Self-Supervised and Representation Learning

Sam Obeng[2,3*] **Tony Nunn**[1,2,*] **Hakeem Shitta-Bey**[1,2,3,*]

[1]Department of Biomedical Engineering, Duke University
[2]Department of Computer Science, Duke University
[3]Department of Electrical and Computer Engineering, Duke University
[*]Equal contribution

Abstract

In scenarios where data labels are scarce, self-supervised learning uses the learned features from unlabeled data to offer a competitive alternative to traditional supervised learning. We investigate the effectiveness and compatibility of two prominent self-supervised models: Simple Framework for Contrastive Learning of Visual Representations (SimCLR) and Rotation Prediction Network (RotNet). We evaluated their performance on the CIFAR-10 dataset and determined their robustness through a series of semi-supervised scenarios with 1%, 10%, and 50% of the data labels available. Additionally, we attempted to leverage the advantages of contrastive learning and rotation prediction by creating a combined, hybrid model and dynamically weighing their contributions during training. Our findings show that SimCLR and RotNet consistently outperform fully supervised learning techniques when in low-label scenarios; however, when combined, their objective feature spaces are nearly opposites, which results in their objective functions conflicting.

1 Intro

Despite major advances in deep learning, many models still rely heavily on large labeled datasets. Creating these labels can be expensive and overly time-consuming. In many cases, data labeling becomes the main bottleneck in scaling deep learning systems. Self-supervised learning (SSL) offers a promising alternative by removing the need for manual labels and instead allowing models to learn directly from the data itself through cleverly designed pretext tasks.

In this work, we focus on evaluating and comparing two SSL methods that approach representation learning in different ways: SimCLR, which uses contrastive learning, and RotNet, which learns by predicting image rotations. Our goal is to better understand how each model performs under varying conditions and how well the learned features transfer when labeled data is limited. We also aim to explore whether different SSL approaches offer unique advantages, and if so, whether those strengths can be combined in a single model.

To investigate this, we pretrain SimCLR and RotNet encoders on unlabeled CIFAR-10 images and evaluate them under semi-supervised settings using 1%, 10%, and 50% of labeled data. Finally, we explore a hybrid approach that combines both methods using a shared encoder and a weighted joint loss, with the aim of leveraging their complementary strengths.

2 Related Works

Recent works in self-supervised learning demonstrate success in learning representations without data labels. Contrastive learning and discriminative pretext tasks are two approaches that are most prominent in the field. SimCLR introduced by Chen et al. [2] uses contrastive learning. It leverages the normalized temperature-scaled cross-entropy loss (NT-Xent) to maximize agreement between different augmented views of the same image. Through strong data augmentations and a 2-layer multilayer perceptron (MLP) head, the encoder backbone promotes instance-level feature learning. SimCLR has shown very promising results in image classification tasks, even surpassing the performance of fully supervised models in scenarios with limited data labels. RotNet introduced by Gidaris et al. [4] uses a pretext task SSL approach instead. The model is trained to classify which rotation (0°, 90°, 180°, and 270°) has been applied to an image. Deceptively simple, the model is designed to learn global semantic patterns that translates to object-level feature representation. Shape and orientation allow RotNet to extend well into classification and detection tasks. Despite their differences, these approaches show practical efficacy in the SSL field: instance-level contrastive learning and object-level pretext tasks. Both methods have inspired an array of follow-up research. For example, Big Self-Supervised Models are Strong Semi-Supervised Learners by Chen et al [3] and Deep Clustering for Unsupervised Learning of Visual Features by Caron et al. [1]. Altogether, these works show promise in the SSL and representation learning field.

3 Methods

3.1 Dataset and Preprocessing

We used the CIFAR-10 dataset, which consists of 60,000 32x32 color images. 50,000 of these images are used for training and 10,000 are used for testing. The only preprocessing performed was mean-std normalization of the data as part of the SimCLR or RotNet training pipelines and discarding the labels of the data for all pretraining tasks in order to ensure self-supervised learning. So, all training was done directly on the raw 32x32 RGB images from the CIFAR-10 dataset.

