Supervised Contrastive Learning
CS 7643 Project Report

Xiaofei Chen
xchen837@gatech.edu

Simin Liu
sliu831@gatech.edu

Zhikang Dong
zdong302@gatech.edu

Yunkai Wang
ywang723@gatech.edu

Abstract

Contrastive learning methods have gained traction in recent years, particularly in supervised contrastive learning (SCL) in which label information boosts performance. In this work, we compare popular loss functions in SCL and analyze the impact on network performance from the choice of loss functions, architecture modifications, and data augmentation. We find that SupCon loss introduced by Khosla et al. achieves the best accuracy of 91.40% across the loss functions we consider, and triplet loss and traditional cross entropy loss are only marginally worse. We do not observe significant performance changes by adding a projector or a normalization layer. Data augmentation improves performance of SCL in a meaningful way. Visualization shows SCL without data augmentation generates tighter and more distinct clusters than standard supervised learning, suggesting it may better suit tasks for which data augmentation is more challenging. Our code repository is at https://github.com/Dongzhikang/cs7643_deep_learning_project.

1. Introduction/Background/Motivation

Contrastive learning, a discriminative approach, aims to learn an embedding space in which raw input data are transformed such that similar samples become close together while dissimilar ones are far apart, thus extracting abstract yet meaningful representations. It has mainly been applied to self-supervised representation learning, leading to state of the art performances in transfer learning for some downstream computer vision tasks [1, 22, 7, 19, 18, 6, 5, 11, 13, 10, 15, 17]. Contrastive learning is also widely used in natural language processing tasks. Kim et al. redesigned contrastive learning to improve BERT [9]. Since there is no label information, the common idea in these works is to produce a positive pair between different augmentations of the same image and negative pairs between different images. Using a contrastive loss function, the model then learns to pull together positive pairs and push apart negative pairs. In this work, we explore contrastive learning in a supervised setting by implementing the paper *Supervised Contrastive Learning* [8], in which a new loss termed SupCon was introduced. SupCon allows for multiple positives, and therefore can incorporate label information in the loss function (positive pairs are formed between augmentations of the same image or of the same label).

Since there is already a PyTorch implementation of the supervised contrastive learning (SCL) framework (https://github.com/HobbitLong/SupContrast), we aim to expand their work by performing experiments on various aspects of design choices, including the loss function, normalization on the embedding, data augmentation, and projector architecture. By doing so, we expect to gain a deeper understanding of design choices of contrastive learning frameworks, supervised and self-supervised.

Contrastive loss is one of the key differences between contrastive methods and other representation learning. In the following section, we will discuss the major contrastive loss functions related to SupCon.

1.1. Pair loss

$$\mathcal{L} = \mathbb{1}(y_i = y_j)||z_i - z_j||_2^2 + \mathbb{1}(y_i \neq y_j)max(0, m - ||z_i - z_j||_2)^2 \tag{1}$$

The original "contrastive loss" was introduced by Chopra et al. [2] and later reformulated by Hadsell et al. [4]. The term "contrastive loss" was originally referring to this specific loss but over time morphed into a general class of loss functions used for contrastive learning. To avoid confusion, we will refer to it as "pair loss". Pair loss takes one pair of embedding vectors (z_i, z_j). If the pair is a positive pair (from the same class), then the loss is essentially their Euclidean distance. If the pair is a negative pair (from different classes), then the loss is essentially a hinge loss with a margin m. After training same-class samples are pushed closer and different-class samples are pushed apart.

1.2. Triplet loss

$$\mathcal{L} = max(0, ||z_i - z_j||_2^2 - ||z_i - z_k||_2^2 + m) \tag{2}$$

Triplet loss operates on a triplet of embedding vectors (anchor z_i, positive z_j, negative z_k) with one positive pair and one negative pair. In the original paper [21], the goal is to learn a distance metric that keeps k-nearest neighbors from the same class close while keeping samples from different classes separated by a large margin. The selection of negative samples z_k is crucial to model performance and convergence [14]. If the negative is already more than a margin away from the anchor than the positive, the model does not need to learn anything. If the negative is closer to the anchor than the positive, it is too hard for the model to learn a good distance metric. The ideal situation is when the negative lies within the margin.

