Triplet Loss for Regularizing Deep Neural Networks

Drake Svoboda
Department of Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109
drakes@umich.edu

Junghwan Kim
Department of Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109
kimjhj@umich.edu

Anshul Aggarwal
Department of Statistics
University of Michigan
Ann Arbor, MI 48109
aanshul@umich.edu

Youngwoo Woo
Department of Chemical Engineering
University of Michigan
Ann Arbor, MI 48109
ywwoo@umich.edu

Abstract

We study a novel method for the regularization of deep neural networks. From the observation that maximizing margin improves generalization performance, we hypothesize that learning features that are separated by large margin in the intermediate layers of neural networks improves generalization performance. Specifically, we propose to use triplet loss on intermediate layers of neural networks. Our theoretical analysis relates triplet loss to multi-class hinge loss and to mixture of Gaussians in feature space. We also explain how this effect improves the performance of a neural network model. We empirically verify that our proposed triplet loss regularization achieves better generalization performance compared to commonly used dropout and batch normalization especially when the size of training data is small. In addition, our proposed regularization shows robustness against adversarially perturbed inputs.

1 Introduction

Deep neural networks are shown both theoretically and empirically to be capable of representing highly complex functions. The universal approximation theorem [19, 9] theoretically guarantees that neural networks approximate any reasonable continuous function. Empirically, deep neural networks have shown unprecedented performance on computer vision [25, 14], natural language processing [8, 22] and speech recognition [15] tasks. In addition, it is possible to achieve zero training error on a high-dimensional image classification task with deep neural networks [54], even when the labels are randomly shuffled.

Due to the highly expressive function class that is represented by deep neural network models, regularization has been considered essential in the training of deep neural networks. Although recent work [54, 35, 27, 12, 2] hints at the innate regularization property of deep neural networks trained by stochastic gradient descent, the gain on generalization performance from additional

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

regularization is non-negligible [54, 49]. Consequently, the common practice when training deep neural networks is to apply regularization methods such as weight decay [16, 26, 34], dropout [46] and data augmentation [25].

We propose a novel triplet loss regularization that strengthens the generalization performance of deep neural networks. Triplet loss penalizes the network when features from the same class are not closer in distance than features from different classes by some margin. When applied to the feature space of intermediate hidden layers in deep neural networks, triplet loss enforces features to have large margin between classes and small variance within classes. We further formalize this analysis by providing a novel probabilistic interpretation of triplet loss as maximizing KL divergence between each component in Gaussian Mixture Models (GMMs).

Our idea draws motivation from the max-margin classifier [33, 28]: the generalization performance, which is measured by the difference between training and test error, is bounded above by the quantity that depends inversely on the margin. Triplet loss approximately maximizes the margin between different classes without computing any batch statistics in the stochastic optimization setting. Finally, we describe how triplet loss leads to better classification performance of deep neural networks by studying information theoretic measures and learning theoretic bounds.

We verify the generalization performance of our method with experimental results. Specifically, our method shows superior generalization performance compared to other commonly used regularization methods. The performance gap increases in the small training data setting where regularization is especially important. In addition, our method also shows high robustness against adversarial attacks even though it is not explicitly designed for adversarial robustness. Figure 1 shows the visualization of feature space under each regularization method. Triplet regularization results in the feature space that separates each class most significantly.

This paper makes the following contributions:

- We propose to use triplet loss for the regularization of deep neural networks. Our usage of triplet loss as a regularization method is novel in contrast to previous works which use triplet loss for ranking or metric learning purposes.
- We provide a theoretical interpretation of triplet loss as maximizing KL divergence between each component in Gaussian Mixture Models. This interpretation provides a theoretical framework for analyzing the generalization of a heuristically designed triplet loss. Using this interpretation, we explain the effect of triplet loss on classification performance.
- We perform comprehensive experiments to verify the regularization effects and adversarial robustness of our proposed method. Our method shows superior regularization performance especially in the setting where the size of training data is small.

2 Proposed Method

We consider a feed-forward neural network that is represented as $\mathbf{h}_\ell = f_\ell(\mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell)$ for $\ell = 1, \cdots, L$ where $\mathbf{W}_\ell$ and $\mathbf{b}_\ell$ define the linear transformation and f_ℓ is the activation function for the ℓ-th hidden layer. We use the ReLU activation function $f(z) = \max(z, 0)$ in this paper. We define $\mathbf{h}_0$ and $\mathbf{h}_L$ as the input and output of the feedforward neural network, respectively. We focus our attention on the classification task where the model predicts which of K classes the input data point belongs to. In this case, we have the final feature $\mathbf{h}_L$ in $\mathbb{R}^K$ and use identity function for f_L. The prediction is made as $\arg\max_k (\mathbf{h}_L)_k$.

We define the training dataset as $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \cdots, N\}$ and the parameters of our neural network as $\mathbf{\Theta} = \{(W_\ell, b_\ell) : \ell = 1, \cdots, L\}$. In addition, $\mathbf{h}_\ell^{(i)}$ denotes the ℓ-th layer's hidden features computed from $\mathbf{x}^{(i)}$.

Motivated by the generalization guarantee of max margin training [33, 28], our proposed regularization term aims to learn a feature space that distinguish features for different classes. Specifically, we propose to apply triplet loss in the feature space of intermediate hidden layers. Triplet loss penalizes when features from the same class are not closer than features from different classes by some margin.