3.2 Model Architecture

3.2.1 SimCLR

SimCLR is a contrastive learning model, which means it attempts to distinguish between similar and dissimilar pairs of data. Essentially, when given an image from the CIFAR-10 dataset, the model performs a series of random augmentations on this image, creating two versions of it, then attempts to maximize the agreement between these two versions (positive pair) while attempting to minimize the agreement between other sample images (negative pairs). By creating a separation of pairs in hyperspace, the model is able to distinguish between images in the absence of labels. We implement SimCLR using the framework provided by Chen et al.[2]. In general, the model consists of four components: data augmentation, a base encoder, a projection head, and a contrastive loss function.

Data Augmentations. As mentioned above, data augmentations are essential for SimCLR to form positive pairs. Our model performed a collection of random augmentations from the following list: horizontal flip, crop, color/brightness shift, and grayscale [2].

ResNet-50 Encoder Head. The ResNet-50 model serves as the base encoder for the backbone of our SimCLR model. We used the standard ResNet-50 architecture from the PyTorch library [6] with slight modifications outlined in Chen et. al [2]. ResNet-50 is designed for 224x224 images from the ImageNet database, so we start by replacing the standard 7x7 convolution layer (stride=2, padding=3) with a 3x3 convolution layer (stride=1, padding=1). This allows more spatial information to be learned from the smaller 32x32 CIFAR-10 images. Lastly, the max pooling layer after the first convolution and the final fully connected layer are removed completely from the model. Removing the first max pooling layer is another byproduct of using smaller images because we want to avoid early down-sampling, which could potentially discard small important features. The fully connected layer is removed because SimCLR does not need a classifier during pretraining, instead the appropriate projection head is attached.

Projection Head. SimCLR uses a projection head to map features to a contrastive space. We use a 2-layer multilayer perceptron (MLP) as our projection head. The ResNet-50 encoder outputs a 2048 dimensional vector, which is then transformed into a lower dimensional space, 128 dimensions, so the contrastive loss function can be more effectively applied. The projection head architecture is simple: the input is taken in, fed through a ReLU layer to introduce some non-linearity, then projected down to 128-dimensional space using a fully connected layer. Current studies [2] demonstrate that introducing non-linearity at this stage leads to an improvement in representation learning.

Training. We train the SimCLR model for 200 epochs across three different batch sizes (64, 128, and 256) and three different learning rates (0.0125, 0.025, and 0.05) for a total of 9 separate training pipelines. We used the Normalized Temperature-Scaled Cross-Entropy Loss (NT-Xent) function [2]. This function encourages positive pairs (augmented views of the same image) to be similar, or grouped together, while pushing negative pairs away in the projected feature space. The cross-entropy objective is depicted in Equation (1), and the final full loss function is depicted in Equation (2). For evaluation purposes, the model state was saved every 20 epochs, then once pretraining was complete, the projection head was replaced with a linear classifier to determine model performance.

For each anchor image i, the NT-Xent loss is computed via the cross-entropy objective:

$$\ell_i = -\log\left(\frac{\exp(\tilde{\mathbf{z}}_a^\top \tilde{\mathbf{z}}_b/\tau)}{\sum_{\substack{l=1 \\ l\neq i}}^{2N} \exp(\tilde{\mathbf{z}}_a^\top \tilde{\mathbf{z}}_b/\tau)}\right) \tag{1}$$

The total loss is the mean over all samples:

$$\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2N}\sum_{i=1}^{2N} \ell_i \tag{2}$$

3.2.2 RotNet

RotNet is a self-supervised learning model that learns visual representations by predicting the rotation applied to an image. When given an image from the CIFAR-10 dataset, the model randomly rotates the image by one of four possible angles: 0°, 90°, 180°, or 270°. The model is then tasked to identify the correct rotation angle. By predicting these rotations, RotNet forces the network to learn meaningful features about the structure and content of the images, even without using any manual pre-input labels about what the image actually contains. We implemented RotNet by following the framework provided by Gidaris et al.[4]. In general, the model consists of three main components: data transformation through rotation, a base encoder, and a rotation classification head trained with a standard cross-entropy loss.