1.3. N-pair loss

$$\mathcal{L} = -\Sigma_{i\in\mathcal{B}} log \frac{exp(z_i \cdot z_j)}{\Sigma_{k\in\mathcal{B},k\neq i} exp(z_i \cdot z_k)} \quad (3)$$

To address the problem with pair loss and triplet loss where there is limited interaction between samples necessitating expensive hard negative mining, N-pair loss generalizes triplet loss to include comparison with multiple negative samples [16]. Given a batch $\mathcal{B}$ of size N+1 containing one anchor (z_i), one positive (z_j), and N-1 negative samples, N-pair loss is essentially the negative log of the softmax function of each positive pairs scored by their dot product. With this formulation, training with N-pair loss pushes the N-1 negative samples away simultaneously instead of one at a time as with triplet loss.

1.4. NT-Xent loss

$$\mathcal{L} = -\Sigma_{i\in\mathcal{B}} log \frac{exp(z_i \cdot z_j/\tau)}{\Sigma_{k\in\mathcal{B},k\neq i} exp(z_i \cdot z_k/\tau)} \quad (4)$$

NT-Xent loss is an extension of N-pair loss with the addition of the temperature parameter (τ) to scale the dot products (cosine similarities if using normalized embedding vectors). The addition of the temperature parameter effectively weighs different samples, and an appropriate temperature can help the model learn from hard negatives. In SimCLR where NT-Xent loss is coined [1], a set of N random samples is augmented to 2N, each being a "view" of the original sample. Note in this "multiviewed batch" $\mathcal{B}$ of 2N samples, for each anchor z_i, there is 1 positive pair and 2N - 2 negative pairs. The denominator has a total of 2N - 1 terms (the positive and negatives).

1.5. SupCon loss

$$\mathcal{L} = -\Sigma_{i\in\mathcal{B}} \frac{1}{|\mathcal{B}_i|} \Sigma_{j\in\mathcal{B}_i} log \frac{exp(z_i \cdot z_j/\tau)}{\Sigma_{k\in\mathcal{B},k\neq i} exp(z_i \cdot z_k/\tau)} \quad (5)$$

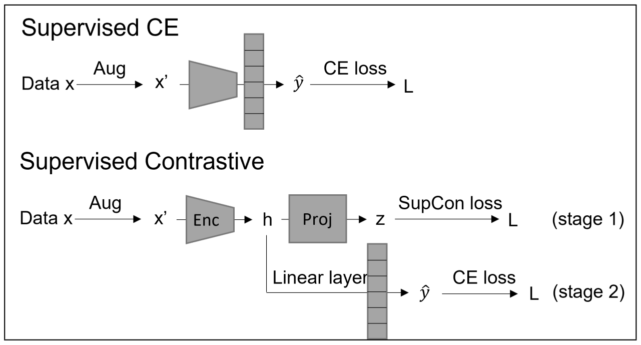

Figure 1. Architecture of the standard supervised classification framework (top) and SCL framework (bottom)

Finally, built upon NT-Xent loss, SupCon loss allows multiple positive and negative samples in the multiviewed batch, leveraging class label information. Here $\mathcal{B}_i = \{j \in \mathcal{B} : y_j = y_i, j \neq i\}$ contains samples belonging to the same class distinct from i, and $|\mathcal{B}_i|$ is its cardinality. Including more positive samples into the set leads to improved results, outperforming SimCLR (NT-Xent loss) and supervised classification with cross entropy (CE) loss.

