Semi-supervised Learning Methods for Image Classification: A Comparative Study

Rui Xiao rxiao65@gatech.edu
Xiaoyi Hu xhu349@gatech.edu
Jince Shi jshi390@gatech.edu
Zhanxu Liu zliu943@gatech.edu

Abstract

Obtaining large amounts of labeled data for image classification is expensive. Semi-supervised learning addresses this by leveraging both labeled and unlabeled data. We analyze four methods—FixMatch, MixMatch, CPC v2, and MoCo—on CIFAR-10, using supervised ResNet-18 as baseline. We compare performance under varying label ratios (1%, 5%, 10%), and investigate the effects of confidence thresholds (FixMatch), mixing parameters (MixMatch), contrastive losses (CPC v2), and memory bank sizes (MoCo).

1. Introduction/Background/Motivation

Image classification is a fundamental task in computer vision. The most common approach is supervised learning; however, obtaining a large amount of labeled data is often expensive and time-consuming. In this analysis, we explore semi-supervised learning methods that combine a small amount of labeled data with a large amount of unlabeled data to achieve comparable image classification performance.

The primary challenge we address is the limited availability of labeled data. By employing semi-supervised learning techniques, we aim to reduce data annotation costs while maintaining or enhancing model performance.

Image classification can typically be approached through four primary methods, ordered from the most commonly used to the least: supervised learning, transfer learning, semi-supervised learning, and unsupervised learning.

Supervised learning trains models on large labeled datasets (e.g., ImageNet, CIFAR-10) but is costly and prone to overfitting when labels are limited.

Transfer learning uses pre-trained models (e.g., VGG, Inception) fine-tuned on new tasks, reducing training time but often struggling if the target data distribution differs significantly.

Unsupervised learning focuses on clustering and representation learning but lacks direct classification capabilities.

Semi-supervised learning This method combines labeled and unlabeled data for classification tasks. Techniques like FixMatch and MixMatch are employed to effectively leverage the unlabeled data. The limitations of semi-supervised learning include: (1) the model's performance heavily depends on the quality of the labeled data, as poorly labeled data can lead to incorrect predictions. (2) the performance is also influenced by the unlabeled data, which may introduce noise; and (3) the complexity of the model is often higher than that of traditional supervised learning.

Leveraging unlabeled data has become increasingly important due to the high cost of manual annotation. This study evaluates semi-supervised methods to show that models trained with limited labeled data can achieve competitive classification performance. By optimizing leading algorithms and measuring trade-offs between accuracy and training time, we aim to identify strategies that are effective when both labels and compute resources are constrained. Our findings can broaden the applicability of deep learning to domains where labeled data is scarce and provide insights for advancing semi-supervised learning research.

We use the CIFAR-10 dataset, a widely adopted benchmark consisting of 60,000 color images (32×32 pixels) across 10 classes. To simulate varying annotation budgets, we randomly sample different proportions of the training set as labeled data, with the rest treated as unlabeled. The test set remains fixed for evaluation, enabling direct comparison to existing literature.

2. Approach

We benchmark four representative semi-supervised frameworks: FixMatch[12], MixMatch[1], CPC v2[8], and MoCo[7] on CIFAR-10 with sparse label budgets of 1%, 5%, and 10%. Standard supervised learning serves as the baseline, utilizing the same ResNet-18 architecture across all methods to ensure a fair comparison. We extend these methods by exploring different confidence threshold strate-

gies for FixMatch, various data mixing ratios for MixMatch, the impact of contrastive loss functions on representation quality for CPC v2, and the effect of memory bank sizes on MoCo's representation quality.

Given that semi-supervised learning methods leverage unlabeled data, we expect their accuracy in image classification to be comparable or exceed that of the baseline.

To establish performance benchmarks, we implemented a supervised baseline model trained solely on the labeled portion of the dataset. This model follows a standard supervised learning approach, utilizing ResNet-18 along with the Adam optimizer, an initial learning rate of 0.0001, and a batch size of 64. It was trained for 40 epochs using a standard cross entropy loss function without additional regularization, specifically across label ratios of 1%, 5%, and 10%. Additionally, the baseline model supports cross-platform GPU usage, ensuring consistent training conditions across all experiments.