RotNet ResNet-18 Based Model The ResNet-18 model serves as the base encoder for the backbone of our RotNet model. We used the standard ResNet-18 architecture from the PyTorch library [6] with slight modifications to the model and data. For the data, the CIFAR-10 images are resized to 224x224 before their rotation because the ResNet-18 backbone takes in 224x224 images by default. Additionally, the fully connected layer of the model is replaced with a simple linear layer with an in dimension of 512 and an out dimension of 4 to classify the 4 rotations.

Training. We train the Rotnet model with the standard cross-entropy loss from pytoch for 200 epochs across three different batch sizes (64, 128, and 256) and a learning rate of 0.001 for a total of 3 separate training pipelines. For evaluation purposes, the model state was saved every 20 epochs, then once pretraining was complete, the projection head was replaced with a L-BFGS optimized linear classifier to determine model performance.

3.2.3 SimCLR+RotNet Hybrid

We implement a lightweight hybrid model that combines SimCLR and RotNet, using a shared ResNet-18 encoder. The model is trained jointly using a weighted sum of the SimCLR and RotNet losses. After training, the encoder is frozen and a linear classifier is attached for evaluation on downstream classification tasks. In general, the model consists of three components: data augmentation, a base encoder, and a combined loss function.

Data Augmentations In the SimCLR+RotNet model, we apply task-specific augmentations aligned with each objective. SimCLR views follow a contrastive augmentation pipeline—random cropping, flipping, color jittering, grayscale conversion, and Gaussian blur—based on Chen et al. [2] with minor modifications. RotNet applies a fixed rotation without additional augmentations. Both views are processed by a shared encoder, with SimCLR features passed to a projection head for contrastive loss and RotNet features to a 4-way classifier. This design minimizes conflicts between contrastive and pretext learning signals.

Hybrid ResNet-18 Based Model The SimCLR+RotNet model uses a shared ResNet-18 encoder, modified for CIFAR-10 by replacing the 7×7, stride-2 convolution with a 3×3, stride-1 convolution and removing the max-pooling layer to better preserve spatial details. ResNet-18 is selected as a lightweight backbone to offset the computational cost of multiple self-supervised objectives. A projection head maps SimCLR embeddings into a contrastive space, while a 4-way classifier predicts image rotations for RotNet. NT-Xent loss and cross-entropy loss are applied to the outputs of the projection and rotation heads, respectively.

Training and Evaluation The model is trained for 200 epochs using an SGD optimizer and a batch size of 256. A warm-up scheduler increases the learning rate from 0.0025 to 0.025 over the first 10 epochs, followed by a cosine annealing scheduler that decays it to 0.0125 over the remaining 190 epochs. The total loss is a weighted sum of the SimCLR and RotNet losses, with RotNet's weight gradually reduced to zero by epoch 25. This scheduling mitigates conflicts between tasks, as RotNet converges early and may otherwise hinder SimCLR's representation learning. For evaluation, the model's encoder is frozen and the L-BFGS linear classifier head is attached.

3.3 Performance Evaluation

3.3.1 L-BFGS Linear Classifier Training

To evaluate the quality of the learned representations we trained a linear classifier over the frozen encoder outputs from our models. We used a Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [5] optimization function to train the linear classifier. Unlike first-order methods, like SGD, BFGS uses gradient information to inform a more rapid convergence. The model is trained in full-batch mode for 20 epochs using the cross-entropy loss.

Given a dataset of N samples with fixed feature vectors $\mathbf{x}_i$ with dimension d extracted from a frozen encoder and their corresponding labels $y_i \in \{0, 1, \ldots, 9\}$, we train a linear classifier defined as [5]:

$$\mathbf{z}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}$$

where $\mathbf{W} \in \mathbb{R}^{10 \times d}$ is the weight matrix, and $\mathbf{b} \in \mathbb{R}^{10}$ is the bias vector.

The output $\mathbf{z}_i$ is passed through a softmax function to produce class probabilities, then the cross-entropy loss is taken over the whole batch. Once trained via L-BFGS, the linear classifier determines top-1 accuracy by comparing the predicted labels to the test labels and returning the accuracy as a ratio of correct predictions over total test examples.