2. Approach

Figure 1 shows the architecture of the SCL framework compared to a standard supervised learning framework. Given a batch of input data, the approach is first to apply data augmentation twice to obtain two copies of the batch. Both copies are then forward propagated through the encoder network (ResNet-50) to obtain a 2048-dimensional embedding. In Stage 1 (pretraining), this embedding is further propagated through the projector network (2-layer MLP) and then the normalized output (128-dimensional) is used to compute the SupCon loss. During Stage 2 (inference), the embedding h from the encoder network is frozen and propagated through a linear layer to make a class prediction. For standard supervised classification, a ResNet-50 is used on augmented data. We followed the SupCon implementation on `https://github.com/HobbitLong/SupContrast` with minimal tweaks to make it work on our end, which means we used their augmentation strategies (resized crop, horizontal flip, color jitter, and grayscale for SCL, and resized crop and horizontal flip for supervised learning) and optimizer (SGD with momentum).

In addition to the basic SCL framework, we are interested in experimenting with the following: 1) Explore all contrastive losses mentioned in Section 1 to compare with SupCon, including pair loss, triplet loss, N-pair loss, and NT-Xent loss. 2) SimCLR [1] introduced a projection head on top of the base neural network encoder and the SupCon paper inherited this architecture but did not experiment with different architectures of the projection network. We plan

Batch size	Learning rate						
	0.01	0.02	0.05	0.1	0.2	0.5	1
128	85.00	87.46	88.62	87.68	84.62	79.92	71.09
256	85.61	85.86	**88.81**	88.17	87.81	84.96	78.49
512	84.28	83.97	87.27	85.30	87.16	87.01	82.44
1024	72.80	74.63	80.57	81.61	86.95	84.63	81.70

Table 1. Validation accuracy using standard CE loss across various learning rates and batch sizes

to evaluate the effect of the projector by applying different architectures (no projection vs linear projection vs simple MLP non-linear projection). 3) Evaluate the effect of normalization on the encoder network and the projection network. Normalization has been shown to improve downstream classification accuracy [14, 20]. The SupCon paper normalized the output of the projector but not the output of the encoder. We tested both networks with and without normalization. 4) Evaluate the effect of data augmentation. To create positive samples, self-supervised contrastive learning relies on data augmentation techniques (cropping, color distortion, Gaussian blurring, etc.), which are often used in computer vision. The SupCon paper, although leveraging label information for contrastive learning, still used augmented data to form positive samples. We are interested in removing data augmentation in the contrastive learning pretraining stage so that our approach may be applied to additional realms (e.g., tabular data).

The dataset we used is the standard CIFAR-10. Initially we proposed to use CIFAR-100, but after running SCL on CIFAR-100, we realized our computational power was too limited to learn a good representation of CIFAR-100. Therefore, we used CIFAR-10 throughout the project. We trained for 40 epochs for each model as we feel it is a good trade-off between performance and training time. As a result, our accuracies were not as high as in the SupCon paper. We could always train for more epochs to get better models (the SupCon paper used 350 epochs for ResNet-50 models) but to have the best possible models is not our learning goal. Note that with 40 epochs, no learning rate decay was applied.

3. Experiments and Results

3.1. Supervised contrastive learning vs supervised learning

Given that we started with a different epoch and a different dataset from the SupCon paper, our first task was to tune the hyperparameters to establish baseline accuracies. As mentioned in Section 2 we fixed our epoch at 40 due to the limitation on our computational power, as training and tuning the networks require substantial usage of GPU for an extended amount of time to fit into the project timeline. However, we do not believe this limitation is likely to alter the conclusion of our experiments – we observed a slowdown in improvements in accuracy at epoch 40, and the accuracy differences were relatively large across most of our comparison results.

We explored various combinations of learning rates and batch sizes and applied them to the standard supervised learning network with CE loss (Table 1). Interestingly, we observed the best validation accuracy when learning rate = 0.05 and batch size = 256, and performance deteriorated with larger batch sizes or higher learning rates. Our optimal batch size was smaller than that from the SupCon paper, possibly due to the relatively simpler dataset we used.

