	95.9	100.0	96.3	100.0	95.9
Visual Transformer [64]	CIFAR-10 (acc.)	100.0	99.3	100.0	99.2	100.0	99.3
ResNet-50	ImageNet (acc.)	77.7	76.2	80.5	76.1	77.7	76.0
[46]	ImageNet (Top-5 acc.)	93.2	93.0	93.4	93.0	93.2	92.9
EfficientNet	ImageNet (acc.)	75.1	74.6	81.4	77.0	75.1	74.6
[124]	ImageNet (Top-5 acc.)	93.0	92.7	94.0	93.3	93.0	92.7

A.5 Regularization terms

We give the regularization term of each task in Table A.12. 0 means we didn't add regularization term. For WSJ, check the details at line 306 of https://github.com/espnet/espnet/blob/master/espnet/nets/pytorch_backend/rnn/decoders.py.

A.6 Variance of accuracy among different random seeds

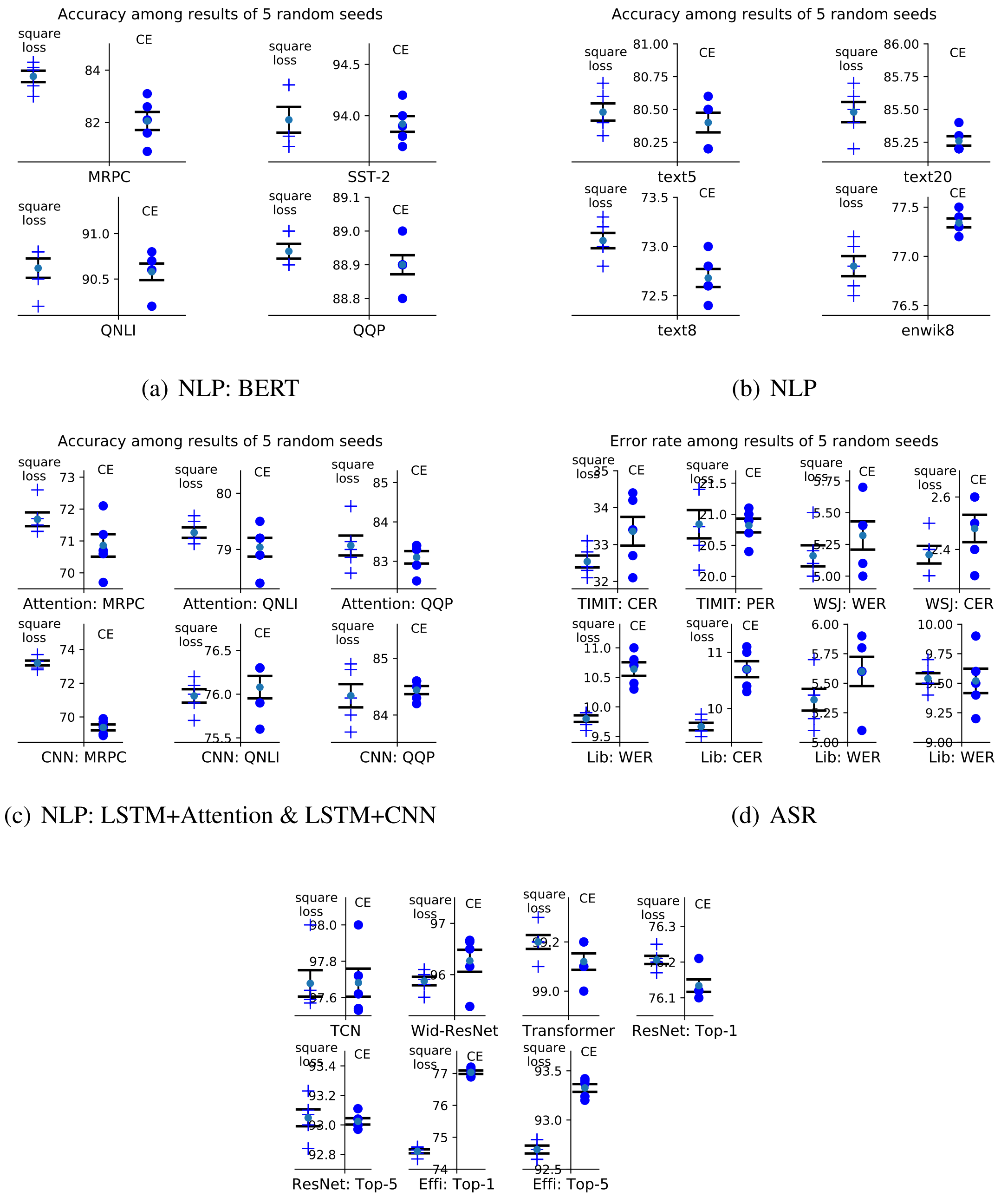
Figure A.1 gives the error bar of 5 runs corresponding to 5 different random seeds, along with the results for each inidividual run. In the left of each subfigure is the result of training with the square loss, while in the right is result of the cross-entropy. As can be seen in Figure A.1, using the square loss has better accuray/error rate and smaller variance in NLP and ASR tasks,

Table A.11. Training with the cross-entropy loss, our results and the reported ones

Model	Task	Our CE result	CE result in the literature
BERT*	MRPC (acc./F1)	85.0/89.6	85.29/89.47 [135]
	SST-2 (acc.)	91.5	91.97 [135]
	QNLI (acc.)	90.7	87.46 [135]
	QQP (acc./F1)	90.7/87.4	88.40/84.31 [135]
	text5 (acc.)	80.5	N/A
	text20 (acc.)	85.2	N/A
Transformer-XL			N/A
LSTM+Attention			N/A
LSTM+CNN			N/A
Attention+CTC	TIMIT (PER)	20.7	20.5 [133]
	TIMIT (CER)	32.7	33.7 [133]
VGG+BLSTMP	WSJ (WER)	5.4	5.3 [133]
	WSJ (CER)	2.6	2.4 [133]
VGG+BLSTM	Librispeech (WER)	10.8	N/A
	Librispeech (CER)	11.0	N/A
Transformer	WSJ (WER)	5.8	5.6
	Librispeech (WER)	9.2	N/A
TCNN	MNIST (acc.)	98.0	97.2 [3]
Wide-ResNet	CIFAR-10 (acc.)	96.5	96.11 [141]
Visual Transformer	CIFAR-10 (acc.)	99.2	99.16 [64]
ResNet-50	ImageNet (acc./Top-5 acc.)	76.1/93.0	76.0/93.0 [124]
EfficientNet	ImageNet (acc./Top-5 acc.)	77.2/93.4	77.3/93.5 [124]

* The implementation in [135] is using bert-base-uncased model, we are using bert-base-cased, which will result in a little difference. Also, as they didn't give test set results, here for BERT, we give the results of validation set.

which indicates that training with the square loss for those classification tasks is statistically better.



(a) NLP: BERT

(b) NLP

(c) NLP: LSTM+Attention & LSTM+CNN

(d) ASR

(e) Vision

Figure A.1. Accuracy/error rate variance of results among 5 random seeds

Table A.12. Regularization term for each task

Model	Task	dropout*	batch norm	Regularization Term
BERT	MRPC/SST-2/QNLI/QQP text5/text20	0.1	N	0
Transformer-XL	text8/enwik8	0.1	N	0
LSTM+Attention	MRPC/QNLI/QQP	0.5	N	0
LSTM+CNN	MRPC/QNLI/QQP	0.0	N	0
Attention+CTC	TIMIT	0.0	N	0
VGG+BLSTMP	WSJ	0.0	N	label smoothing based
VGG+BLSTM	Librispeech	0.2	N	0
Transformer	WSJ/Librispeech	0.1	N	0
TCN	MNIST	0.05	N	0
Wide-ResNet	CIFAR-10	0.0	N	0
Visual Transformer	CIFAR-10	0.0	N	0
ResNet-50	ImageNet	0.0	Y	$\frac{10^{-4}}{2}\sum_{i=1}^{n}\boldsymbol{w_i^2}$
EfficientNet	ImageNet	0.0	Y	$\frac{10^{-5}}{2}\sum_{i=1}^{n}\boldsymbol{w_i^2}$

* For dropout, 0.0 means have not apply dropout.