For our MixMatch experiments, we based our implementation on the GitHub repository provided by [10]. Key modifications included: replacing Wide-ResNet with a custom ResNet-18 for fair comparison, fixing tensor reshaping issues in `utils/eval.py` to ensure PyTorch compatibility, and adding MPS backend support for Apple Silicon (M1) GPUs.

We expected MixMatch to perform strongly due to its integration of consistency regularization, entropy minimization, and mixup data augmentation, which theoretically provide strong supervision even with limited labeled data.

For our FixMatch experiments, we refered to the pytorch implementation of paper from [9]. To compare to other models while retaining FixMatch's semi-supervised benefits, we replaced the original Wide-ResNet-28-2 for a CIFAR-adapted ResNet-18 and ran a 3×3 grid over confidence thresholds τ 0.95, 0.90, 0.80 and label ratios 1 %, 5 %, 10 %.

Following the original setup [12], we utilized Exponential Moving Average to stabilize evaluation: EMA weights are applied only during evaluation to reduce prediction variance in noisy or semi-supervised settings. A cosine learning rate schedule was also adopted for training. Due to computational constraints, we limited all experiments to 50 epochs, compared to the original 1024 epochs, which we consider excessive for CIFAR-10 with ResNet-18. Final performance is reported as Top-1 CIFAR-10 accuracy after 50 epochs.

In our experiments with the CPC v2, we implemented efficient classification using a ResNet-18 model trained on labeled data, with features extracted from a pre-trained CPC v2 model that employs a ResNet-18 encoder. During the pre-training stage, we applied layer normalization and utilized patch-based augmentation. We randomly applied the color augmentations from Szegedy et al. (2014)[13] with a probability of 0.8. Using primitives from De Fauw et al. (2018)[6], we applied random elastic deformation and shearing with a probability of 0.2. We also randomly applied color-histogram augmentations with a probability of 0.2. While we generally maintained the CPC v2 architecture, we modified the contrastive loss functions. Although our dataset differs from that used in the original paper and we used a different ResNet variant for feature extraction, we could still roughly compare accuracy using 1% labeled data. Despite our efforts, the performance of CPC v2 was found to be comparable to that of the baseline model with 1% labeled data. We consider this is a success, as it indicates that the model is effectively learning from the data. However, several factors may have contributed to this result, including the smaller size of the encoder, the limited amount of unlabeled data, and the reduced image size, which made the extraction of meaningful patches difficult.

While the original MoCo v2 CIFAR-10 demo used a kNN monitor, we replaced it with a supervised linear classifier to enable more controlled and quantitative evaluation of representation quality under limited labels. This decision to adopt a linear probing protocol follows standard practice in contrastive learning [2, 7]. We also customized the augmentation pipeline, adjusting parameters beyond those in SimCLR [2] and MoCo v2 [11], aiming for stronger feature diversity and robustness. Although our model did not fully match baseline performance—likely due to fewer training epochs and limited labeled data—it still demonstrated meaningful representation learning, with consistent accuracy gains and stable convergence.

Posted Challenges In our implementation of semi-supervised models, we anticipated substantial GPU memory consumption and long training times. To mitigate these challenges, we adopted ResNet-18 as the encoder, preserving core method optimizations while significantly reducing resource demands.

During implementation, we encountered several model-specific challenges: For MixMatch, we experienced numerical instability when using exactly 5% labeled data (2500 samples), consistently resulting in NaN values during training. To resolve this, we slightly increased the number of labeled samples to 2624. Additionally, MixMatch was highly sensitive to the Beta distribution parameter α and sharpening temperature T, requiring careful tuning. We implemented gradient clipping and refined probability handling in the loss calculation to stabilize training.

FixMatch follows a simple design but requires prolonged training. With limited computational resources, we implemented necessary optimizations based on the PyTorch framework and reduced the number of training epochs to accelerate experimentation.