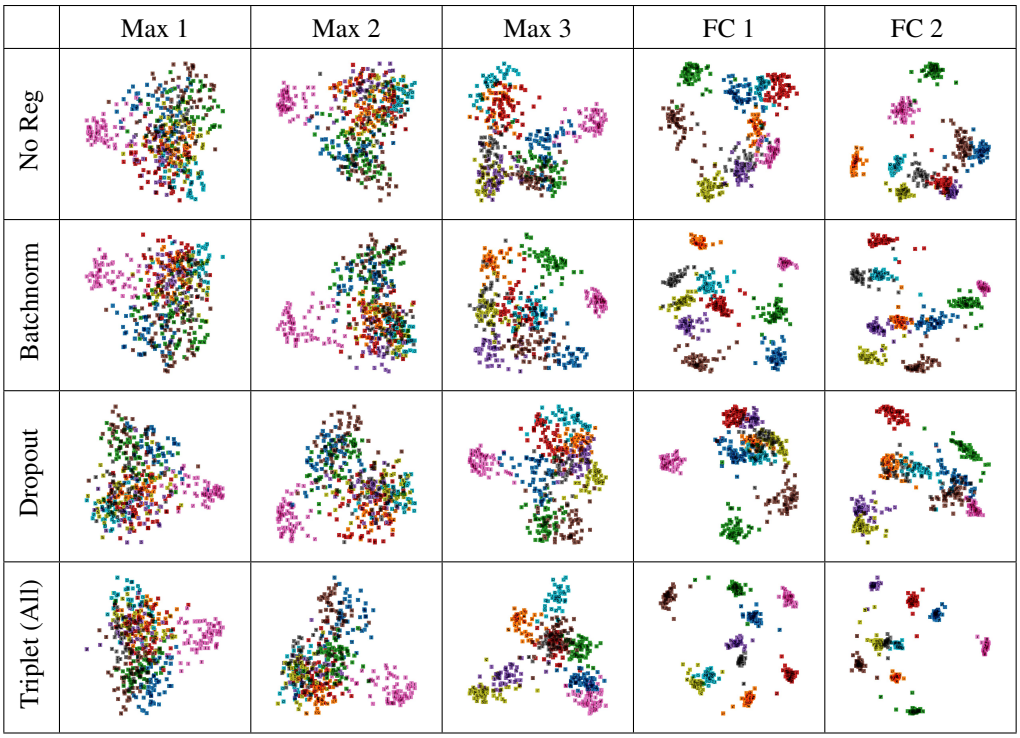


Figure 1: **Layer Visualizations.** We visualize the feature space of hidden layers in AlexNet. Hidden features are computed for 250 test examples and projected into two dimensions using PCA. Max 1, Max 2, and Max 3 correspond to each max pooling layer. FC 1 and FC 2 correspond to the final fully connected layers. Triplet regularization gives better separation in the final layers.

Formally, our triplet regularization term is defined as

$$R(\mathbf{\Theta}) = \frac{1}{|T|} \sum_{(\mathbf{x}^{(\text{anc})},\mathbf{x}^{(\text{pos})},\mathbf{x}^{(\text{neg})})\in T} \sum_{\ell \in H} \max(0, \|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(pos)}\|^2 - \|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(neg)}\|^2 + \alpha) \quad (1)$$

where $H \subseteq \{1, \cdots, L\}$ is the subset of hidden layers that we choose to apply triplet loss to and T is the set of all valid triplets in the data. A valid triplet is defined as $(\mathbf{x}^{(\text{anc})}, \mathbf{x}^{(\text{pos})}, \mathbf{x}^{(\text{neg})})$ where $\mathbf{x}^{(\text{anc})}$ and $\mathbf{x}^{(\text{pos})}$ belongs to the same class k while $\mathbf{x}^{(\text{neg})}$ corresponds to the different class, other than k. That is, $y^{(anc)} = y^{(pos)} \neq y^{(neg)}$. We refer to $\mathbf{x}^{(\text{anc})}$, $\mathbf{x}^{(\text{pos})}$ and $\mathbf{x}^{(\text{neg})}$ as an anchor example, a positive example and a negative example, respectively. α is a hyperparameter that represents the desired margin.

Our regularization term is introduced into the objective function as

$$J(\mathbf{\Theta}) = L(\mathbf{\Theta}) + \lambda R(\mathbf{\Theta}) \quad (2)$$

where $L(\mathbf{\Theta})$ is any valid multi-class classification loss function (e.g. cross entropy loss, hinge loss). The $L(\mathbf{\Theta})$ term will enforce that the network makes accurate predictions and the $R(\mathbf{\Theta})$ term will enforce that selected hidden layers will have large inter-class margins and low intra-class variance.

In this paper, we specifically use multi-class hinge loss that is defined as

$$L(\mathbf{\Theta}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y^{(i)}} \max(0, (\mathbf{h}_L^{(i)})_j - (\mathbf{h}_L^{(i)})_{y^{(i)}} + \Delta) \quad (3)$$

which is interpreted as a triplet loss as shown in section 3.

To train our model, we apply mini-batch stochastic gradient descent on the regularized objective function in Equation 2. For each example in the mini-batch, a stochastic estimator of $L(\mathbf{\Theta})$ is naturally constructed. For the construction of a stochastic estimator of $R(\mathbf{\Theta})$, we consider two online

triplet selection methods: `all` and `hard` [44]. The `all` method selects every valid triplet within a mini-batch. In the `hard` method, each example acts once as an anchor example while a single positive example and a single negative example is chosen within the mini-batch. Specifically, we choose the closest negative example to and the furthest positive example form the anchor example within the same mini-batch to construct a triplet with the largest triplet loss. Since the triplets are built within each mini-batch, we require large enough mini-batch size so that there are both positive and negative examples for each anchor example. Note that when we are using the hard method, using larger mini-batch size induce more bias on our choice of positive and negative examples.

3 Theoretical Analysis

In this section, we describe a theoretical analysis of triplet loss. We provide a novel interpretation of triplet loss by drawing connections to multi-class hinge loss and mixture of Gaussians. Then, we associate these interpretations to the classification performance of the corresponding neural network through mutual information and generalization bound. Proofs of all theorems and lemmas are in the Appendix A.

3.1 Multi-Class Hinge Loss as Triplet Loss

The interpretation of multi-class hinge loss as triplet loss is given in Theorem 1.

Theorem 1. *The multi-class hinge loss in Equation 3 corresponds to triplet loss where each data point acts as an anchor example and the positive and negative examples are virtual data points whose features are c times a one-hot vector for some positive constant c and the margin is $\alpha = 2c\Delta$.*

Proof. See Appendix A, Theorem 1. □

We regard the hidden features $ce_{y^{(i)}}$ of positive examples as ideal hidden features for the correct class and the hidden features ce_j of negative examples as ideal hidden features for incorrect classes. Note that the above theorem is valid for any positive constant c. The derived triplet loss is weighted by $\frac{1}{2c}$ and the margin for triplet loss is proportional to c.

3.2 Effects of Triplet Loss on Feature Space

We study the effect of triplet loss on features space. Specifically, we show that triplet loss imposes penalty on the distance to the centroid of the corresponding class while rewards the distance to the centroids of other classes. We relate this property to the mixture of Gaussian and analyze triplet loss in terms of the probability distribution over feature space.