For the SupCon network, learning rates and batch sizes were also the first set of hyperparameters we explored. As described in Section 2, SupCon first learns a feature representation, which is frozen at inference time and fed into a linear layer for classification. The implementation allows flexibility to have different hyperparameters in representation learning and linear classifier, but we applied the same set of hyperparameters across both stages for consistency. We observed that the best validation accuracy was achieved at batch size 256 and learning rate of 0.05 (Table 2).

With this insight, we moved on to testing the temperature τ in SupCon loss. The SupCon paper showed that τ at 0.1 was optimal with ImageNet and ResNet-50 encoder, which was what we used as the default. Temperature has an important role in SupCon learning because of its effects on smoothness and hard positives/negatives. We performed empirical testing on CIFAR-10 around 0.1, and accuracy peaked when $\tau = 0.15$ (Table 3).

Overall all of our tuning, SupCon achieved an accuracy of 91.40%, much higher than the 88.81% accuracy achieved by standard supervised learning framework. This conclusion is consistent with the SupCon paper, but we observed a much larger difference between the two frameworks. It is possible that SupCon trains more efficiently, and the gap may narrow as epoch becomes sufficiently large.

Additionally, we found both standard supervised learning and SupCon networks achieved the best performances at the same learning rate and batch size. This observation suggests that this set of hyperparameters may be the best combination for the given dataset and base architecture despite some variations in loss functions and layer construction, and we will use this setup throughout the remainder

Batch size	Learning rate			
	0.05	0.1	0.2	0.5
256	**90.95**	90.30	89.39	87.33
512	89.33	90.49	90.45	89.21

Table 2. Validation accuracy for SupCon across various learning rates and batch sizes

Temperature	0.07	0.10	0.15	0.20
Accuracy	89.18	90.95	**91.40**	91.31

Table 3. Validation accuracy for SupCon with varying temperatures

Loss function	Accuracy
CE	88.81
SupCon	90.95
SimCLR (NT-Xent)	75.54
N-pair	59.42
Triplet	90.03
Pair	22.62

Table 4. Validation accuracy using different loss functions

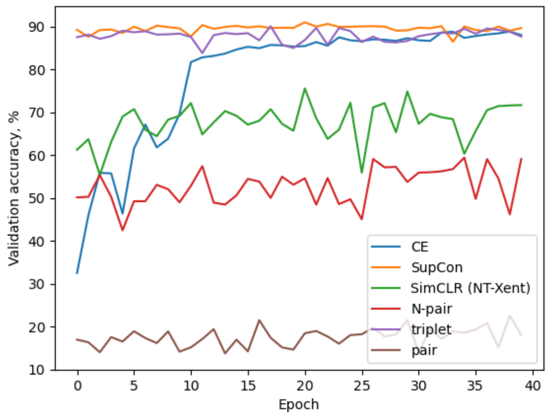

Figure 2. Learning curve (validation accuracy) of using different loss functions on CIFAR-10 image classification. Cross entropy (CE) is used for supervised learning. All others are used for contrastive learning.

evaluations (but keeping the default temperature 0.1).

3.2. SupCon vs NT-Xent vs N-pair vs triplet vs pair losses

To compare SupCon with other contrastive loss functions, we replaced SupCon loss with pair, triplet, N-pair, or NT-Xent losses. In Table 4, results for supervised learning with CE loss and SCL with SupCon are copied from Section 3.1. Self-supervised learning using NT-Xent and N-pair losses performed significantly worse than CE or SupCon, likely due to insufficient training at 40 epochs. For triplet (with semi hard negative mining) and pair losses, we used default settings on the miner. While triplet loss produced a comparable result as SupCon, pair loss performed poorly, emphasizing the importance of positive/negative data mining. We noticed that except for CE supervised learning, all contrastive learning curves are almost flat (Figure 2). This makes sense because the evaluation stage (Stage 2) is only a linear classifier. If we get a good embedding from contrastive learning (Stage 1), we could use as few as 1-2 epochs to train the classifier to get good results. The downside is that Stage 1 takes about 5 times longer to train than CE supervised learning.