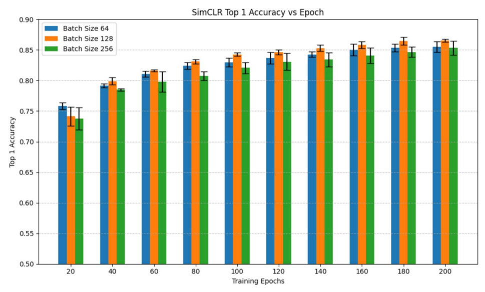
3.3.2 Pretraining for Semi-Supervised Models

The approach and dataset was slightly altered for experimenting with semi-supervised learning models. We tested three semi-supervised scenarios by controlling the amount of access the model has to the data: 1%, 10%, or 50%. In order to do this, we generate a number of random indices equal to the number of images desired, then use these indices and the torch.utils.data.Subset [7] package to get the wanted percentage of images from the CIFAR-10 dataset. It is important to note that SimCLR never sees these labels. Pretraining is still performed without labels entirely; however, now the linear classifier is being trained with limited access to labels.

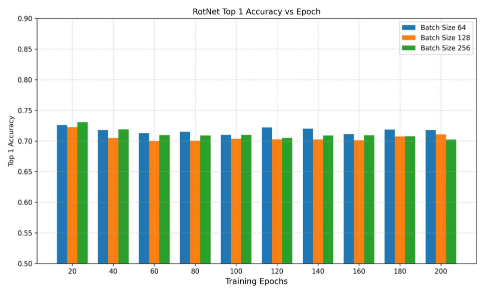
4 Results

4.1 Performance as a Function of Pretraining

To better understand the power or representation learning and determine the robustness of our chosen models, we evaluated their performance over a large sweep of different pretraining parameters. First, for the SimCLR model, as aforementioned, we performed nine pretraining pipelines: one for every combination of batch size (64, 128, and 256) and learning rate (0.0125, 0.025, and 0.05). For each pipeline, the encoder was frozen and saved every 20 epochs for 200 epochs to get evaluations across different lengths of pretraining. Figure 1a, show these results where each bar represents an average performance across all three learning rates and the error bars are the standard deviation.



(a) SimCLR



(b) RotNet

Figure 1: Effect of batch size and pretraining length on SimCLR and RotNet performance

The results in Figure 1a show that longer pretraining lengths and larger batch sizes lead to improved performance for SimCLR. Larger batch sizes translate to more data available, allowing more positive pairs to be formed and resulting in stronger feature representations. Similarly, increasing the number of epochs gives the model additional time to refine its representations. These experiments were repeated for RotNet; however, learning rate was not varied, as RotNet lacks the sensitivity to hyperparameters observed in SimCLR. Following Gidaris et al. [4] and related studies, we used a fixed learning rate without scheduling, given our relatively low total number of epochs (200). Figure 1b shows RotNet performance under varying batch sizes and pretraining lengths, with starkly different trends compared to SimCLR. RotNet focuses on global semantic understanding rather than instance-level features, leading to a lower peak performance. Moreover, RotNet exhibits a plateau or even slight decay in performance as training progresses. This behavior stems from the simplicity of its pretext task: as the model becomes proficient at predicting rotations, it overfits to this specific task rather than improving its general semantic understanding, consistent with observations in previous studies [4].

4.2 Semi-Supervised Learning Performance

Ultimately, to test the true effectiveness of our models we evaluated their performance in real-world scenarios and compared them to their fully-supervised model backbones. Figure 2 demonstrates that self-supervised models flourish when data labels are scarce. We observed stark performance differences between the SSL models and the fully supervised models when access to labels was limited. Rotnet, being less robust than SimCLR, plateaued in performance much earlier. Increasing access to labels allowed the fully supervised models to have enough learning information to perform greater than the SSL models.