Data preprocessing posed specific challenges for CPC

v2, particularly patch extraction. While the original paper utilized high-resolution ImageNet images (224×224), CIFAR-10 images (32×32) lacked sufficient detail. To address this, we resized images to 300×300, followed by random cropping to 260×260. Additionally, we found that publicly available reimplementations of CPC v2 were often inaccurate or lacked critical optimizations described in the original work. Consequently, we reimplemented CPC v2 from scratch to ensure fidelity to the intended design.

With MoCo Model, our objective was to evaluate how memory bank size impacts MoCo representation quality under varying labeled data ratios (1%, 5%, 10%). Although the original MoCo paper discussed the use of a linear classifier as a standard evaluation protocol to assess representation quality, the publicly available ResNet-18 demo implementation relied instead on a kNN-based monitor for evaluation, rather than a supervised classifier. Although more recent work, such as UniMoCo [5], has extended the MoCo framework to simultaneously address unsupervised, semi-supervised, and fully supervised scenarios in a unified manner, we chose to remain to the original MoCo v2 [4] architecture and manually incorporated a supervised linear classifier for our evaluations. We anticipated that adapting the MoCo demo code to support linear probing would not be straightforward, since the original implementation was optimized exclusively for kNN-based evaluation and did not readily provide hooks to train a supervised classifier. In addition, we expected that training the linear classifier with only a small subset of labeled data (particularly 1%) could lead to instability, requiring careful tuning of hyperparameters such as the learning rate, batch size, and number of training epochs. As anticipated, initial attempts revealed that the model exhibited slow convergence and high variance in validation performance across epochs when trained with limited labeled data.

3. Experiments and Results

FixMatch For our FixMatch experiments, success is defined as matching the original Wide-ResNet-28-2 accuracy ($\approx$ 95% with 10% labels) while cutting model size and training cost.

Table 1. Test accuracy (%) of FixMatch using different confidence thresholds and labeled data ratios.

Threshold	1% Labels	5% Labels	10% Labels
0.95	93.15	94.20	**94.99**
0.90	93.59	**95.17**	94.47
0.80	93.81	94.30	**94.95**

At 10% labels, a test accuracy of 94.99% ($\tau = 0.95$) effectively reproduces the original 95%, demonstrating that the backbone replacement with ResNet-18 can approximately reproduce original performance under resource constraints.

Even with only 1% labeled data, the model consistently achieves over 93% accuracy across thresholds, confirming that label-efficiency is preserved. All τ values keep 1 %-label accuracy $>$ 93 %, showing ResNet-18 preserves FixMatch's label-efficiency.

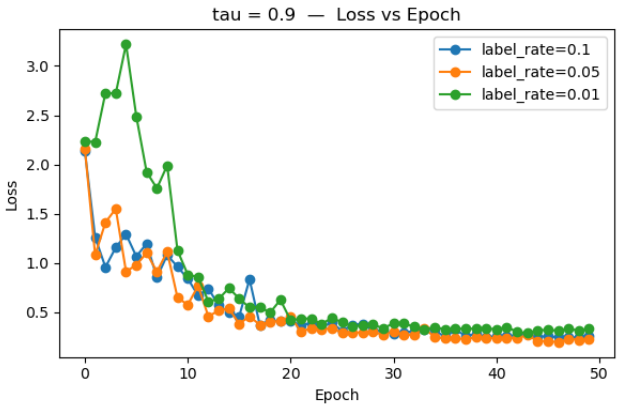

Figure 1. Loss curve under different label ratios with $\tau = 0.9$

Analyzing the loss dynamics reveals that all curves drop steeply within the first 5 epochs and then stabilize. Using the $\tau = 0.9$ results in Figure 1 as an example, the initial total loss scales with the amount of labeled data, following the trend 1% $>$ 5% $>$ 10%. However, by approximately epoch 25, the curves converge, suggesting that FixMatch remains robust under extremely low supervision once sufficient training progress is made. The total loss follows the standard FixMatch formulation:

$$\mathcal{L} = \mathcal{L}_x + \lambda_u \mathcal{L}_u$$

The FixMatch model on CIFAR-10 further reveals several important characteristics. First, a clear separation is observed between the behavior of labeled and pseudo-labeled data. During the early stages, the model must rely heavily on the limited labeled data while gradually improving the quality of pseudo-labels. Initially, the total loss is higher when labeled data is scarcer, but as training proceeds, the gaps diminish, and all loss curves almost overlap after epoch 25. This convergence indicates strong robustness even when using very few labels.