Since our derivation is the same for each layer $\ell \in H$, we only consider the following regularization term corresponding to the ℓ-th layer.

$$R^{\ell}(\mathbf{\Theta}) = \frac{1}{|T|} \sum_{(\mathbf{x}^{(\text{anc})}, \mathbf{x}^{(\text{pos})}, \mathbf{x}^{(\text{neg})}) \in T} \max(0, \|\mathbf{h}_{\ell}^{(anc)} - \mathbf{h}_{\ell}^{(pos)}\|^2 - \|\mathbf{h}_{\ell}^{(anc)} - \mathbf{h}_{\ell}^{(neg)}\|^2 + \alpha)$$

Lemma 1 shows how triplet loss induces feature clusters for each class.

Lemma 1. *Suppose that the training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \cdots, N\}$ is given and the triplets in T are chosen with uniform probability over all valid triplets. Then, the following inequality holds:*

$$R^{\ell}(\mathbf{\Theta}) \geq S^{\ell}(\mathbf{\Theta}) \tag{4}$$

where

$$S^{\ell}(\mathbf{\Theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(w_{y^{(i)}} \|\mathbf{h}_{\ell}^{(i)} - \mathbf{m}_{y^{(i)}}\|^2 - \sum_{\substack{j=1 \\ j \neq y^{(i)}}}^{K} \frac{n_j}{N - n_{y^{(i)}}} \|\mathbf{h}_{\ell}^{(i)} - \mathbf{m}_j\|^2 \right) + \alpha \tag{5}$$

and

$$n_k = \sum_{i=1}^{N} \mathbb{1}(y^{(i)} = k), w_k = \frac{2n_k}{n_k - 1} - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n_j}{N - n_j} \text{ and } \mathbf{m}_k = \frac{1}{n_k} \sum_{i=1}^{N} \mathbf{h}_\ell^{(i)} \mathbb{1}(y^{(i)} = k)$$

Moreover, the inequality gets tighter as α increases and becomes equality in the limit $\alpha \to \infty$.

Proof. See Appendix A, Lemma 1. □

Lemma 1 states that, for large enough α, triplet loss minimizes the distance to correct centroid and maximizes the distance to incorrect centroids. Note that if there are the same number n of examples for each class, weight terms become $w_k = \frac{2n}{n-1} - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n}{nK-n} = \frac{n+1}{n-1}$ and $\frac{n_j}{N-n_k} = \frac{n}{nK-n} = \frac{1}{K-1}$.

We further develop the probabilistic interpretation of Lemma 1. From the clustering induced by minimizing $S^\ell(\mathbf{\Theta})$, we model feature space as a mixture of Gaussians where each Gaussian component corresponds to a single class. We define the probability density function of the Gaussian mixture as

$$p(\mathbf{h}) = \sum_{k=1}^{K} \gamma_k \mathcal{N}(\mathbf{h}; \boldsymbol{\mu}_k, \sigma^2 I)$$

Theorem 2 provides $S^\ell(\mathbf{\Theta})$ under this probability model.

Theorem 2. *We have the following probabilistic interpretation of $S^\ell(\mathbf{\Theta})$.*

$$S^\ell(\mathbf{\Theta}) = \frac{2\sigma^2}{N} \sum_{i=1}^{N} \left(-(w_{y^{(i)}} - 1) \log \hat{p}(\mathbf{h}_\ell^{(i)} | y^{(i)}) - \sum_{\substack{j=1 \\ j \neq y^{(i)}}}^{K} \frac{n_j}{N - n_{y^{(i)}}} \log \frac{\hat{p}(\mathbf{h}_\ell^{(i)} | y^{(i)})}{\hat{p}(\mathbf{h}_\ell^{(i)} | j)} \right) + \text{const.}$$

Under the assumption of the same number of examples for each class, it is further simplified as

$$S^\ell(\mathbf{\Theta}) = \frac{4\sigma^2}{n-1} \hat{H}(\mathbf{h}_\ell | y) - \frac{2\sigma^2}{K(K-1)} \sum_{j \neq k} \hat{D}_{KL}(p(\mathbf{h}_\ell | j) || p(\mathbf{h}_\ell | k)) + \text{const.} \tag{6}$$

Proof. See Appendix A, Theorem 2. □

Theorem 2 states that, for large enough α, triplet loss maximizes the likelihood of feature given the class it belongs to and the KL divergence between Gaussians for different classes. Note that learning is applied on feature space and not on labels. Therefore, triplet loss induces feature space where clusters for each class are more separated.

3.3 Effects of Triplet Loss on Classification Performance

We study the effect of feature space separation on the performance of classification. From Theorem 2, we show that minimizing triplet loss decreases the conditional entropy $H(\mathbf{h}_\ell | y)$ of hidden feature $\mathbf{h}_\ell$ given the label y. In addition, the Gaussians for different classes are separated by maximizing the KL divergence which decreases the uncertainty on the label given a hidden feature. That is, the conditional entropy $H(y | \mathbf{h}_\ell)$ of the label y given hidden feature $\mathbf{h}_\ell$ also decreases. This leads to the increase in the mutual information $I(\mathbf{h}_\ell; y)$ between hidden feature $\mathbf{h}_\ell$ and label y. These analyses of information measures are consistent with the previously proposed information theoretic regularization approaches [45, 6, 42] and consequently explain the improvement in classification performance.

The separation of different classes in feature space also leads to larger margins between different classes. If the learned classifier shows larger margin between different classes, then the generalization bound is smaller [23]. That is, the difference in the expected errors for the learned classifier and the optimal classifier is small. This analysis shows the regularization effect of triplet loss.

4 Experimental Results

Experimental Setups. We refer to triplet regularization using the `all` and `hard` method as `all` triplet regularization and `hard` triplet regularization respectively. We evaluate `all` and `hard` triplet regularization on two benchmark datasets: MNIST [10] and CIFAR-10 [24]. MNIST consists of a 60,000 image training set and 10,000 image test set consisting of 28×28 grey scale images that represent hand-written digits. CIFAR-10 consists of a 50,000 image training set and 10,000 image test set consisting of 32×32 RGB images in 10 classes. All of our performance is measured on the designated test sets. For the experiments on MNIST and CIFAR-10, we use AlexNet [25] and ResNet-18 [14], respectively. In both neural network architectures, we decrease the size of kernels in convolutional layers to compensate for the small image sizes present in MNIST and CIFAR-10. Additionlly, we add a second fully connected layer to the end of ResNet-18. All our experiments are done in code implemented with the PyTorch library [36].

Baseline Methods. We compare the performance of our proposed method against the following regularization methods which are most commonly used in the context of deep neural network training.