4.3 SimCLR+RotNet

To evaluate the performance of the SimCLR+RotNet model, we compare the Top-1 classification accuracies achieved by SimCLR+RotNet, SimCLR, and RotNet individually. Our results show that the SimCLR+RotNet model outperforms RotNet, but falls slightly short of SimCLR's performance. The motivation behind the hybrid model was to combine the strengths of both SimCLR and RotNet, aiming to extract more comprehensive and well-rounded feature representations. SimCLR excels at capturing instance-level semantic features through contrastive learning, while RotNet focuses on

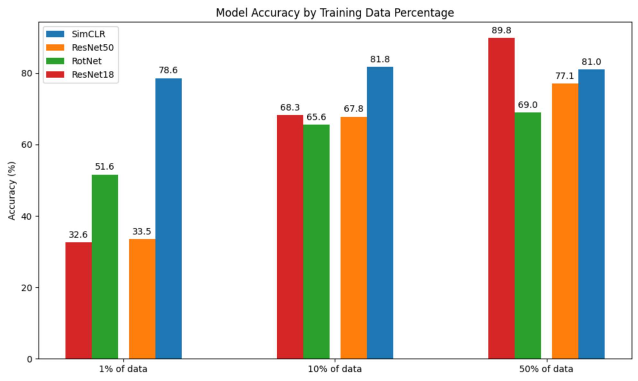


Figure 2: Comparison of SimCLR and RotNet with their fully supervised backbones

global structural understanding by predicting image rotations. However, our findings indicate that these two learning signals may conflict when combined, potentially interfering with each other during joint training. This interference could explain why the hybrid model does not fully meet its intended goal of surpassing both individual approaches. Figure 3 below presents the performance trends of all three models over 200 training epochs using a batch size of 256.

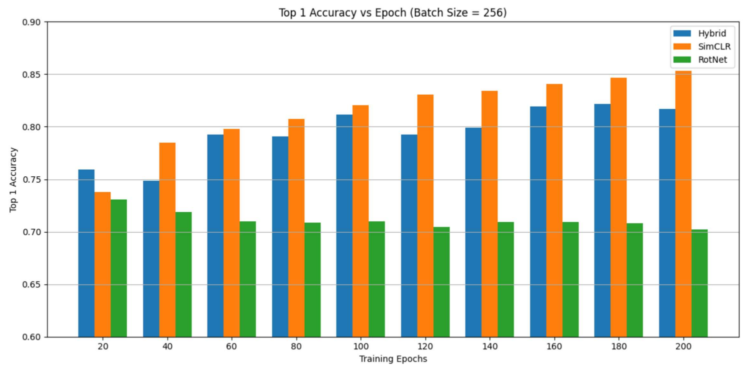


Figure 3: Performance Comparison of SimCLR, RotNet, and SimCLR+RotNet

5 Conclusion

In this work, we explored self-supervised learning as a robust approach to representation learning. We evaluated two very different pretext strategie: SimCLR is a contrastive learning framework focuses on maximizing agreement between augmented views of an image, while RotNet is a rotation classifier that discriminately predicts image rotations to gain global semantic understanding. Both models demonstrated significant improvement in performance over their fully-supervised counterparts when labels were scarce. This highlights the power of self-supervision and representation learning in real-world scenarios when put in low-label regimes.

To further explore the extent of representation learning, we proposed a hybrid model that combines SimCLR and RotNet with a dynamic weighting schedule that gradually shifts focus away from rotation prediction. Ideally, we aimed to combine the rich instance-level feature learning with global semantic understanding. Unfortunately, we observe that the contrasting goals of both models actually combat each other rather than interact positively.

Through extensive experiments with the CIFAR-10 dataset, we demonstrate how self-supervised models are viable options in real-world scenarios. They not only close the gap with fully supervised models, but also surpass them when label availability is low. These results reinforce the practicality of self-supervised models explained in modern academia and open opportunity for future "hybrid model" objectives.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[3] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. arXiv preprint arXiv:2006.10029.

[4] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018. URL `https://arxiv.org/abs/1803.07728`.

[5] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989. doi: 10.1007/BF01589116.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[7] PyTorch Contributors. torch.utils.data.subset, 2017. URL `https://pytorch.org/docs/stable/data.html#torch.utils.data.Subset`. Accessed: 2025-04-25.