Appendix B

B.1 Proof of Lemma 4

Proof. To be concise in notation, in the proof, we denote $p_{ij} = p$ and $\gamma^+ = \gamma$

Case 1 $y_{ij} = 1$

$$\min_z f(z) = \|R - (z)_+\|^2 + \gamma\|z - p\|^2 = \begin{cases} (R-z)^2 + \gamma(z-p)^2 & \text{if } z > 0 \\ R^2 + \gamma(z-p)^2 & \text{if } z < 0 \end{cases}$$

There are two local minima at $z_1 = \frac{R+\gamma p}{1+\gamma}$ and $z_2 = p$. Then

$$\min_{z_1} f(z_1) = \begin{cases} \frac{\gamma}{1+\gamma}(R-p)^2 & \text{if } p > -\frac{R}{\gamma} \\ R^2 + \frac{\gamma}{(1+\gamma)^2}(R-p)^2 & \text{if } p < -\frac{R}{\gamma} \end{cases}$$

$$\min_{z_2} f(z_2) = \begin{cases} (R-p)^2 & \text{if } p > 0 \\ R^2 & \text{if } p < 0 \end{cases}$$

When $-\frac{R}{\gamma} < p < 0$, need to compare R^2 and $\frac{\gamma}{1+\gamma}(R-p)^2$ and see which one is smaller. With some calculation, when $p = p_\star$, the two is equal. Then

$$\min_z f(z) = \begin{cases} (R-p)^2 & \text{if } p > 0 \\ R^2 & \text{if } p_\star < p < 0 \\ \frac{\gamma}{1+\gamma}(R-p)^2 & \text{if } -\frac{R}{\gamma} < p < p_\star \\ R^2 + \frac{\gamma}{(1+\gamma)^2}(R-p)^2 & \text{if } p < -\frac{R}{\gamma} \end{cases}$$

Hence,

$$g(p,\gamma) = \text{arg } \min_z f(z) = \begin{cases} \frac{R+\gamma p}{1+\gamma} & p > p_\star \\ p & \text{otherwise} \end{cases}$$

Case 2 $y_{ij} = 0$, similarly,

$$\min_z f(z) = \|(z)_+\|^2 + \gamma \|z-p\|^2 = \begin{cases} z^2 + \gamma(z-p)^2 & \text{if } z > 0 \\ \gamma(z-p)^2 & \text{if } z < 0 \end{cases}$$

There are two local minima at $z_3 = \frac{\gamma p}{1+\gamma}$ and $z_4 = p$. Then

$$\min_{z_3} f(z_3) = \begin{cases} \frac{\gamma p^2}{1+\gamma} & \text{if } p > 0 \\ \frac{\gamma p^2}{(1+\gamma)^2} & \text{if } p < 0 \end{cases}$$

$$\min_{z_4} f(z_4) = \begin{cases} p^2 & \text{if } p > 0 \\ 0 & \text{if } p < 0 \end{cases}$$

$$\min_z f(z) = \begin{cases} \frac{\gamma p^2}{1+\gamma} & \text{if } p > 0 \\ 0 & \text{if } p < 0 \end{cases}$$

Hence,

$$g(p,\gamma) = \arg\min_z f(z) = \begin{cases} \frac{\gamma p}{1+\gamma} & \text{if } p > 0 \\ p & \text{if } p < 0 \end{cases}$$

□

B.2 Stieltjes Transform

For a non-negative random variable S^2, define the Stieltjes transform as the function

$$\mathscr{T}_1(u) = \mathscr{T}(u) := \mathbb{E}\frac{u}{S^2+u} \qquad u > 0 \tag{B.1}$$

We state a few quantities related to this transform are given below.

If $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ and $x_{ij} \sim \mathscr{N}(0,\sigma_{tr}^2)$ i.i.d., and $\boldsymbol{X} = \boldsymbol{USV}^\top$ with $\beta = \lim_{n\to\infty}\frac{n}{p}$. S^2 obeys the M-P law and the Stieltjes transform of the M-P law is

$$S(-z,\beta) = \frac{-(1-\beta+z)+\sqrt{(1-\beta+z)^2+4\beta z}}{2\beta z} \tag{B.2}$$

$$\mathbb{E}\frac{u}{S^2+u} = u\mathbb{E}\frac{1}{S^2+u} = u\int_a^b \frac{1}{S^2+u}dMP(S^2) = uS(-u,\beta) \tag{B.3}$$

1. $\mathscr{T}(u) = \frac{-(1-\beta+u)+\sqrt{(1-\beta+u)^2+4\beta u}}{2\beta}$

2. $\mathscr{T}_2(u) = \mathscr{T}'(u) = \frac{1}{2\beta}(-1+\frac{1+\beta+u}{\sqrt{(1-\beta+u)^2+4\beta u}})$

3. $\mathscr{T}'' = -\frac{2}{[(1-\beta+u)^2+4\beta u]^{3/2}}$

4. $\mathscr{T}_3(u) = T(u) - uT'(u) = \frac{-(1-\beta+2u)+\sqrt{(1-\beta+u)^2+4\beta u}}{2\beta} + \frac{u(1+\beta+u)}{2\beta\sqrt{(1-\beta+u)^2+4\beta u}}$

5. $\mathscr{T}_4(u) = 1 - T(u) - uT'(u) = \frac{(1+\beta)-\sqrt{(1-\beta+u)^2+4\beta u}}{2\beta} - \frac{u(1+\beta+u)}{2\beta\sqrt{(1-\beta+u)^2+4\beta u}}$

When $u \to 0$,

$$\lim_{u\to 0^+} T(u) = \begin{cases} 0 & \beta < 1 \\ \frac{\beta-1}{\beta} & \beta > 1 \end{cases} \tag{B.4a}$$

$$\lim_{u\to 0^+} T'(u) = \begin{cases} \frac{1}{1-\beta} & \beta < 1 \\ \frac{1}{\beta(\beta-1)} & \beta > 1 \end{cases} \tag{B.4b}$$

$$\lim_{u\to 0^+} T''(u) = \begin{cases} -\frac{2}{(1-\beta)^3} & \beta < 1 \\ -\frac{2}{(\beta-1)^3} & \beta > 1 \end{cases} \tag{B.4c}$$

$$\lim_{u\to 0^+} T_3(u) = \begin{cases} 0 & \beta < 1 \\ \frac{\beta-1}{\beta} & \beta > 1 \end{cases} \tag{B.4d}$$

$$\lim_{u\to 0^+} T_4(u) = \begin{cases} 1 & \beta < 1 \\ \frac{1}{\beta} & \beta > 1 \end{cases} \tag{B.4e}$$

When u is small and $\beta < 1$,

$$\mathscr{T}(u) = \frac{1}{1-\beta}u - \frac{1}{(1-\beta)^3}u^2 + o(u^3) \tag{B.5a}$$

$$\mathscr{T}'(u) = \frac{1}{1-\beta} - \frac{2u}{(1-\beta)^3} + o(u^2) \tag{B.5b}$$

$$\mathscr{T}_3(u) = \frac{u^2}{(1-\beta)^3} + o(u^3) \tag{B.5c}$$

$$\mathscr{T}_4(u) = 1 - \frac{2}{1-\beta}u + \frac{3u^2}{(1-\beta)^3} + o(u^3) \tag{B.5d}$$

similarly when u is small and $\beta > 1$,

$$\mathscr{T}(u) = \frac{\beta-1}{\beta} + \frac{1}{\beta(\beta-1)}u - \frac{1}{(\beta-1)^3}u^2 + o(u^3) \tag{B.6a}$$

$$\mathscr{T}'(u) = \frac{1}{\beta(\beta-1)} - \frac{2u}{(\beta-1)^3} + o(u^2) \tag{B.6b}$$

$$\mathscr{T}_3(u) = \frac{\beta-1}{\beta} + \frac{u^2}{(\beta-1)^3} + o(u^3) \tag{B.6c}$$

$$\mathscr{T}_4(u) = \frac{1}{\beta} - \frac{2}{\beta(\beta-1)}u + \frac{3u^2}{(\beta-1)^3} + o(u^3) \tag{B.6d}$$