We projected the embeddings learned onto a 2-dimensional space by PCA and t-SNE to see if the models learned any meaningful representations during training (Figure 3). For CE supervised learning, it is the output before the last fully connected layer. For all contrastive learning losses, it is the output h of the encoder during pretraining. PCA projections generally show overlapping clusters of the labels, even for SupCon whose embedding resulted in the best classification accuracy. This is expected as PCA only tries to maximize the variance of the data. T-SNE, on the other hand, preserves pair-wise distances of the data. We saw 10 clusters of labels using t-SNE to project the CE embedding, and even tighter clusters with SupCon and triplet loss embedding. This means that compared to standard supervised learning using CE, SCL (using SupCon or triplet loss) indeed learned a more distinctive embedding between the labels, which helped the Stage 2 linear classifier get a higher accuracy. All other contrastive learning losses (i.e., pair, N-pair, and NT-Xent) did not learn embeddings that can be mapped to distinctive 2-dimensional clusters using t-SNE, consistent with their inferior classification accuracies.

3.3. No projector vs linear projector vs MLP projector

In SimCLR [1], the authors experimented with different architectures of the projector network and found that a simple 2-layer MLP worked better than a linear layer, which was better than no projector at all. The SupCon paper did not experiment with different projector architectures, which prompted us to do so. Our results (Table 5) show that projector architecture did not affect classification accuracy much. Changes from the original framework (2-layer MLP) accuracy (90.95%) to no projector (90.93%) or a linear projector (90.57%) were minimal, suggesting the existence of a projector may not be as important as in a self-supervised contrastive learning setting.