1. **Dropout** [46] randomly sets a portion of hidden units to zero during training. Dropout prevents over-fitting by preventing the network from relying on specific hidden units and from developing co-dependency between hidden units. One interpretation of dropout is it learns an ensemble of multiple neural networks that each use a subset of the full neural network architecture.
2. **Batch Normalization** [20] normalizes the activations of each layer by mean and variance computed from the current mini-batch. It was originally proposed as a remedy for the internal covariate shift of the activation distribution which complicates the learning of subsequent layers due to the consistent adaptation to new input distributions. Batch normalization not only facilitates the learning dynamics but also implicitly regularizes the neural network as shown empirically [41, 5] and theoretically [31].
3. **Weight Decay** [26] decreases the parameter value in every iteration. It is equivalent to an ℓ_2 norm penalty that constrains the complexity of the network.

Hyperparameter Selection. We train AlexNet on MNIST for 70 epochs and ResNet-18 on CIFAR-10 for 150 epochs. We use the plain stochastic gradient descent algorithm with a batch size of 128. This batch size is chosen to enable reasonable construction of triplets. Learning rate is decayed from 0.1 to 0.001 when training AlexNet and from 0.05 to 0.001 when training ResNet-18. For ResNet-18, we apply batch normalization and data augmentation by default and refer to this as "No Regularization." In this case, two data augmentation methods are used: cropping and flipping.

Triplet loss is applied to the final fully connected layers of both AlexNet and ResNet-18. We fix the margin Δ for multi-class hinge loss to 1 and the margin α for triplet loss to 0.5. We conduct 3-fold cross validation with `all` triplet regularization to select the regularization parameter λ. The results of cross-validation are shown in Appendix C, Figure 5. The selected regularization parameters are 0.15 and 0.05 for AlexNet and ResNet-18, respectively. For more details, links to our code are available in Appendix B.

Additional computational cost for triplet selection and regularization gradient computation is minor. We observe that training time only increase by a few seconds per epoch on a single 12GB Tesla K80 GPU in a Google Cloud environment.

4.1 Accuracy

We compare the test accuracy of triplet loss regularization and other baseline methods. We measure the test accuracy at the end of each epoch and report the highest. The result is summarized in Table 1. For MNIST, triplet regularization outperforms dropout and no regularization. The highest accuracy is achieved for batch normalization but `hard` triplet regularization achieves comparable accuracy. For CIFAR-10, triplet regularization alone is outperformed by weight decay but still out performs dropout and no regularization. However, triplet regularization improves the performance when used with weight decay. Triplet regularization combined with weight decay gives the best result.

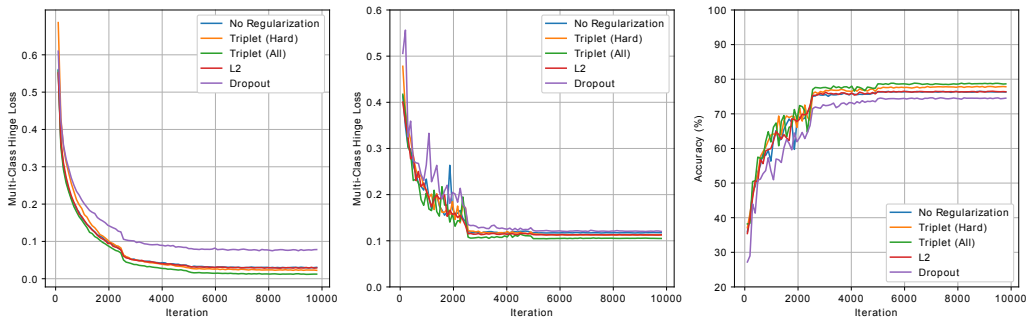
Table 1: **Test Accuracy.** The tables compare the accuracy result of each regularization method. The best result is highlighted in bold text. In MNIST, triplet regularization achieves comparable result to the best performing batch normalization. In CIFAR-10, triplet regularization achieves the highest accuracy when combined with weight decay.

(a) MNIST

Method	Accuracy (%)
No Regularization	99.35
Dropout	99.41
Batchnorm	**99.63**
Triplet (Hard)	99.62
Triplet (All)	99.49

(b) CIFAR-10

Method	Accuracy (%)
No Regularization	90.08
Weight decay (ℓ_2)	91.93
Dropout	86.37
Triplet (Hard)	90.75
Triplet (Hard) + ℓ_2	**92.76**
Triplet (All)	90.64
Triplet (All) + ℓ_2	92.47



(a) Training Loss (b) Test Loss (c) Test Accuracy

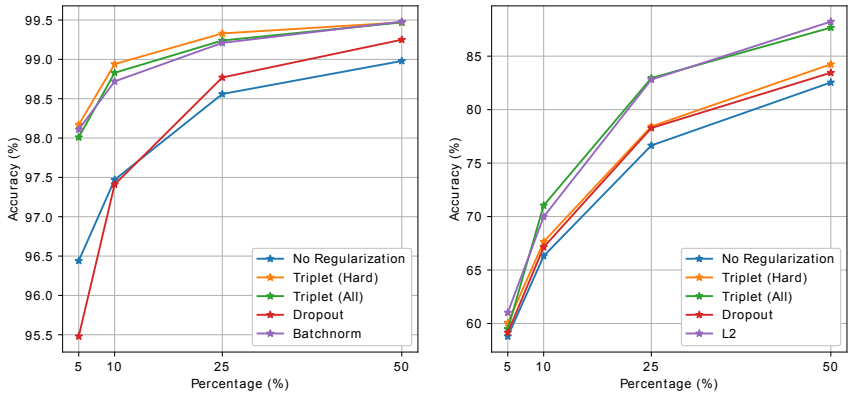
Figure 2: **Learning Curve.** The figures show the learning curve for each regularization method when trained on CIFAR-10. Each model is trained for 100 epochs on 25% of the training data. We show training loss, test loss and test accuracy. Triplet regularization achieves the best performance in terms of both loss and accuracy.

Figure 2 shows the learning curve of each regularization method on CIFAR-10 dataset. For this result, we train for 100 epochs on 25% of the training data. Every regularization method shows a gap between training and test loss because of the small training data size. Triplet regularization achieves the best performance in both test loss and test accuracy. This is due to the better optimization performance of triplet loss as shown in the trainig loss curve. Even though triplet regularization achieves lower training loss, the gap between training and test loss does not change much across regularization methods except for dropout. Note that dropout cannot achieves good training loss because of the innate noise in the training process.

We visualize the resulting feature space in Figure 1 to highlight the effect of triplet regularization. We use AlexNet architecture trained on MNIST for the visualization. We sample 250 random test examples and map features to two dimensions using PCA. Activations for each class are clustered more tightly in the fully connected layer for triplet regularization compared to other baseline method. This visualization is consistent with our theoretical result that entropy for each cluster is minimized and divergence between clusters is maximized.