When $\beta = 1$, i.e. $p = n$ and $\sigma^2 = 1$, $S(z) = -\frac{1}{2} + \frac{1}{2}\sqrt{1-\frac{4}{z}}$, then

$$T(u) := \mathbb{E}\,\frac{u}{S^2+u} = uS(-u) = u(-\frac{1}{2} + \frac{1}{2}\sqrt{1+\frac{4}{u}}) = \frac{u}{2}(\sqrt{1+\frac{4}{u}} - 1) \tag{B.7}$$

$$T_2(u) := T'(u) = \frac{1}{2}(\sqrt{1+\frac{4}{u}} - 1) - \frac{1}{\sqrt{u^2+4u}} \tag{B.8}$$

Lemma 6. *For $u > 0$ we have*

1. $\mathscr{T}_2(u) := \mathbb{E}\,\frac{S}{S^2+u}^2 = \mathscr{T}'(u)$

2. $\mathscr{T}_3(u) := \mathbb{E}\,\frac{u}{S^2+u}^2 = \mathscr{T}(u) - u\mathscr{T}'(u)$

3. $\mathscr{T}_4(u) := \mathbb{E}\,\frac{S^2}{S^2+u}^2 = 1 - \mathscr{T}(u) - u\mathscr{T}'(u)$

Proof. 1. Observe that

$$\mathscr{T}'(u) = \mathbb{E}\,\frac{(S^2+u)\cdot 1 - 1\cdot u}{(S^2+u)^2} = \mathbb{E}\,\frac{S^2}{(S^2+u)^2}$$

2. Consider the following equalities

$$\begin{aligned}\mathscr{T}(u)+\mathscr{T}'(u) &= \mathbb{E}\left(\frac{u}{S^2+u}+\frac{S^2}{(S^2+u)^2}\right)=\mathbb{E}\,\frac{uS^2+u^2}{(S^2+u)^2}+\mathbb{E}\,\frac{S^2}{(S^2+u)^2}\\ &= (u+1)\,\mathbb{E}\,\frac{S^2}{(S^2+u)^2}+\mathbb{E}\,\frac{u^2}{(S^2+u)^2}=(u+1)\mathscr{T}'(u)+\text{LHS}\end{aligned}$$

3. Observe that

$$\mathbb{E}\,\frac{S^2}{S^2+u}^{\,2}=1-2\,\mathbb{E}\,\frac{u}{S^2+u}+\mathbb{E}\,\frac{u}{S^2+u}^{\,2}=1-2\mathscr{T}(u)+(\mathscr{T}(u)-u\mathscr{T}'(u))$$

This concludes the proof. □

Appendix C

C.1 Datasets

Datasets used in our tests include the following.

- CIFAR-100: [68] consists of 50, 000 32×32 pixel training images and 10, 000 32 × 32 pixel test images in 100 different classes. It is a balanced dataset with 6, 00 images of each class.

- SVHN: [96] is a real-world image dataset obtained from house numbers in Google Street View images and it incorporates over 600,000 digit images with labeles. It is a good choice for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting.

- STL-10: [12] is an image recognition dataset mainly for developing unsupervised feature learning as it contains many images without labels. The resolution of this dataset is 96x96 and this makes it a challenging benchmark.

See Appendix A of [51] for details of other datasets.

C.2 Hyperparameters

Detailed hyperparameter settings for CIFAR-100, SVHN, and STL-10 are shown in Table C.1. For the other tasks, we follow the exact same settings as provided in Appendix B of [51].

Table C.1. Hyper-parameters for CIFAR-100, SVHN, and STL-10.

Model	Task	Hyper-parameters	Epochs training w/		
			squentropy	square loss	CE
Wide-ResNet	CIFAR-100	lr=0.1, layer=28 wide-factor=20, batch size: 128	200	200	200
VGG	SVHN	lr=0.1 for cross-entropy lr=0.0.02 for squentropy and square loss	200	200	200
Resnet-18	STL-10	lr=0.1 for cross-entropy for squentropy and square loss lr=0.02	200	200	200

C.3 More reliability diagrams

We provide the reliability diagrams for more tasks. Note that the values given for ECE (Expected calibration error as defined in (4.2) and the smaller the better) in these plots are percentages as in Table 4.1.

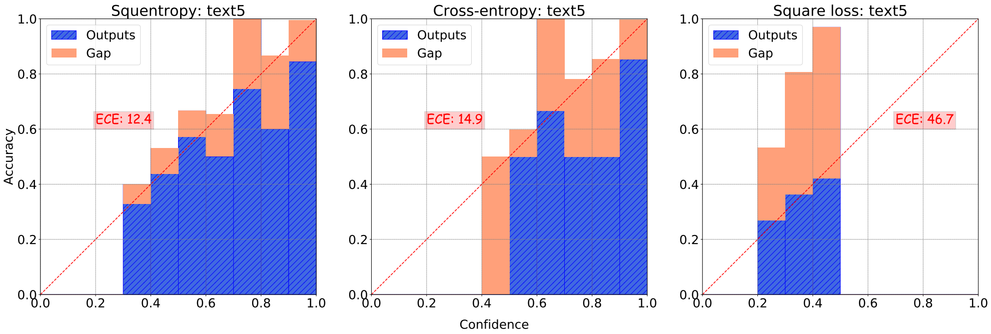


Figure C.1. Reliability diagrams for a pretrained BERT on text5 data. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

C.4 Results for 121 tabular datasets

We list the test accuracy and calibration results (ECE) of each tabular dataset in Tables C.2, C.3 and C.4. Note that the square loss of in those tables are all rescaled square loss defined in Equation (4.3). with $t = 1, M = 5$.

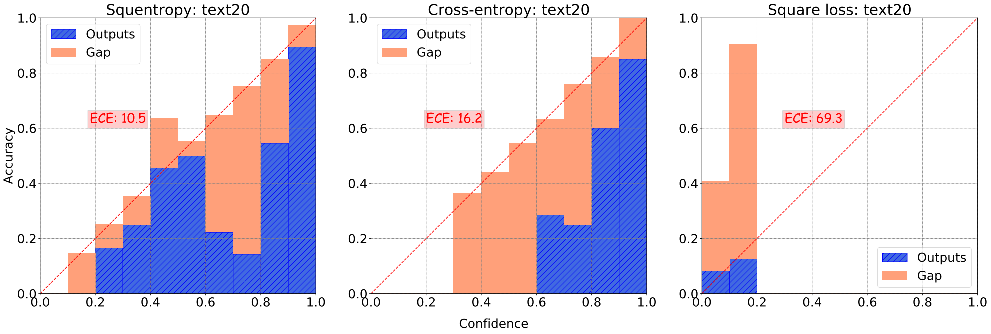


Figure C.2. Reliability diagrams for a pretrained BERT on text20 data. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

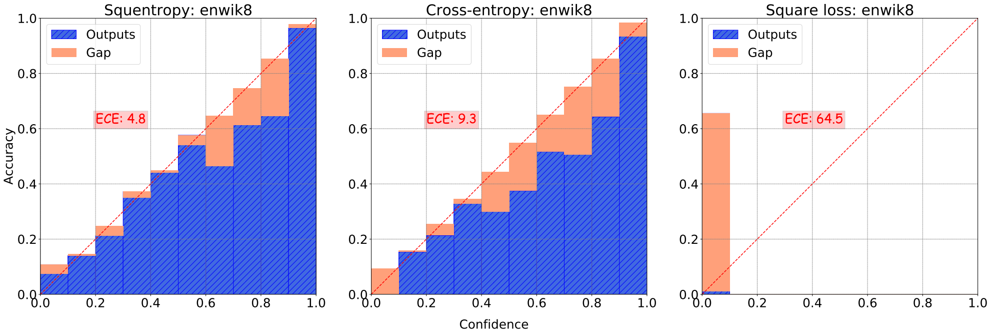


Figure C.3. Reliability diagrams for a Transformer-XL on enwik8. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