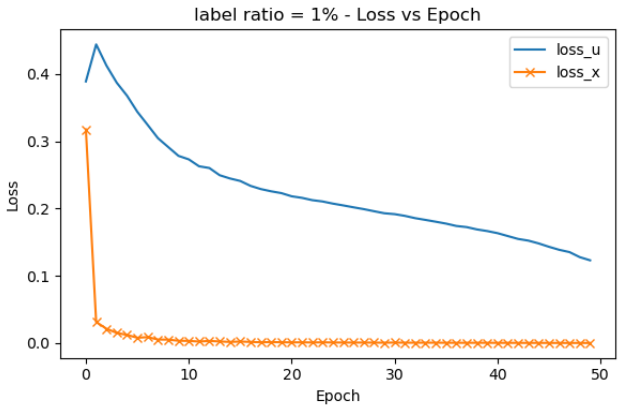

Figure 2. Cross-entropy loss ($\mathcal{L}_x$) and consistency loss ($\mathcal{L}_u$) during training with 1% labeled data.

Figure 2 shows the decomposition of the total loss into the supervised loss $\mathcal{L}_x$ and the unsupervised consistency loss $\mathcal{L}_u$. With fewer labels, the model begins with noisier pseudo-labels, leading to a higher initial $\mathcal{L}_u$. However, the model quickly compensates, reducing $\mathcal{L}_u$ significantly as pseudo-label quality improves. Once the small labeled set is fitted, further gains come almost entirely from minimizing the consistency loss.

Threshold sensitivity is observed to be related to the label ratio, but the relationship is not strictly monotonic. This suggests that finer threshold tuning could yield additional improvements. Finally, ResNet-18 demonstrates efficient convergence behavior, reaching a near-optimal region within 50 epochs, confirming that extended training is unnecessary under the current setup.

CPC v2 We found that CPC v2 (with ResNet-18 encoder) was comparable to the baseline when only 1% of the labeled data was used, as shown in Table 5. This indicates that this architecture is effective in low-data scenarios. However, as the amount of labeled data increased, the advantage of this self-supervised pre-training decreased, and its performance fell below the of the baseline. We believe this can be attributed to several factors:

1. Encoder Size: The encoder's smaller size, while beneficial for faster training and inference, may limit its capacity to extract complex features. This trade-off between speed, memory and model capacity could hinder performance as more labeled data becomes available.

2. Dataset Characteristics: The CIFAR-10 dataset consists of relatively small images. In the CPC v2 framework, we first extract patches from these small images. Given the limited size of the images, the benefits of the CPC v2 architecture may not fully realized, as the model might struggle to learn meaningful representations from such small patches.

The contrastive loss function used in CPC is InfoNCE, and we also tested the impact of different contrastive loss functions: NT-Xent[3] and triplet loss.

The NT-Xent loss function is quite similar to InfoNCE, but includes a temperature parameter to control the sharpness of the distribution of similarity. In contrast, triplet loss is quite different from InfoNCE. It compares an anchor, a positive sample, and a negative sample, ensuring that the distance between the anchor and positive sample is smaller than that between the anchor and negative sample by a margin. Both NT-Xent and triplet loss introduces an additional parameter compared to InfoNCE.

The accuracy of efficient classification (ResNet-18) using 1% labeled data, with features extracted using the CPC v2 architecture and employing all three contrastive loss functions is shown in Table 2. As indicated in the table, the model utilizing NT-Xent achieved the highest accuracy, while the accuracy obtained with triplet loss was the lowest.