4.2 Effect of Data Size

The regularization affects the performance more significantly when the training data size is small. Especially, deep neural networks with a large number of parameters are prone to overfitting in the small training data regime. We use a subset of our training data to simulate this regime and study the performance change when the size of training data changes. We use 5%, 10%, 25% and 50% of training data. The result is shown in Figure 3. In MNIST, `hard` triplet regularization performs the



(a) MNIST (b) CIFAR-10

Figure 3: **Limited Training Data.** We measure the test accuracy of each regularization method as training data size decreases. In both dataset, the best performing method is triplet regularization consistently across different portions of training data. In MNIST dataset, the performance gap increases as we use smaller portion of the training dataset.

best consistently across different size of training data. In addition, the performance gap increases as we further decreases the percentage of data. In CIFAR-10, weight decay and `all` triplet regularization performs the best. We conclude that our triplet regularization outperforms baseline regularization methods consistently when the training data size is small. Especially, as shown in MNIST result, the regularization effect increases when training data size decreases.

4.3 Robustness to Adversarial Examples

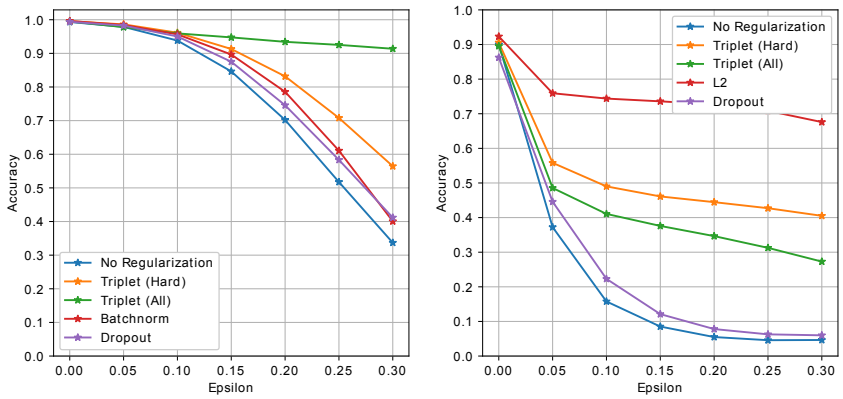
In this section, we compare the adversarial robustness of each regularization method. The ideal regularization in the context of deep neural network training induces the learning of features that only catch predictive properties of input on output prediction and that is invariant to small irrelevant perturbations. Adversarial perturbations are such small irrelevant perturbations that affects the performance in the worst possible way. Especially, we use the Fast Gradient Sign Method (FGSM) to generate adversarial perturbations. FGSM searches for adversarial examples by perturbing an input x in the direction of the gradient $\nabla L_x(x)$ of the multi-class cross-entropy loss $L(x)$ [13]. This perturbation increases the loss and can cause the model to make incorrect predictions.

We perturb each input in the test set using the FGSM algorithm and evaluate the performance of each method on the perturbed inputs as we change the size ϵ of perturbations. The results are shown in Figure 4. In MNIST dataset, `all` triplet regularization shows superior performance consistently across different size of perturbations. In addition, the performance gap increases as the size of perturbation increases. In CIFAR-10 dataset, weight decay is performing the best. However, triplet regularization still outperforms other regularization methods by large margin across all values of ϵ.

We relate the adversarial robustness of our method to the theoretical analysis. Theorem 2 states that triplet loss decreases the entropy of features for each class and increases the KL divergence between feature distributions of different classes. By decreasing the entropy, features that fall in the same class are clustered more closely. Therefore, the effect of adversarial perturbations on feature is constrained. On the other hand, increased KL divergence leads to better separation of features from the different classes. This leads to easier classification even when features are perturbed adversarially.

5 Related Works

Penalty Based Regularization. One class of regularization methods explicitly states desirable properties of the final models as a penalty function. The training process optimizes the summation of the original objective function and the penalty function. In the Lagrangian interpretation, this process



(a) MNIST (b) CIFAR-10

Figure 4: **FGSM Adversarial Attacks.** We show the adversarial robustness by measuring test accuracy as we inject larger perturbation magnitude ϵ. The triplet loss shows superior performance on the MNIST dataset and outperforms dropout and no regularization case in CIFAR-10 dataset. Note that the performance gap increases as we inject larger adversarial perturbations.

is equivalent to imposing an upper bound constraint on the penalty function. Our method also induces a regularization effect by introducing a penalty term.

Some penalty terms directly apply to the parameter norm: ℓ_2 norm penalty as in weight decay, ℓ_1 norm penalty [50] or group sparsity penalty [52] to learn sparse parameter, spectral norm penalty [53, 48] to restrict the perturbation invariance. Other penalty terms apply on the derivatives to constrain the complexity of the model: Jacobian penalty [38], Hessian penalty [37] and Tikhonov regularization [4] penalizes the first or higher order derivatives of the model with respect to the input variable; MDL penalty [17] penalizes the derivative of the model with respect to the parameter; Gradient regularization [32] penalizes the derivative of the loss function with respect to the parameter. These methods constrain the model function to be invariant to input variable or parameter values. Furthermore, there are penalty terms that directly apply on the feature space: Variance Constancy Loss [30] stabilizes the variance of activations across mini-batches to replace normalization; Orthogonality penalty [51, 3] stabilizes the distribution of activations across layers by preserving energy; Mutual exclusivity penalty [40, 39] penalizes the decision boundary being in the high-density region in the feature space. Our proposed triplet loss regularization method applies penalty in the feature space of intermediate layers. Consequently, our proposed method distinguishes the features of different classes as much as possible

Triplet loss [7, 44] is originally devised to perform ranking tasks. Although it was exploited in the context of metric learning [18] and adversarial training [29], we are the first to apply triplet loss for the regularization of deep neural network training.

Generalization of Neural Networks. Margin based analysis of generalization provides rich theoretical framework [23, 33, 28]. In the context of deep neural network, generalization bound decreases as the output layer margin increases [47]. Some prior works [11, 21] explicitly maximizes the margin at intermediate layers to achieve regularization performance. Our work is different from the above works by proposing a novel application of triplet loss for regularization and by analyzing the effect on predictive performance.

There are a thread of works [45, 1, 43] that analyze generalization of neural network in information theoretic framework. Based on this analysis, some works [6, 42] regularize directly on information theoretic measures such as conditional entropy and mutual information. On the other hand, our proposed triplet loss implicitly achieve the regularization effect on conditional entropy and mutual information.