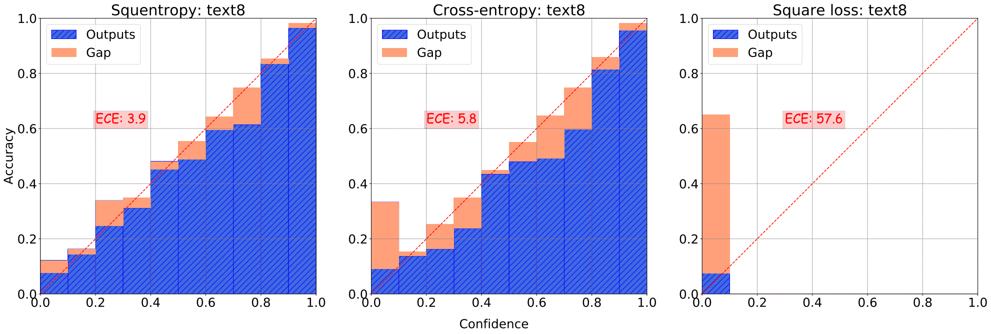


Figure C.4. Reliability diagrams for a Transformer-XL on text8. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

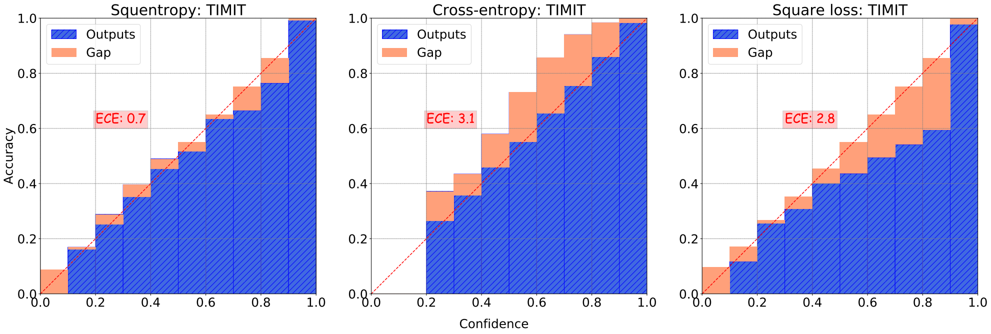


Figure C.5. Reliability diagrams for a Attention+CTC model on TIMIT. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

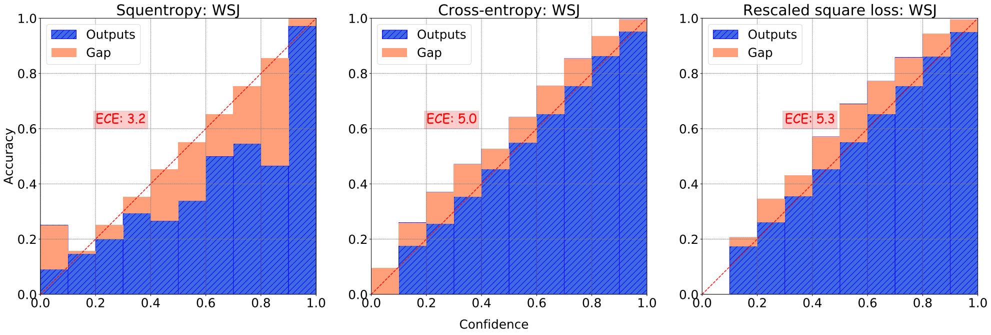


Figure C.6. Reliability diagrams for a VGG+BLSTMP model on WSJ. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss.

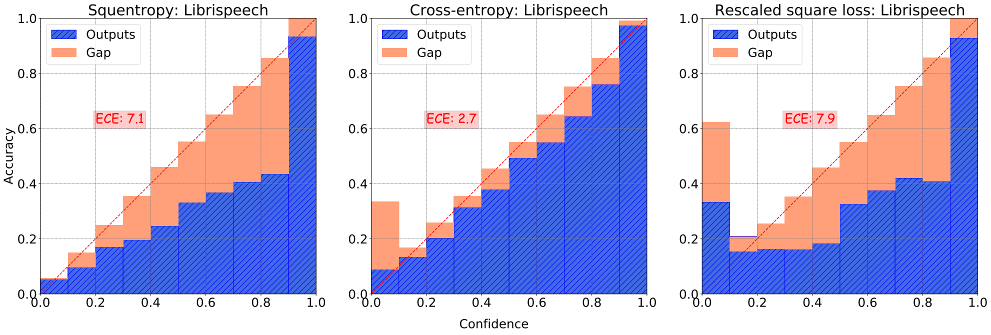


Figure C.7. Reliability diagrams for a VGG+BLSTM model on Librispeech. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss.

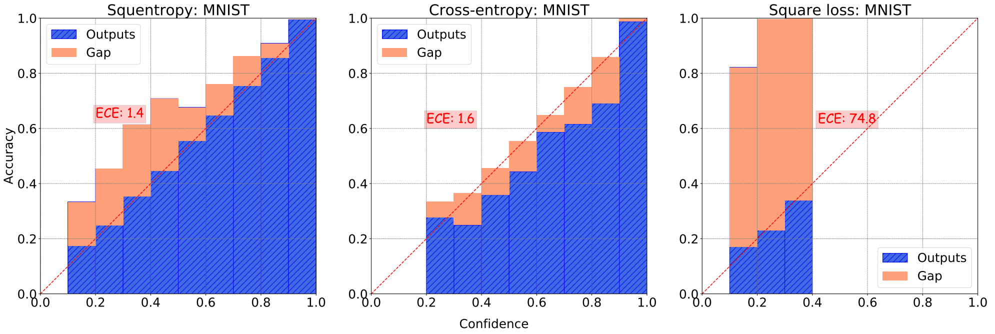


Figure C.8. Reliability diagrams for a TCN on MNIST. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

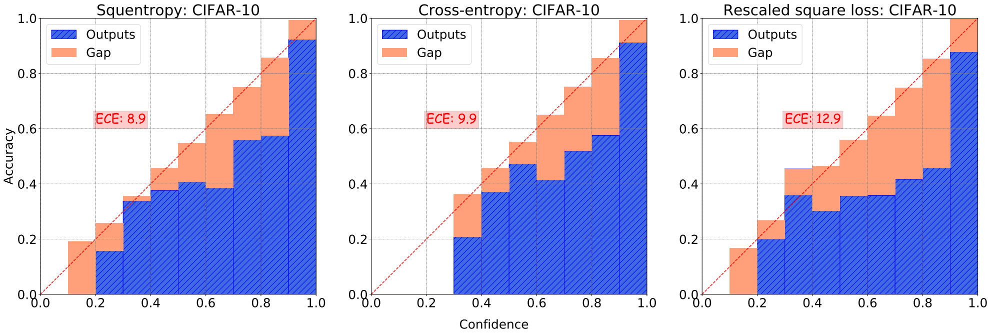


Figure C.9. Reliability diagrams for a Resnet18 on CIFAR-10.*Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss.

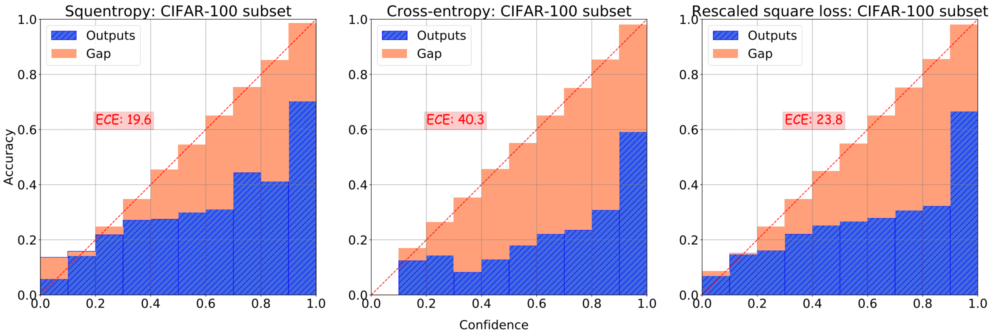


Figure C.10. Reliability diagrams for a Wide Resnet on CIFAR-100 subset. *Left:* squentropy, *middle:* cross-entropy, *right:* scaled square loss.