Reasons for observations: The superior performance of NT-Xent can be due to its normalization of similarity scores, which enhances the model's ability to differentiate between positive and negative pairs, resulting in more robust feature representations. Thus, the highest accuracy achieved by NT-Xent aligns with expectations.

The lower accuracy of triplet loss could be considered to be somewhat expected. Triplet loss requires careful selection of anchor, positive, and negative samples. This sensitivity to sample selection may hinder the model's ability to learn meaningful embeddings. In this case, InfoNCE and NT-Xent may have been more effective due to their ability to leverage a larger number of negative samples, which can help the model learn more robust representations.

Table 2. Test accuracy (%) of ResNet-18 classifier using 1% labeled CIFAR-10 data, with features extracted by CPC v2 under different contrastive losses.

Contrastive Loss	**Accuracy**
InfoNCE	30.05%
NT-Xent	32.44%
Triplet Loss	28.71%

MixMatch For our experiments, we measured the effectiveness of MixMatch across different label ratios (1%, 5%, and 10%) on the CIFAR-10 dataset. Each configuration was trained for 40 epochs with a learning rate of 0.0001, using key hyperparameters $\alpha = 1.0$ for the Beta distribution mixing coefficient and $\lambda_u = 1.0$ for weighting the unsupervised loss component. These values were chosen to balance the contribution of labeled and unlabeled data during training.

The results, summarized in Table 3, revealed several interesting patterns. First, contrary to expectations, increasing the amount of labeled data from 1% to 10% did not lead to consistent improvements in accuracy. The best performance was observed with 5% labeled data, achieving 16.21% test accuracy, while both 1% and 10% labeled data yielded slightly lower performance.

Table 3. Test accuracy (%) of MixMatch using different labeled data ratios after 40 epochs.

Metric	1%	5%	10%
Valid Acc.	15.58	16.14	15.46
Test Acc.	15.38	16.21	15.67
Train Loss	2.24	2.30	2.33
Loss X	2.18	2.24	2.27
Loss U	0.057	0.057	0.059

Analysis of the loss components shows that the supervised loss (Loss X) remained relatively stable across different label ratios, ranging from 2.18 to 2.27. Similarly, the unsupervised consistency loss (Loss U) also remained consistent, around 0.057-0.059 across all configurations. This suggests that the MixMatch algorithm maintained similar learning dynamics regardless of the amount of labeled data available.

The modest performance of MixMatch in our experiments can be attributed to several factors:

1. Training Duration: The original MixMatch paper reported results after substantially longer training (1024 epochs), while our experiments were limited to 40 epochs due to computational constraints.

2. Hyperparameter Sensitivity: MixMatch's performance is known to be sensitive to its mixing hyperparameters (α) and sharpening temperature (T). Our fixed hyperparameter settings may not have been optimal for the ResNet-18 architecture.

3. Learning Rate Schedule: We used a constant learning rate rather than a more sophisticated schedule, which may have limited the model's ability to converge to better solutions.

4. Architectural Differences: While we maintained the core MixMatch algorithm, adapting it to ResNet-18 from the original Wide-ResNet architecture may have influenced the model's capacity to effectively leverage the unlabeled data.

Despite these challenges, our implementation provides valuable insights into MixMatch's behavior with different label ratios. The consistency of the unsupervised loss component across different labeled data percentages suggests that the algorithm's ability to incorporate unlabeled data remains stable even as the amount of labeled data changes.
MoCo We evaluated the effectiveness of our MoCo-based representation learning framework by replacing the standard kNN monitor with a supervised linear classifier. The original MoCo demo [7, 11] reported an accuracy of 85.3% using a monitor based on kNN on the test set. However, to study the influence of labeled data availability and to enable fair comparison with semi-supervised methods (e.g., FixMatch, MixMatch), we implemented a linear classifier trained with different percentages of labeled data.

The success of this work was measured by the quality of representations learned by the MoCo encoder in a fully unsupervised setting. We evaluated representation quality through linear probing—a widely adopted protocol in contrastive learning. Specifically, a linear classifier was trained on top of frozen MoCo features using only a small subset of labeled CIFAR-10 training data (1%, 5%, 10%), and evaluated on the full CIFAR-10 test set. The goal was to determine whether MoCo's unsupervised representations were linearly separable under low-label regimes.