6 Conclusion

We propose a novel triplet loss regularization for deep neural networks. Our triplet loss regularization is applied in the feature space of intermediate layers of a neural network. We analyze the triplet loss theoretically and explain the effect on classification performance. Our experimental results show that our method improves both generalization performance and adversarial robustness.

In the future, it would be interesting to compare our method against more recent state-of-the-art regularization methods and possibly substitute our regularization term with the theoretical lower bound in Equation 4. We would also like to explore the effects of hyperparameters in more detail by conducting a more rigorous and extensive search. Additionally, we would like to test the performance of our model on larger datasets such as ImageNet with larger models.

Self Evaluation. Overall, we were able to achieve our goals set-out in the project proposal and are happy with our final project. Additional evaluation of our project can be found in Appendix D.

References

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

[2] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *arXiv preprint arXiv:1905.13655*, 2019.

[3] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, pages 4261–4271, 2018.

[4] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[5] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

[6] Michael Blot, Thomas Robert, Nicolas Thome, and Matthieu Cord. Shade: Information-based regularization for deep learning. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 813–817. IEEE, 2018.

[7] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010.

[8] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[9] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

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

[11] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In *Advances in neural information processing systems*, pages 842–852, 2018.

[12] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in deep linear neural networks. *arXiv preprint arXiv:1904.13262*, 2019.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[14] 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.

[15] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[16] Geoffrey E Hinton. Learning translation invariant recognition in a massively parallel networks. In *International Conference on Parallel Architectures and Languages Europe*, pages 1–13. Springer, 1987.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536, 1995.

[18] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[19] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *International Conference on Learning Representations*, 2019.

[22] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

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

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[26] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[27] Masayoshi Kubo, Ryotaro Banno, Hidetaka Manabe, and Masataka Minoji. Implicit regularization in over-parameterized neural networks. *arXiv preprint arXiv:1903.01997*, 2019.

[28] John Langford and John Shawe-Taylor. Pac-bayes & margins. In *Advances in neural information processing systems*, pages 439–446, 2003.

[29] Pengcheng Li, Jinfeng Yi, Bowen Zhou, and Lijun Zhang. Improving the robustness of deep neural networks via adversarial training with triplet loss. *arXiv preprint arXiv:1905.11713*, 2019.

[30] Etai Littwin and Lior Wolf. Regularizing by the variance of the activations' sample-variances. In *Advances in Neural Information Processing Systems*, pages 2115–2125, 2018.

[31] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *International Conference on Learning Representations*, 2019.

[32] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE International Conference on Data Mining*, pages 301–309. IEEE, 2015.

[33] David McAllester. Simplified pac-bayesian margin bounds. In *Learning theory and Kernel machines*, pages 203–215. Springer, 2003.

[34] John E Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In *Advances in neural information processing systems*, pages 847–854, 1992.

[35] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[37] Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*, 2011.

[38] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, pages 833–840, 2011.

[39] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1908–1912. IEEE, 2016.

[40] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.

[41] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[42] Antoine Saporta, Yifu Chen, Michael Blot, and Matthieu Cord. Reve: Regularizing deep learning with variational entropy bound. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1610–1614. IEEE, 2019.

[43] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2019.

[44] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[45] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[47] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[48] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6541–6550, 2018.

[49] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel. *arXiv preprint arXiv:1810.05369*, 2018.

[50] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[51] Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6176–6185, 2017.

[52] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3958–3966. JMLR. org, 2017.

[53] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[54] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations*, 2017.

A Proofs of Theoretical Analysis

In this section, we provide the proofs of theorems and lemmas in Section 3.

Theorem 1. *The multi-class hinge loss in Equation 3 corresponds to triplet loss for multiple triplets where each data point acts as an anchor example and the positive and negative examples are virtual data points whose features are c times a one-hot vector for some positive constant c. The margin is $\alpha = 2c\Delta$.*

Proof. We define the one-hot vector notation $\mathbf{e}_i \in \mathbb{R}^K$ as a vector with a 1 in position i and 0's in other positions. Using this notation, multi-class hinge loss term corresponding to example $\mathbf{x}^{(i)}$ is represented as

$$\begin{aligned} L_i(\Theta) &= \sum_{j \neq y^{(i)}} \max(0, (\mathbf{h}_L^{(i)})_j - (\mathbf{h}_L^{(i)})_{y^{(i)}} + \Delta) \\ &= \frac{1}{2c} \sum_{j \neq y^{(i)}} \max(0, 2c(\mathbf{h}_L^{(i)})^T e_j - 2c(\mathbf{h}_L^{(i)})^T e_{y^{(i)}} + 2c\Delta) \\ &= \frac{1}{2c} \sum_{j \neq y^{(i)}} \max(0, \|\mathbf{h}_L^{(i)} - ce_{y^{(i)}}\|^2 - \|\mathbf{h}_L^{(i)} - ce_j\|^2 + 2c\Delta) \end{aligned}$$

where c is any positive constant and we use the fact that $\|e_i\| = 1$. Therefore, the multi-class hinge loss is the summation of triplet losses for every $j \neq y^{(i)}$ with the margin $\alpha = 2c\Delta$. The anchor features are $\mathbf{h}_L^{(i)}$ which corresponds to $\mathbf{x}^{(i)}$. The positive features $ce_{y^{(i)}}$ and the negative features ce_j are one-hot vectors multiplied by the constant factor $c > 0$ and do not correspond to any real data points. □

Lemma 1. *Suppose that the training dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}) : i = 1, \cdots, N\}$ is given and the triplets in T are chosen with uniform probability over all valid triplets. Then, the following inequality holds:*

$$R^\ell(\mathbf{\Theta}) \geq S^\ell(\mathbf{\Theta}) \tag{7}$$

where

$$S^\ell(\mathbf{\Theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(w_{y^{(i)}} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_{y^{(i)}}\|^2 - \sum_{\substack{j=1 \\ j \neq y^{(i)}}}^{K} \frac{n_j}{N - n_{y^{(i)}}} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_j\|^2 \right) + \alpha \tag{8}$$

and

$$n_k = \sum_{i=1}^{N} \mathbb{1}(y^{(i)} = k), w_k = \frac{2n_k}{n_k - 1} - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n_j}{N - n_j} \text{ and } \mathbf{m}_k = \frac{1}{n_k} \sum_{i=1}^{N} \mathbf{h}_\ell^{(i)} \mathbb{1}(y^{(i)} = k)$$

Moreover, the inequality gets tighter as α increases and becomes equality in the limit $\alpha \to \infty$.