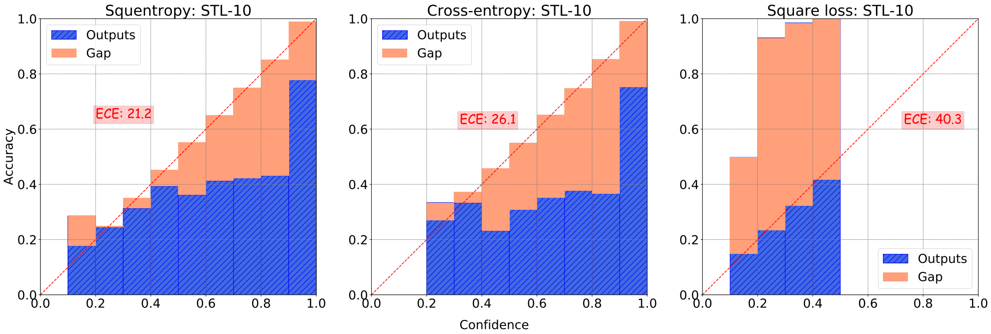


Figure C.11. Reliability diagrams for a Resnet18 on STL10. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

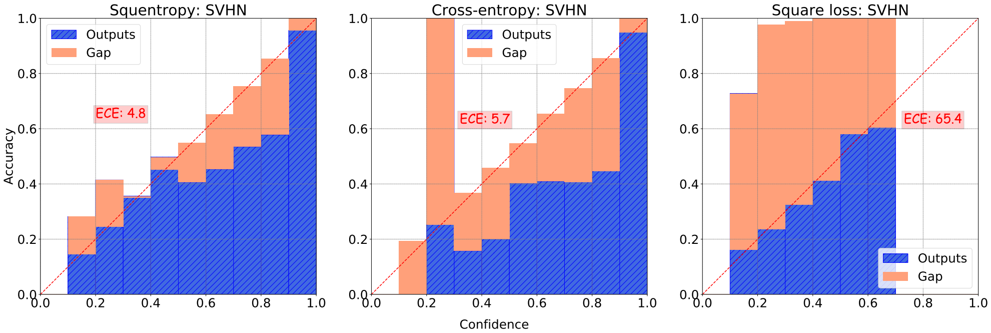


Figure C.12. Reliability diagrams for a VGG on SVHN. *Left:* squentropy, *middle:* cross-entropy, *right:* square loss.

Table C.2. Test accuracy (Acc)/ECE for 121 tabular datasets

Dataset	Squentropy		Cross-entropy		Rescaled square	
	Acc	ECE	Acc	ECE	Acc	ECE
abalone	66.0	**3.9**	67.9	14.1	**68.3**	13.8
acute-inflammation	**96.4**	**3.8**	91.3	4.7	95.8	4.3
acute-nephritis	**100.0**	**1.7**	**100.0**	3.0	**100.0**	4.8
adult	84.0	**4.7**	**85.1**	5.7	**85.1**	10.7
annealing	94.3	**3.6**	93.3	3.9	**94.4**	4.1
arrhythmia	68.1	21.4	67.0	24.5	**68.4**	**17.3**
audiology-std	72.6	26.3	**73.0**	**26.0**	70.3	29.7
balance-scale	**97.2**	5.0	96.3	**3.8**	96.5	4.0
balloons	**95.6**	23.7	95.4	**21.4**	90.0	25.8
bank	**89.9**	**4.1**	89.8	7.7	89.2	10.6
blood	81.6	11.7	**81.9**	**7.0**	81.7	16.6
breast-cancer	**76.8**	26.6	75.6	**24.5**	75.5	27.5
breast-cancer-wisc	**98.0**	4.8	97.6	4.9	97.1	**4.5**
breast-cancer-wisc-diag	**99.5**	3.5	99.2	3.4	98.8	**2.2**
breast-cancer-wisc-prog	**89.6**	16.3	87.9	**15.7**	89.0	19.3
breast-tissue	83.3	17.9	**84.1**	**17.1**	83.6	19.2
car	**100.0**	**0.4**	**100.0**	**0.4**	**100.0**	2.3
cardiotocography-10clases	87.7	6.3	**87.8**	7.3	87.2	**3.9**
cardiotocography-3clases	94.8	**4.0**	**94.9**	5.5	94.7	4.6
chess-krvk	86.6	3.7	**87.8**	**1.8**	86.1	16.0
chess-krvkp	**99.8**	**0.4**	99.7	0.5	**99.8**	0.7
congressional-voting	**65.9**	9.7	65.7	**9.2**	65.1	18.3
conn-bench-sonar-mines-rocks	90.6	11.7	**91.4**	**10.2**	**91.4**	13.1
conn-bench-vowel-deterding	**99.6**	2.7	99.1	**1.9**	98.9	9.3
connect-4	89.4	**3.3**	89.0	5.1	**89.7**	6.6
contrac	57.7	**13.6**	58.6	30.4	**58.0**	35.3
credit-approval	**89.1**	**9.4**	88.4	11.7	88.6	14.3
cylinder-bands	81.9	**13.1**	**84.1**	14.8	83.9	17.8
dermatology	**97.6**	4.7	97.2	4.4	97.3	**3.5**
echocardiogram	**85.0**	**17.6**	84.9	17.8	84.4	19.3
ecoli	**88.2**	12.4	88.1	**7.8**	**88.2**	9.8
energy-y1	97.3	4.2	**97.5**	4.2	97.3	**3.0**
energy-y2	**97.2**	4.4	96.7	5.3	96.4	**4.0**
fertility	**94.6**	23.3	93.4	26.3	90.0	**19.4**
flags	**60.1**	27.1	58.4	31.3	57.4	**20.6**
glass	**78.7**	**18.7**	78.1	25.4	78.1	20.8
haberman-survival	**80.3**	**12.2**	79.8	17.9	79.7	22.9
hayes-roth	**85.8**	**5.4**	84.1	11.4	83.7	13.6
heart-cleveland	62.5	32.0	62.1	32.5	**64.6**	**25.7**
heart-hungarian	85.3	17.6	84.8	**16.0**	**85.4**	19.2