Table 4. Test accuracy (%) of linear classifier trained on top of MoCo encoder using different queue sizes and labeled data ratios.

Queue Size	1% Labels	5% Labels	10% Labels
2048	66.45	71.33	72.32
4096	67.00	71.50	72.85
8192	**67.46**	**72.17**	**73.03**
16384	65.99	71.43	72.30

To further understand the behavior of MoCo under varying contrastive learning conditions, we investigated the impact of different queue sizes, a core hyperparameter in the MoCo framework, on representation quality. The queue stores negative samples for contrastive learning, and its length determines both the diversity and the temporal recency of those samples. We trained MoCo with queue sizes of 2048, 4096, 8192, and 16384 while holding all other hyperparameters constant, and evaluated the learned representations using the same linear probing setup across the 1%, 5%, and 10% labeled data. As shown in Table 4, increasing the queue size from 2048 to 8192 consistently improved the accuracy of the linear classifier, with 8192 producing the highest performance across all data regimes labeled. For example, at 1% labeled data, accuracy increased from 66.45% (queue=2048) to 67.46% (queue=8192), and at 10% labeled data, from 72.32% to 73.03%. However, further increasing the queue to 16384 resulted in slightly degraded performance, indicating that excessively large queues may introduce stale or noisy negatives, diminishing the benefit of contrastive diversity.

In addition, we replicated the training environment and implementation used in the official MoCo demo [7, 11], which evaluated the quality of the representation using a nonparametric kNN monitor. Without using labeled data during training or evaluation, their model achieved 78.9% test precision after 120 epochs. In contrast, our evaluation used a supervised linear classifier trained on top of frozen MoCo features using 10% of labeled training data, achieving a best accuracy of 73.03%. Although our results appear slightly lower in magnitude, they offer a more rigorous and controlled analysis of feature quality under partial supervision and are directly comparable to semi-supervised baselines such as FixMatch and MixMatch. These findings confirm that MoCo learns strong general-purpose features without labels, and that such representations can be effec-

tively adapted to low-label supervised tasks.

Table 5. The accuracy (%) of image classification is evaluated using various methods. ResNet-18 is trained on raw image pixels, representing the supervised learning method. All other semi-supervised methods also utilize the ResNet-18 architecture. Notably, in CPC v2, ResNet-18 is trained on the proposed CPC v2 features, which are derived from the ResNet-18 encoder.

Methods	1% Labels	5% Labels	10% Labels
ResNet-18	42.04	60.78	70.38
FixMatch	93.15	94.2	94.99
MixMatch	15.38	16.21	15.67
CPC v2	30.05	37.26	39.47
MoCo	67.46	72.17	73.03

4. Other Sections

Deep Learing framework and whether overfit We used PyTorch in this analysis. Throughout our experiments, We found no strong evidence of overfitting. The loss curves and test accuracy trends indicate that the performance on the test set is comparable to that on the training set, suggesting that the model generalizes well.

Hyper-parameters and learnable parameters Key hyper-parameters of FixMatch included the confidence threshold (τ), number of training epochs, learning rate, and batch size. We systematically explored different thresholds (0.8, 0.9, 0.95) to optimize pseudo-label quality without introducing excessive noise. A standard SGD optimizer with momentum was used for training. Reducing the total number of epochs from 1024 to 50 proved sufficient when using the ResNet-18 backbone, balancing training cost and convergence speed.

The contrastive Predictive Coding method involves two training stages: pre-training and efficient classification. During the pre-training stage, an encoder (often a ResNet architecture) is used with a PixelCNN. In this phase, both the encoder and the PixelCNN have learnable parameters. Once pre-trainig is complete, the weights of the encoder are frozen, and the model transitions to the classification stage. ResNet-18 in our analysis is employed after the encoder for image classification. In this stage, only the layers of ResNet-18 have learnable parameters.