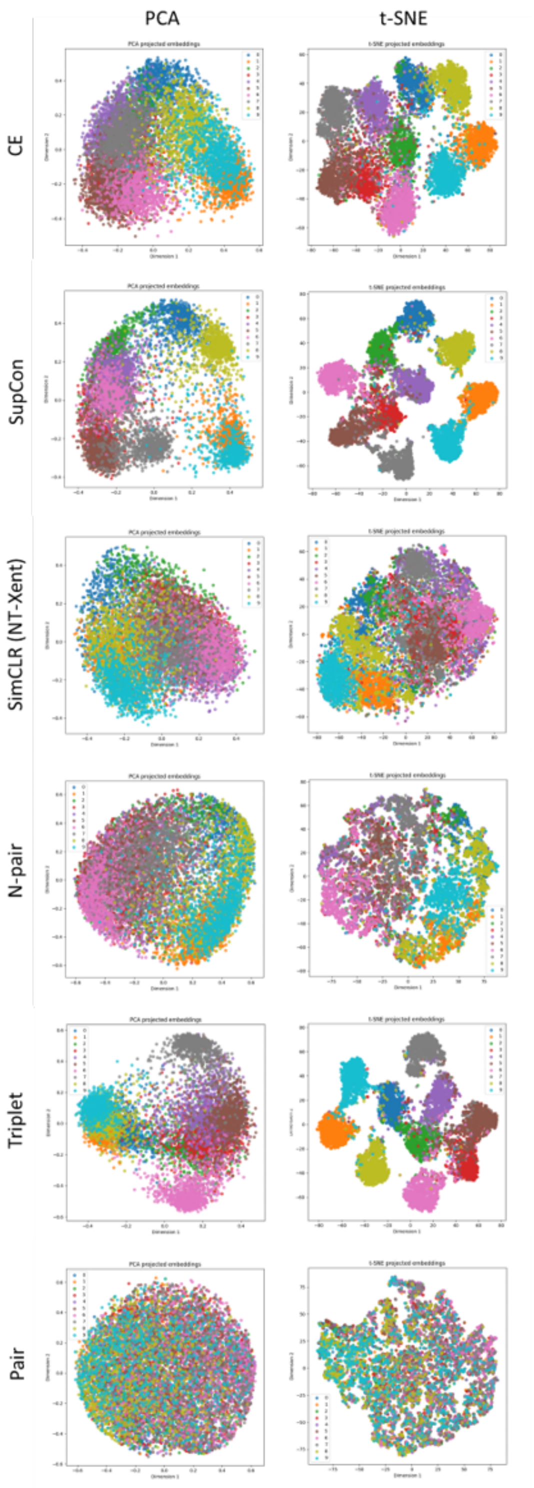

Figure 3. 2-dimensional PCA and t-SNE projections of the embeddings learned using different loss functions

3.4. Normalization vs no normalization on the encoder and projector networks

Although the SupCon paper mentioned in their framework the output of the encoder was normalized to improve the classification accuracy, we found in their PyTorch code implementation that the encoder output was not normalized (only the projector output was normalized). After experimenting with normalized vs unnormalized encoder/projector output, we did not see much effect of the normalization except when neither of the networks were normalized we got nan loss (Table 5). The reason for that is because SupCon uses dot products as the scoring function and taking the exponential of the dot product may produce a very large number. Normalizing either the encoder or the projector output (or using cosine similarity instead of dot product as the scoring function) is enough to solve the problem. Although we did not see the importance of projector normalization in our experiments as [1] has shown, Sections 3.3 and 3.4 collectively show that SupCon is stable to changes in projector architecture and encoder/projector normalization (provided using cosine similarity), in line with the observation that SupCon is less sensitive to a range of hyperparameters [8].

3.5. Data augmentation vs no augmentation

Finally, we tested the effect of data augmentation on the performance of SCL. Deep learning for computer vision tasks relies heavily on data augmentation to avoid overfitting and enhance performance [3, 12, 23]. This is especially imperative for self-supervised contrastive learning, where an image needs the augmentation of itself to form a positive pair during training. The composition of data augmentations is also critical [1]. In SCL, since label information is already available and therefore positive pairs can be formed between images with the same label without augmented data, we tested a framework without data augmentation. As shown in Table 5, SCL without data augmentation only had an 82.18% classification accuracy, indicating data augmentation is important to SCL. However, compared to supervised learning (Figure 3 first row, we copied the t-SNE projection in Figure 4 (left) for comparison convenience), SCL without data augmentation showed more distinctive, tighter clusters of the embeddings, suggesting it may perform better in some tasks based on distances. Indeed, when we tested a simple k-nearest neighbor classifier using the embeddings from these two frameworks, SCL without data augmentation resulted in a higher accuracy (83.99% vs 82.42%). Note that this comparison was unfair to SCL without data augmentation because in supervised learning we performed data augmentation. Together these results suggest although data augmentation can boost SCL performance, in situations where data augmentation is more challenging (e.g., tabular data), embeddings learned

SCL framework modification	Accuracy
No projector	90.93
Linear projector	90.57
Normalized encoder, normalized projector	89.62
Normalized encoder, unnormalized projector	90.92
Unnormalized encoder, unnormalized projector	nan loss
Unnormalized encoder, unnormalized projector, cosine similarity scoring function	90.5
No Augmentation	82.18

Table 5. Classification accuracy with modified SCL frameworks. The original framework setting was normalized MLP projector, unnormalized encoder, and with augmentation. For comparison, the validation accuracy was 90.95% as shown in Table 4.