Table C.3. Test accuracy (Acc)/ECE for 121 tabular datasets

Dataset	Squentropy		Cross-entropy		Rescaled square	
	Acc	ECE	Acc	ECE	Acc	ECE
heart-switzerland	43.8	50.4	43.6	47.7	**46.4**	**44.4**
heart-va	42.1	**46.0**	**46.4**	54.1	42.0	47.9
hepatitis	77.4	18.1	75.3	20.6	**84.5**	**14.8**
hill-valley	66.0	**14.2**	**71.9**	36.3	71.1	40.2
horse-colic	85.9	**13.3**	84.4	13.9	**86.0**	14.2
ilpd-indian-liver	**77.2**	**12.7**	75.3	22.9	75.9	25.7
image-segmentation	**96.8**	**5.4**	94.7	6.5	94.8	6.9
ionosphere	**98.3**	7.0	98.2	**5.5**	97.2	6.2
iris	97.2	3.9	97.1	4.3	**98.0**	**2.9**
led-display	75.2	10.7	**75.3**	**5.2**	74.9	8.3
lenses	76.6	20.8	68.4	21.5	**80.0**	**17.8**
letter	**98.8**	**1.1**	98.6	1.2	98.4	16.6
libras	**93.1**	**4.9**	92.9	5.7	92.5	12.9
low-res-spect	**95.9**	**4.0**	95.2	5.0	94.2	7.7
lung-cancer	60.6	**27.1**	54.7	40.4	**62.9**	40.4
lymphography	89.2	**6.4**	87.1	7.1	**91.3**	8.0
magic	88.3	5.5	89.1	**5.3**	**89.2**	8.3
mammographic	81.9	9.4	83.3	**8.0**	**83.4**	14.6
miniboone	**81.7**	**20.3**	81.5	**20.3**	77.9	27.2
molec-biol-promoter	**87.9**	11.4	78.6	**9.9**	85.5	14.8
molec-biol-splice	**87.9**	**7.1**	84.2	10.8	87.2	8.2
monks-1	85.4	14.0	83.6	**13.5**	**87.2**	14.6
monks-2	72.9	**6.9**	89.9	12.8	**95.9**	14.5
monks-3	**93.4**	**6.7**	91.6	7.6	92.0	9.4
mushroom	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	0.7
musk-1	95.2	**3.3**	94.6	6.3	**95.6**	4.9
musk-2	**100.0**	1.1	**100.0**	0.2	**100.0**	**0.7**
nursery	**100.0**	0.1	**100.0**	**0.0**	**100.0**	2.7
oocytes_merluccius_nucleus_4d	85.1	**3.5**	86.6	10.4	**87.1**	14.4
oocytes_merluccius_states_2f	**95.4**	**3.2**	95.2	6.3	95.2	4.6
oocytes_trisopterus_nucleus_2f	89.0	**5.4**	**89.7**	9.2	89.0	10.7
oocytes_trisopterus_states_5b	97.1	4.3	97.1	4.6	**97.3**	**3.7**
optical	**99.6**	**0.9**	99.3	1.2	99.0	6.6
ozone	**97.7**	4.5	97.5	3.9	97.1	**3.1**
page-blocks	**97.7**	2.2	97.5	2.3	97.1	**1.8**
parkinsons	97.0	**2.9**	**97.9**	5.8	97.4	4.2
pendigits	99.8	**0.2**	99.8	0.3	**99.9**	6.1
pima	**79.9**	**20.8**	78.0	22.4	77.4	24.6
pittsburg-bridges-MATERIAL	79.7	15.4	80.4	15.8	**89.1**	**14.0**
pittsburg-bridges-REL-L	68.2	30.2	66.2	**28.3**	**73.3**	36.5

Table C.4. Test accuracy (Acc)/ECE for 121 tabular datasets

Dataset	Squentropy		Cross-entropy		Rescaled square	
	Acc	ECE	Acc	ECE	Acc	ECE
pittsburg-bridges-SPAN	73.2	31.0	69.9	**30.8**	**73.7**	34.6
pittsburg-bridges-T-OR-D	**90.1**	19.3	90.0	24.5	89.5	**17.9**
pittsburg-bridges-TYPE	63.4	34.8	63.3	**33.0**	**66.7**	39.7
planning	**77.9**	**23.4**	75.6	33.5	76.8	31.7
plant-margin	**85.1**	**4.4**	84.0	5.9	82.9	55.7
plant-shape	**74.3**	**7.8**	73.9	13.9	70.6	49.1
plant-texture	**85.3**	**3.1**	84.3	6.2	82.6	52.8
post-operative	**73.9**	35.2	70.4	**34.7**	62.2	35.3
primary-tumor	**50.0**	27.7	49.8	38.5	48.5	**24.0**
ringnorm	**98.6**	1.7	98.5	2.1	98.1	**1.5**
seeds	**100.0**	**4.0**	99.0	6.3	98.6	5.9
semeion	**95.4**	**2.3**	94.9	3.5	94.8	10.7
soybean	**92.2**	**3.4**	90.8	3.9	91.3	17.5
spambase	**95.6**	**4.1**	95.4	4.6	95.0	4.9
spect	**76.8**	**37.3**	75.4	41.7	76.2	42.2
spectf	79.3	18.2	82.9	**17.1**	**83.8**	21.8
statlog-australian-credit	61.2	**24.1**	64.5	34.0	**66.4**	34.1
statlog-german-credit	**80.3**	**15.7**	79.1	21.2	79.6	23.1
statlog-heart	**86.9**	18.0	86.4	**15.4**	85.9	18.8
statlog-image	**99.3**	**1.4**	99.1	**1.4**	98.8	4.1
statlog-landsat	**93.3**	5.8	93.0	6.8	92.7	**2.7**
statlog-shuttle	**99.8**	**0.5**	**99.8**	**0.5**	**99.8**	3.7
statlog-vehicle	**87.5**	**7.8**	86.9	9.7	86.8	11.9
steel-plates	**78.8**	**9.7**	78.5	14.4	78.7	10.2
synthetic-control	**99.1**	2.1	**99.1**	**2.0**	98.7	4.9
teaching	**65.1**	**29.9**	63.0	30.5	63.9	37.2
thyroid	**98.5**	2.0	97.6	2.6	97.9	**1.4**
tic-tac-toe	**99.8**	0.3	**99.8**	**0.2**	**99.8**	0.6
titanic	78.6	12.6	78.4	**4.2**	**78.9**	13.6
trains	**100.0**	34.1	90.4	**27.2**	80.0	53.0
twonorm	**98.2**	**2.0**	98.1	2.6	97.7	**2.0**
vertebral-column-2clases	91.2	**8.5**	91.1	8.6	**91.3**	13.1
vertebral-column-3clases	**88.0**	15.4	86.6	**15.1**	87.1	16.1
wall-following	**96.1**	2.2	95.8	3.2	95.7	**1.7**
waveform	86.5	**9.0**	86.8	11.2	**86.9**	12.0
waveform-noise	85.4	**9.4**	85.4	11.9	**86.1**	14.1
wine	**100.0**	3.3	**100.0**	3.0	**100.0**	**2.9**
wine-quality-red	68.8	23.2	68.9	26.6	**69.3**	**19.7**
wine-quality-white	65.0	19.9	**65.9**	25.3	65.5	**17.2**
yeast	63.3	21.4	63.1	29.9	**63.5**	**18.8**
zoo	**92.0**	4.8	91.9	**3.9**	91.4	9.1

Appendix D

D.1 Experimental setup for figure 5.3

Datasets. We consider image classification tasks with MNIST [72], FashionMNIST [136], CIFAR-10 [68], SVHN [96] and STL-10 datasets [12]. SVHN was sub-sampled to $N = 4600$ samples per class as training set and $N = 1500$ samples per class for test set. Other datasets are following the standard setup. No data argumentation was done and we pre-process the images pixel-wise by subtracting the mean and dividing by the standard deviation.

Models. We train standard Resnet18 and DenseNet201 for MNIST, FashionMNIST, CIFAR10 and SVHN. Resnet50 and DenseNet201 were trained for STL10. For all datasets we also train VGG11 with batch normalization. All models were trained from scratch with open source code from torchvision models.

Optimization mechenism. We use stochastic gradient descent (SGD) with momentum 0.9 and minimize the cross-entropy loss. All tasks were trained on a single GPU with batch size 128 and 80000 SGD iterations. Initial learning rate is 0.1 for Resnet18 and Resnet50 and 0.01 for DenseNet201 and VGG architectures. We decay the learning rate with cosine annealing scheme.

D.2 Experimental setup for transfer learning

Super-class pre-training. For MNIST, we set all odd numbers as one class and all even numbers as the other class. We train the model with the first $N = 1000$ training samples as train set and the first $N = 200$ test samples as test set. For CIFAR-10, we combine samples of 'airplane, automobile, ship, truck' as one (objects) class and 'bird, cat, frog, horse' as the other (animals) class. The two classes are balanced and have 40000 training samples, and 8000 test samples. We use a subset with $N = 20000$ training samples (to keep each class balanced, we randomly choose 2500 samples from each original class) and $N = 4000$ (500 samples from each original class) test samples for pre-training. The learning rate for MNIST with fully-connected networks is 0.001 while for CIFAR-10 with ResNet18 is 0.1. We decay learning rate with cosine annealing scheme. The models were trained minimizing the cross-entropy loss using SGD with momentum 0.9 for 100000 SGD iterations.

Fine-tuning. We initialize the weights (other than the last classification layer) of the downstream task with the pre-trained weights and fine-tune the whole network. For MNIST, we do the standard 10-class classification, while we sample another 500 samples from training set for training and 100 samples from the test set for inference. For CIFAR-10 we implement a 8-class classification ('airplane, automobile, ship, truck, bird, cat, frog, horse') with another 10000 training samples as train set and another 2000 test samples as test set. The optimization methodology is the same as in pre-training, other than the learning rate. We search over 0.0005 to 0.25 in fine-tuning for both MNIST and CIFAR-10 and report the best test accuracy of all swept learning rates.