For our MixMatch experiments, we measured MixMatch's effectiveness across different label ratios (1%, 5%, and 10%) on CIFAR-10. Each configuration trained for 40 epochs with learning rate 0.0001, using Beta distribution parameter $\alpha = 1.0$ and unsupervised loss weight $\lambda_u = 1.0$. We employed SGD optimizer with momentum (0.9) and weight decay (5e-4).

Starting Codebases and Modifications For FixMatch, our implementation was based on an unofficial pytorch implementation[9], which we checked against the structure described in the original papers [12]. To accommodate computational constraints, we modified the backbone selection, adjusted loss scaling, and refined the handling of label expansion.

Our MixMatch implementation started with the GitHub repository [10], but required significant modifications: adding a custom ResNet-18 architecture, modifying training code for compatibility, fixing tensor handling issues by replacing .view(-1) with .reshape(-1), adding Apple Silicon GPU support, and implementing stability enhancements like gradient clipping (max norm 5.0).

For CPC v2, we initially attempted to use publicly available reimplementation code. However, none of the available versions were fully functional or correctly incorporated the CPC v2 optimizations. Most resembled CPC v1 architectures but diverged significantly from the intended design of CPC v2. Consequently, we reimplemented the CPC v2 framework from scratch to ensure fidelity to the original methodology.

For MoCo v2, we started with the official MoCo repository [11], specifically leveraging the CIFAR-10 demo adapted for Colab GPU environments. The provided codebase included a complete implementation of the core momentum contrast mechanism, encoder setup, and a kNN-based monitor for unsupervised evaluation. However, the repository did not provide a supervised linear classifier for representation evaluation. To address this, we extended the codebase by implementing a supervised linear probing protocol on top of the frozen MoCo encoder.

5. Work Division

Student Name	Contributed Aspects	Details
Rui Xiao	implementing CPC v2 (with ResNet-18 encoder) and experimenting with constrastive loss functions.	wrote the code for CPC v2 with ResNet-18 encoder from scrach, and tested the impact of the contrastive loss functions on the accuracy of efficient classification using 1% labeled data.
Xiaoyi Hu	Implemented FixMatch framework and ran trials	Built the FixMatch training pipeline with a ResNet-18 backbone, executed trials varying label ratio and confidence threshold
Jince Shi	Implemented MixMatch method and baseline model	Modified the MixMatch framework to use ResNet-18 architecture, conducted experiments with different label percentages (1%, 5%, 10%)
Zhanxu Liu	Implemented MoCo method and results analysis	Built the MoCo representation learning framework with a linear classifier for evaluation, conducted experiments with different queue sizes, and analyzed representation quality under varying labeled data ratios

Table 6. Contributions of team members.

References

[1] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *CoRR*, abs/1905.02249, 2019. 1

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. 2

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

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020. 3

[5] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Unimoco: Unsupervised, semi-supervised and full-supervised visual representation learning, 2021. 3

[6] Romera-Paredes B Nikolov S Tomasev N Blackwell S Askham H Glorot X O'Donoghue B Visentin D van den Driessche G Lakshminarayanan B Meyer C Mackinder F Bouton S Ayoub K Chopra R King D Karthikesalingam A Hughes CO Raine R Hughes J Sim DA Egan C Tufail A Montgomery H Hassabis D Rees G Back T Khaw PT Suleyman M Cornebise J Keane PA Ronneberger O. De Fauw J, Ledsam JR. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nat Med*, 2018 Sep. 2

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. 1, 2, 5

[8] 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

[9] Jungdae Kim. Pytorch implementation of fixmatch. https://github.com/kekmodel/FixMatch-pytorch, 2020. 2, 6

[10] Yu-Min Liang. Mixmatch-pytorch. https://github.com/YU1ut/MixMatch-pytorch, 2020. 2, 6

[11] Facebook AI Research. Moco: Momentum contrast for unsupervised visual representation learning. https://github.com/facebookresearch/moco, 2020. 2, 5, 6

[12] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *CoRR*, abs/2001.07685, 2020. 1, 2, 6

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. 2