Proof. We first decompose uniform random sampling of a valid triplet $(\mathbf{x}^{(anc)}, \mathbf{x}^{(pos)}, \mathbf{x}^{(neg)}) \in T$ into multiple stages as

1. choose a class k with probability $\frac{n_k}{N}$
2. pick anchor example $\mathbf{x}^{(anc)}$ with uniform random probability among examples with label k
3. sample a positive example $\mathbf{x}^{(pos)}$ with uniform probability over examples with label k except for the anchor example
4. choose another class $j \neq k$ with probability $\frac{n_j}{N-n_k}$
5. sample a negative example $\mathbf{x}^{(neg)}$ with uniform probability over examples with label j

We denote the set of indices for examples in class k as $C_k = \{i : y^{(i)} = k\}$. Then, the expected triplet loss is computed as

$$
\begin{aligned}
R^\ell(\mathbf{\Theta}) &= \frac{1}{|T|} \sum_{(\mathbf{x}^{(\text{anc})}, \mathbf{x}^{(\text{pos})}, \mathbf{x}^{(\text{neg})}) \in T} \max(0, \|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(pos)}\|^2 - \|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(neg)}\|^2 + \alpha) \\
&\geq \frac{1}{|T|} \sum_{(\mathbf{x}^{(\text{anc})}, \mathbf{x}^{(\text{pos})}, \mathbf{x}^{(\text{neg})}) \in T} \left[\|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(pos)}\|^2 - \|\mathbf{h}_\ell^{(anc)} - \mathbf{h}_\ell^{(neg)}\|^2 + \alpha \right] \\
&= \sum_{k=1}^{K} \frac{n_k}{N} \sum_{i \in C_k} \frac{1}{n_k} \left[\sum_{\substack{i_p \in C_k \\ i_p \neq i}} \frac{1}{n_k - 1} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_p)}\|^2 - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n_j}{N - n_k} \sum_{i_n \in C_j} \frac{1}{n_j} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_n)}\|^2 \right] + \alpha \\
&= \frac{1}{N} \sum_{k=1}^{K} \left[\frac{1}{n_k - 1} \sum_{i, i_p \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_p)}\|^2 - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{1}{N - n_k} \sum_{\substack{i \in C_k \\ i_n \in C_j}} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_n)}\|^2 \right] + \alpha
\end{aligned}
$$

The distances between anchor and positive examples are simplified as

$$
\sum_{i, i_p \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_p)}\|^2 = 2n_k \sum_{i \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_k\|^2
$$

and the distances between anchor and negative examples are simplified as

$$
\begin{aligned}
\sum_{\substack{i \in C_k \\ i_n \in C_j}} \|\mathbf{h}_\ell^{(i)} - \mathbf{h}_\ell^{(i_n)}\|^2 &= \sum_{\substack{i \in C_k \\ i_n \in C_j}} \left(\|\mathbf{h}_\ell^{(i)} - \mathbf{m}_j\|^2 + \|\mathbf{m}_j - \mathbf{h}_\ell^{(i_n)}\|^2 + 2\langle \mathbf{h}_\ell^{(i)} - \mathbf{m}_j, \mathbf{m}_j - \mathbf{h}_\ell^{(i_n)} \rangle \right) \\
&= n_j \sum_{i \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_j\|^2 + n_k \sum_{i_n \in C_j} \|\mathbf{m}_j - \mathbf{h}_\ell^{(i_n)}\|^2 \\
&\quad + 2\langle \sum_{i \in C_k} (\mathbf{h}_\ell^{(i)} - \mathbf{m}_j), \sum_{i_n \in C_j} (\mathbf{m}_j - \mathbf{h}_\ell^{(i_n)}) \rangle \\
&= n_j \sum_{i \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_j\|^2 + n_k \sum_{i_n \in C_j} \|\mathbf{m}_j - \mathbf{h}_\ell^{(i_n)}\|^2
\end{aligned}
$$

Therefore, we have

$$
\begin{aligned}
R^\ell(\mathbf{\Theta}) \geq \frac{1}{N} \sum_{k=1}^{K} \Bigg[&\frac{2n_k}{n_k - 1} \sum_{i \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_k\|^2 \\
&- \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n_j}{N - n_k} \sum_{i \in C_k} \|\mathbf{h}_\ell^{(i)} - \mathbf{m}_j\|^2 - \sum_{\substack{j=1 \\ j \neq k}}^{K} \frac{n_k}{N - n_k} \sum_{i_n \in C_j} \|\mathbf{m}_j - \mathbf{h}_\ell^{(i_n)}\|^2 \Bigg] + \alpha
\end{aligned}
$$

$$= \frac{1}{N}\sum_{k=1}^{K}\sum_{i\in C_k}\left[\left(\frac{2n_k}{n_k-1}-\sum_{\substack{j=1\\ j\neq k}}^{K}\frac{n_j}{N-n_j}\right)\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_k\|^2-\sum_{\substack{j=1\\ j\neq k}}^{K}\frac{n_j}{N-n_k}\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_j\|^2\right]+\alpha$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(w_{y^{(i)}}\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_{y^{(i)}}\|^2-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_j\|^2\right)+\alpha = S^\ell(\mathbf{\Theta})$$

The inequality comes from

$$\max(0,\|\mathbf{h}_\ell^{(anc)}-\mathbf{h}_\ell^{(pos)}\|^2-\|\mathbf{h}_\ell^{(anc)}-\mathbf{h}_\ell^{(neg)}\|^2+\alpha)\geq\|\mathbf{h}_\ell^{(anc)}-\mathbf{h}_\ell^{(pos)}\|^2-\|\mathbf{h}_\ell^{(anc)}-\mathbf{h}_\ell^{(neg)}\|^2+\alpha$$

where the left hand side equals the right hand side for large enough α. As α increases, this equality is satisfied by more triplets and eventually by all triplets. □

Theorem 2. *We have the following probabilistic interpretation of $S^\ell(\mathbf{\Theta})$.*

$$S^\ell(\mathbf{\Theta}) = \frac{2\sigma^2}{N}\sum_{i=1}^{N}\left(-(w_{y^{(i)}}-1)\log\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}\log\frac{\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})}{\hat{p}(\mathbf{h}_\ell^{(i)}|j)}\right)+\text{const.}$$