D.3 Proof of Lemma 5

Proof. As given in Lemma 5, for $(x_1, 0) \sim \mathscr{D}_2$ and $(x_2, 1) \sim \mathscr{D}_2$, there have $h(x_1) = \mu_1$ and $h(x_2) = \mu_2$, where x_1 and x_2 are samples from class 0 and class 1 correspondingly. For a test sample $x \sim \mathscr{D}$, which is sampled from the target distribution, the corresponding feature representation $h(x)$ also maps x to μ_1 or μ_2, due to the superclass property.

For any function $f : \mathbb{R}^d \to \{0, 1, ..., 2k-1\}$, which maps the feature representation

$h(x) \in \mathbb{R}^d$ of a sample x to class index $\hat{y}$,

$$\hat{y} = f(h(x)) = \begin{cases} f(\mu_1) = k_1 & \text{if } x \text{ is superclassed to class 0} \\ f(\mu_2) = k_2 & \text{if } x \text{ is superclassed to class 1} \end{cases}$$

where k_1, k_2 are class index from $\{$0,1,.., 2k-1$\}$. As we are considering the class-balanced case, for each sample $x \sim \mathscr{D}$, the probability of its true label y to be k_1 or k_2 is $Pr(y = k_1) = \frac{1}{2k}$ and $Pr(y = k_2) = \frac{1}{2k}$. Then

$$\Pr_{(x,y)\sim\mathscr{D}}[\hat{y} = y] = \begin{cases} \dfrac{1}{k} & \text{if } y = k_1 \text{ or } y = k_2 \\ 0 & \text{if } y \neq k_1 \text{ and } y \neq k_2 \end{cases}$$

Hence

$$\Pr_{(x,y)\sim\mathscr{D}}[f(h(x)) = y] \leq \frac{1}{k}$$

and the equality holds when $y = k_1$ or $y = k_2$. □

Bibliography

[1] Emmanuel Abbe and Colin Sandon. On the universality of deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20061–20072. Curran Associates, Inc., 2020.

[2] Amirhesam Abedsoltan, Mikhail Belkin, and Parthe Pandit. Toward large kernel models. *arXiv preprint arXiv:2302.02605*, 2023.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[4] Andrzej Banburski, Fernanda De La Torre, Nishka Pant, Ishana Shastri, and Tomaso Poggio. Distribution of classification margins: Are all data equal? *arXiv preprint arXiv:2107.10199*, 2021.

[5] Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing*, 400:113–136, 2020.

[6] Erin J Bredensteiner and Kristin P Bennett. Multicategory classification by support vector machines. In *Computational optimization*, pages 53–79. Springer, 1999.

[7] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[8] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler M., Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[9] Jie Chen, Lingfei Wu, Kartik Audhkhasi, Brian Kingsbury, and Bhuvana Ramabhadrari. Efficient one-vs-one kernel ridge regression for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2454–2458. IEEE, 2016.

[10] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1657–1668, 2017.

[11] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[12] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[13] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[14] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, page 2978–2988, 2019.

[15] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983.

[16] Ahmet Demirkaya, Jiasi Chen, and Samet Oymak. Exploring the role of loss functions in multiclass classification. In *2020 54th annual conference on information sciences and systems (ciss)*, pages 1–5. IEEE, 2020.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2019.

[18] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[19] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *Advances in Neural Information Processing Systems*, 33:21981–21993, 2020.

[20] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[21] David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[22] Weinan E and Stephan Wojtowytsch. On the emergence of tetrahedral symmetry in the final and penultimate layers of neural network classifiers. *arXiv preprint arXiv:2012.05420*, 2020.

[23] Michael Elad, Dror Simon, and Aviad Aberdam. Another step toward demystifying deep neural networks. *Proceedings of the National Academy of Sciences*, 117(44):27070–27072, 2020.

[24] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *Advances in neural information processing systems*, 31, 2018.

[25] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson Fletcher. Generalization error of generalized linear models in high dimensions. In *International Conference on Machine Learning*, pages 2892–2901. PMLR, 2020.

[26] Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. In *International Conference on Machine Learning*, pages 3004–3014. PMLR, 2021.

[27] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences*, 118(43), 2021.

[28] Oliver Y Feng, Ramji Venkataramanan, Cynthia Rush, and Richard J Samworth. A unifying tutorial on approximate message passing. *Foundations and Trends® in Machine Learning*, 15(4):335–536, 2022.

[29] Yutong Feng, Jianwen Jiang, Mingqian Tang, Rong Jin, and Yue Gao. Rethinking supervised pre-training for better downstream transferring. *arXiv preprint arXiv:2110.06014*, 2021.

[30] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

[31] Spencer Frei, Niladri S Chatterji, and Peter Bartlett. Benign overfitting without linearity: Neural network classifiers trained by gradient descent for noisy linear data. In *Conference on Learning Theory*, pages 2668–2703. PMLR, 2022.

[32] Krzysztof Gajowniczek, Leszek J Chmielewski, Arkadiusz Orłowski, and Tomasz Zabkowski. Generalized entropy cost function in neural networks. In *International Conference on Artificial Neural Networks*, pages 128–136. Springer, 2017.

[33] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[34] Tomer Galanti. A note on the implicit bias towards minimal depth of deep neural networks. *arXiv preprint arXiv:2202.09028*, 2022.

[35] Tomer Galanti, András György, and Marcus Hutter. On the role of neural collapse in transfer learning. *arXiv preprint arXiv:2112.15121*, 2021.

[36] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[37] BG Giraud and R Peschanski. On positive functions with positive fourier transforms. *Acta Physica Polonica B*, 37:331, 2006.

[38] Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*, pages 3607–3616. PMLR, 2020.

[39] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, volume 13, pages 1756–1760, 2013.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[41] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[42] X. Y. Han, Vardan Papyan, and David L. Donoho. Neural collapse under mse loss: Proximity to and dynamics on the central path. *ArXiv*, abs/2106.02073, 2021.

[43] XY Han, Vardan Papyan, and David L Donoho. Neural collapse under mse loss: Proximity to and dynamics on the central path. *arXiv preprint arXiv:2106.02073*, 2021.

[44] Frank E Harrell Jr. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[45] Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, 2016.

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

[47] Eric W Healy, Sarah E Yoho, Yuxuan Wang, and DeLiang Wang. An algorithm to improve speech recognition in noise for hearing-impaired listeners. *The Journal of the Acoustical Society of America*, 134(4):3029–3038, 2013.

[48] Le Hou, Chen-Ping Yu, and Dimitris Samaras. Squared earth mover's distance-based loss for training deep neural networks. *arXiv preprint arXiv:1611.05916*, 2016.

[49] Tianyang Hu, Jun Wang, Wenjia Wang, and Zhenguo Li. Understanding square loss in training overparametrized neural network classifiers. *arXiv preprint arXiv:2112.03657*, 2021.

[50] Po-Sen Huang, Haim Avron, Tara N Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on timit. In *Acoustics, Speech and Signal Processing (ICASSP), 2014*, pages 205–209, 2014.

[51] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020.

[52] Like Hui, Mikhail Belkin, and Preetum Nakkiran. Limitations of neural collapse for understanding generalization in deep learning. *arXiv preprint arXiv:2202.08384*, 2022.

[53] Like Hui, Mikhail Belkin, and Stephen Wright. Cut your losses with squentropy. *arXiv preprint arXiv:2302.03952*, 2023.

[54] Like Hui, Siyuan Ma, and Mikhail Belkin. Kernel machines beat deep neural networks on mask-based single-channel speech enhancement. *arXiv preprint arXiv:1811.02095*, 2018.

[55] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First Quora Dataset Release: Question Pairs. In *https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs*, 2017.

[56] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[57] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[58] Wenlong Ji, Yiping Lu, Yiliang Zhang, Zhun Deng, and Weijie J Su. How gradient descent separates data with neural collapse: A layer-peeled perspective. 2021.