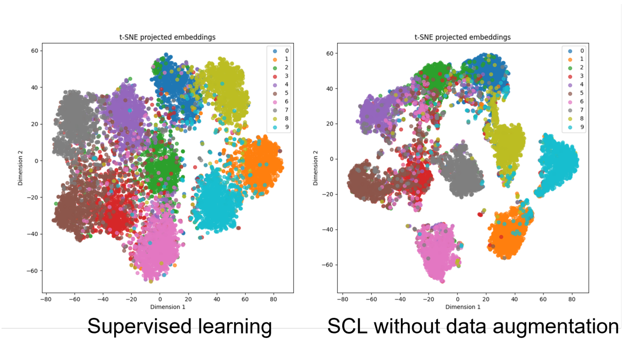

Figure 4. 2-dimensional t-SNE projections of the embeddings learned with supervised learning and SCL without data augmentation

from SCL even without data augmentation can be more informative than embeddings learned from supervised learning in tasks involving distance metrics.

4. Discussion

In this work, we explore variations of supervised contrastive learning and evaluate the impact on network accuracies by the choice of the loss function, adoption of a projector or a normalization layer, and data augmentation. We carefully design our experiments and interpret network performance strictly from an empirical standpoint. We successfully replicated the SupCon paper and found that SupCon loss is superior to CE and other contrastive loss functions with regards to feature representation learning and driving better classification accuracies. Our new finding is that the projector network, while found significant in SimCLR [1], does not affect SupCon performance as much. Normalization of the encoder and projector outputs also have small impact on SupCon, which is slightly different than conclusions of the SimCLR paper [1]. Finally, while data augmentation is important for SupCon to enhance feature representation learning, embeddings learned using SupCon without data augmentation may still find use cases in situations where augmenting data is more challenging.

One limitation of our work is that we used the same set of hyperparameters that is optimal to the baseline networks throughout the various comparison experiments, mainly due to limited computational resources. While we think increasing epoch is not likely going to alter our conclusions, it is possible that the best accuracies are achieved under different hyperparameters when comparing different variations of the network. For example, adding a normalization layer may warrant a higher learning rate which could have implications for batch size, and hence additional tuning after introducing the normalization layer in theory may result in better accuracies.

Another limitation is that we only focused on the best accuracies across variations. Practically, computational resources may be scarce, and the amount of training may be limited (as in the case of this project, for example). A framework still has a practical value if it trains more efficiently or converges faster even if it has a marginally lower accuracy.

Student	Contributed details
Xiaofei Chen	Tweaked the original code to run on Colab (detailed changes see top comments in code). Ran Section 3.2 - 3.5 experiments
Simin Liu	Ran Section 3.1 experiments on SupCon and hyperparameter tuning. Added code to capture performance statistics and learning curves (see main_linear_w_output.py)
Zhikang Dong	Implemented loss functions (see main_triplet.py in code)
Yunkai Wang	Ran Section 3.1 experiments on CE supervised learning hyperparameter tuning

Table 6. Contributions of team members

5. Work Division

Contributions of team members are provided in Table 6.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 2, 4, 5, 6

[2] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. 1

[3] Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp, 2021. 5

[4] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. 1

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. 1

[6] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2019. 1

[7] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding, 2020. 1

[8] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. 1, 5

[9] Taeuk Kim, Kang Min Yoo, and Sang goo Lee. Self-guided contrastive learning for bert sentence representations, 2021. 1

[10] Guillaume LORRE, Jaonary RABARISOA, Astrid ORCESI, Samia AINOUZ, and Stephane CANU. Temporal contrastive pretraining for video action recognition. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 651–659, 2020. 1

[11] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations, 2019. 1

[12] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022. 5

[13] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning, 2021. 1

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. 2, 3

[15] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video, 2018. 1

[16] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1857–1865, Red Hook, NY, USA, 2016. Curran Associates Inc. 2

[17] Li Tao, Xueting Wang, and Toshihiko Yamasaki. Self-supervised video representation learning using inter-intra contrastive framework. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, oct 2020. 1

[18] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding, 2020. 1

[19] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. 1

[20] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2022. 3

[21] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, jun 2009. 2

[22] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 1

[23] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey, 2022. 5