Under the assumption of the same number of examples for each class, it is further simplified as

$$S^\ell(\mathbf{\Theta}) = \frac{4\sigma^2}{n-1}\hat{H}(\mathbf{h}_\ell|y)-\frac{2\sigma^2}{K(K-1)}\sum_{j\neq k}\hat{D}_{KL}(p(\mathbf{h}_\ell|j)||p(\mathbf{h}_\ell|k))+\text{const.} \tag{9}$$

Proof. From the Gaussian mixture model, we have the following log conditional probability.

$$\log p(\mathbf{h}_\ell|y=k) = -\frac{1}{2\sigma^2}\|\mathbf{h}_\ell-\boldsymbol{\mu}_k\|^2-\frac{1}{2}\log(2\pi\sigma^2)$$

Given the dataset, $\mathbf{m}_k$ is the empirical estimator of $\boldsymbol{\mu}_k$ so we have the following estimated log conditional probability

$$\log\hat{p}(\mathbf{h}_\ell|y=k) = -\frac{1}{2\sigma^2}\|\mathbf{h}_\ell-\mathbf{m}_k\|^2-\frac{1}{2}\log(2\pi\sigma^2)$$

Using this estimator, we have

$$S^\ell(\mathbf{\Theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(w_{y^{(i)}}\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_{y^{(i)}}\|^2-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_j\|^2\right)+\alpha$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left((w_{y^{(i)}}-1)\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_{y^{(i)}}\|^2-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}(\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_j\|^2-\|\mathbf{h}_\ell^{(i)}-\mathbf{m}_{y^{(i)}}\|^2)\right)+\alpha$$

$$= \frac{2\sigma^2}{N}\sum_{i=1}^{N}\left(-(w_{y^{(i)}}-1)(\log\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})+\frac{1}{2}\log(2\pi\sigma^2))-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}\log\frac{\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})}{\hat{p}(\mathbf{h}_\ell^{(i)}|j)}\right)+\alpha$$

$$= \frac{2\sigma^2}{N}\sum_{i=1}^{N}\left(-(w_{y^{(i)}}-1)\log\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})-\sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{n_j}{N-n_{y^{(i)}}}\log\frac{\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})}{\hat{p}(\mathbf{h}_\ell^{(i)}|j)}\right)+\text{const.}$$

If we assume that there are the same number n of examples for each class, it is further simplified as

$$\frac{2\sigma^2}{N}\sum_{i=1}^{N}\left(-(\frac{n+1}{n-1}-1)\log\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)}) - \sum_{\substack{j=1\\ j\neq y^{(i)}}}^{K}\frac{1}{K-1}\log\frac{\hat{p}(\mathbf{h}_\ell^{(i)}|y^{(i)})}{\hat{p}(\mathbf{h}_\ell^{(i)}|j)}\right) + \text{const.}$$

$$= \frac{4\sigma^2}{n-1}\hat{H}(\mathbf{h}_\ell|y) - \frac{2\sigma^2}{K(K-1)}\sum_{\substack{j,k=1\\ j\neq k}}^{K}\hat{D}_{KL}(p(\mathbf{h}_\ell|j)||p(\mathbf{h}_\ell|k)) + \text{const.}$$

where $\hat{H}$ and $\hat{D}_{KL}$ are the empirical estimation of entropy and KL divergence, respectively. □

B Project Code

Our code is implemented on top of the PyTorch library [36] and the custom library which one of our members maintains. Specifically, our triplet loss penalty is implemented in the deep.lib library.

Our implementations of neural network architectures are based on the baseline implementations in the PyTorch library. For the implementation of FGSM for adversarial perturbation, we use the implementation in the official PyTorch tutorial. We submit the anonymized Google Colab Notebooks that we used for our experiments: MNIST[1], CIFAR-10[2], MNIST Adversarial Examples[3] and CIFAR-10 Adversarial Examples[4].

C Cross-Validation

We conduct cross-validation and hyperparameter search in parallel with our primary experiments due to time-constraints. Due to this, the hyperparameters used in our experiments do not match the optimal parameters from cross-validation. The results of 3-fold cross validation are shown in Figure 5.

D Project Review

We were able to satisfactorily achieve our goals from our project proposal. There were a few implementation issues we faced during the course of the project. Google Colab provides GPU runtimes for limited slots. Due to these constraints we could not materialize some additional interesting ideas and experiments. For example, in our initial proposal, we had desired to test our method on ImageNet; however, due to time and resource constraints, we have left training on ImageNet as future work.

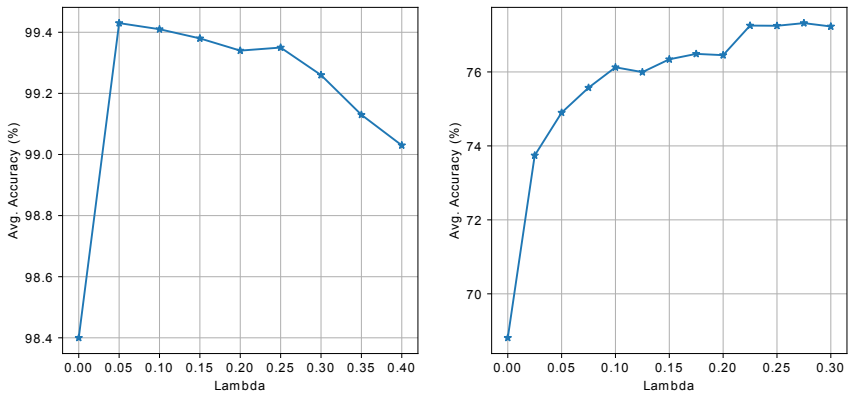
Overall, we are happy with our project and our results.

[1] https://colab.research.google.com/drive/1ogaCk7Ac3L_KWkDOFRjVAtuXBuD9MPfW

[2] https://colab.research.google.com/drive/1HACNOQtIi9DABJtSZwloN5_PhWXH5J-o

[3] https://colab.research.google.com/drive/1LscKwP9FsCnAJ1-1xMtBrrUseuD_F3Vx

[4] https://colab.research.google.com/drive/1Rgmm9HQU23yXWaenxPKIN7YJkIGvEGky



(a) MNIST (b) CIFAR-10

Figure 5: **Cross Validation for Parameter Selection.** We conduct cross validation to select the parameter λ in equation 2 for *all* triplet regularization. For both MNIST and CIFAR-10, the training set is split into 3 folds. The vertical axis is the average accuracy achieved across each fold. The margin parameter α in equation 1 is fixed at $.5$. To reduce the time-cost of cross validation, training is stopped early at 30 epochs and 80 epochs for MNIST and CIFAR-10 respectively and only 75% of the CIFAR-10 training data is used.