[59] Wenlong Ji, Yiping Lu, Yiliang Zhang, Zhun Deng, and Weijie J Su. An unconstrained layer-peeled perspective on neural collapse. *arXiv preprint arXiv:2110.02796*, 2021.

[60] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798, 2019.

[61] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[62] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4835–4839. IEEE, 2017.

[63] Douglas M Kline and Victor L Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318, 2005.

[64] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[65] Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50, 2002.

[66] Aran Komatsuzaki. One epoch is all you need. *arXiv preprint arXiv:1906.06669*, 2019.

[67] Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34:28648–28662, 2021.

[68] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[69] Thomas Kuhn. *The structure of scientific revolutions*. University of Chicago Press, 1962.

[70] Himanshu Kumar and PS Sastry. Robust loss functions for learning multi-class classifiers. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 687–692. IEEE, 2018.

[71] Wuwei Lan and Wei Xu. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902, 2018.

[72] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[73] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

[74] Yunwen Lei, Urun Dogan, Alexander Binder, and Marius Kloft. Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms. *Advances in neural information processing systems*, 28, 2015.

[75] Yunwen Lei, Ürün Dogan, Ding-Xuan Zhou, and Marius Kloft. Data-dependent generalization bounds for multi-class classification. *IEEE Transactions on Information Theory*, 65(5):2995–3021, 2019.

[76] Leo Lightburn and Mike Brookes. A weighted stoi intelligibility metric based on mutual information. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5365–5369. IEEE, 2016.

[77] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[78] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European conference on computer vision*, pages 438–455. Springer, 2020.

[79] Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. The devil is in the margin: Margin-based label smoothing for network calibration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 80–88, 2022.

[80] Bruno Loureiro, Gabriele Sicuro, Cédric Gerbelot, Alessandro Pacco, Florent Krzakala, and Lenka Zdeborová. Learning gaussian mixtures with generalized linear models: Precise asymptotics in high-dimensions. *Advances in Neural Information Processing Systems*, 34:10144–10157, 2021.

[81] Jianfeng Lu and Stefan Steinerberger. Neural collapse with cross-entropy loss. *ArXiv*, abs/2012.08465, 2020.

[82] Zhiyun Lu, Dong Quo, Alireza Bagheri Garakani, Kuan Liu, Avner May, Aurélien Bellet, Linxi Fan, Michael Collins, Brian Kingsbury, Michael Picheny, and Fei Sha. A comparison between deep neural nets and kernel acoustic models for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5070–5074. IEEE, 2016.

[83] Siyuan Ma and Mikhail Belkin. Learning kernels that adapt to gpu. *arXiv preprint arXiv:1806.06144*, 2018.

[84] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[85] Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

[86] Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *arXiv preprint arXiv:2011.11619*, 2020.

[87] Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *Sampling Theory, Signal Processing, and Data Analysis*, 20(2):1–13, 2022.

[88] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Triggered attention for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5666–5670. IEEE, 2019.

[89] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

[90] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[91] LLC MultiMedia. Large text compression benchmark. 2009.

[92] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter? *The Journal of Machine Learning Research*, 22(1):10104–10172, 2021.

[93] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[94] Preetum Nakkiran. *Towards an Empirical Theory of Deep Learning*. Doctoral dissertation, Harvard University Graduate School of Arts and Sciences, 2021.

[95] Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers. In *International Conference on Learning Representations*, 2020.

[96] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[97] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[98] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[99] Ashutosh Pandey and Deliang Wang. A new framework for supervised speech enhancement in the time domain. *Proc. Interspeech 2018*, pages 1136–1140, 2018.

[100] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[101] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[102] Douglas B Paul and Janet M Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

[103] Federico Pernici, Matteo Bruni, Claudio Baecchi, and Alberto Del Bimbo. Regular polytope networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[104] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[105] Tomaso Poggio and Qianli Liao. Explicit regularization and implicit bias in deep network classifiers trained with the square loss. *arXiv preprint arXiv:2101.00072*, 2020.

[106] Tomaso Poggio and Qianli Liao. Implicit dynamic regularization in deep networks. Technical report, Center for Brains, Minds and Machines (CBMM), 2020.

[107] Qichao Que and Mikhail Belkin. Back to the future: Radial basis function networks revisited. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1375–1383, Cadiz, Spain, 2016. PMLR.

[108] Qichao Que and Mikhail Belkin. Back to the future: Radial basis function networks revisited. In *Artificial intelligence and statistics*, pages 1375–1383. PMLR, 2016.

[109] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[110] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[111] Akshay Rangamani and Andrzej Banburski-Fahey. Neural collapse in deep homogeneous classifiers and the role of weight decay. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4243–4247. IEEE, 2022.

[112] Akshay Rangamani, Mengjia Xu, Andrzej Banburski, Qianli Liao, and Tomaso Poggio. Dynamics and neural collapse in deep classifiers trained with the square loss. 2021.

[113] Sundeep Rangan, Philip Schniter, and Alyson K Fletcher. Vector approximate message passing. *IEEE Transactions on Information Theory*, 65(10):6664–6684, 2019.

[114] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[115] Ryan Michael Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, MaSSachuSettS InStitute of Technology, 2002.

[116] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pages 8242–8252. PMLR, 2020.

[117] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[118] Arash Sangari and William Sethares. Convergence analysis of two loss functions in soft-max regression. *IEEE Transactions on Signal Processing*, 64(5):1280–1288, 2015.

[119] Philip Schniter, Sundeep Rangan, and Alyson K Fletcher. Vector approximate message passing for the generalized linear model. In *2016 50th Asilomar conference on signals, systems and computers*, pages 1525–1529. IEEE, 2016.

[120] Bernhard Schölkopf, Alexander J Smola, and Francis Bach. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[121] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[122] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[123] Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

[124] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[125] Christos Thrampoulidis, Samet Oymak, and Mahdi Soltanolkotabi. Theoretical insights into multiclass classification: A high-dimensional asymptotic view. *Advances in Neural Information Processing Systems*, 33:8907–8920, 2020.

[126] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[127] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[128] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726., 2018.

[129] Ke Wang, Vidya Muthukumar, and Christos Thrampoulidis. Benign overfitting in multiclass classification: All roads lead to interpolation. *Advances in Neural Information Processing Systems*, 34:24164–24179, 2021.

[130] Yuxuan Wang, Arun Narayanan, and DeLiang Wang. On training targets for supervised speech separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(12):1849–1858, 2014.

[131] Yuxuan Wang and DeLiang Wang. Towards scaling up classification-based speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7):1381–1390, 2013.

[132] Zhong-Qiu Wang and DeLiang Wang. Recurrent deep stacking networks for supervised speech separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 71–75. IEEE, 2017.

[133] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. Espnet: End-to-end speech processing toolkit. In *Interspeech*, pages 2207–2211, 2018.

[134] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

[135] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[136] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[137] Chen Xing, Sercan Arik, Zizhao Zhang, and Tomas Pfister. Distance-based learning from errors for confidence calibration. *ICLR*, 2020.

[138] Mengjia Xu, Akshay Rangamani, Andrzej Banburski, Qianli Liao, Tomer Galanti, and Tomaso Poggio. Deep classifiers trained with the square loss. *CBMM Memo No. 117*, 2022.

[139] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(1):7–19, 2015.

[140] Yibo Yang, Liang Xie, Shixiang Chen, Xiangtai Li, Zhouchen Lin, and Dacheng Tao. Do we really need a learnable classifier at the end of deep neural network? *arXiv preprint arXiv:2203.09081*, 2022.

[141] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.

[142] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. https://d2l.ai.

[143] Hui Zhang, Xueliang Zhang, and Guanglai Gao. Training supervised speech separation system to improve stoi and pesq directly. In *Acoustics, Speech and Signal Processing (ICASSP), 2018*, pages 5374–5378, 2018.

[144] Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(5):49, 2018.

[145] Jinxin Zhou, Xiao Li, Tianyu Ding, Chong You, Qing Qu, and Zhihui Zhu. On the optimization landscape of neural collapse under mse loss: Global optimality with unconstrained features. *arXiv preprint arXiv:2203.01238*, 2022.

[146] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *arXiv preprint arXiv:2105.02375*, 2021.