Final Project : Self-Supervised Learning for an image classification task

Kim Hyuk Jin
2021142265
ekaterina9@yonsei.ac.kr

1. Introduction

In recent years, self-supervised learning (SSL) has emerged as a promising approach in the fields of machine learning and computer vision, enabling the training of models without the need for large amounts of labeled data. Traditional supervised learning methods rely heavily on annotated datasets, which are time-consuming and costly to produce. SSL methods address this limitation by learning from the intrinsic structure of the data itself, using pretext tasks to generate supervisory signals.

Among various SSL techniques, contrastive learning has shown significant success. However, it often requires large batch sizes and extensive computation, which can be impractical in many real-world applications. To overcome these challenges, new methodologies have been proposed that combine clustering with instance learning, offering a more efficient and scalable solution.

SwAV (Swapping Assignments between Views) represents a significant advancement in this direction. By simultaneously clustering the data while learning representations, SwAV leverages the benefits of both clustering and instance discrimination without the need for negative samples or large batch sizes. This approach not only enhances the quality of the learned representations but also reduces computational overhead.

The objective of this study is to explore the application of SwAV in a specific domain, evaluating its performance and comparing it with other SSL methods. We aim to demonstrate that SwAV can achieve superior results with less computational resources, making it a viable option for various practical applications.

The remainder of this paper is structured as follows: Section 2 provides a detailed overview of the SwAV algorithm and its theoretical underpinnings. Section 3 describes the experimental setup, including the datasets and evaluation metrics used. In Section 4, I present the results and discuss the implications of findings. Finally, Section 5 concludes the paper and outlines potential directions for future research.

2. Related Work

2.1. Self-Supervised Learning (SSL)

Self-supervised learning (SSL) has gained significant attention in recent years as an alternative to traditional supervised learning, particularly in the context of computer vision. SSL methods aim to learn useful feature representations from unlabeled data by leveraging intrinsic supervisory signals. Prominent approaches in SSL include as follows:

Contrastive Learning. This family of methods, such as SimCLR[3] and MoCo[4], focuses on learning representations by contrasting positive pairs (augmentations of the same image) against negative pairs (augmentations of different images). These methods have shown remarkable success but often require large batch sizes or memory banks to store negative samples, leading to increased computational costs.

Pretext Tasks. Techniques such as rotation prediction, jigsaw puzzles, and colorization have been used to create surrogate tasks that encourage the model to learn meaningful features. While effective, these methods often require carefully designed tasks that may not generalize well to all types of data.

2.2. Clustering-Based Self-Supervised Learning

Clustering-based SSL methods, such as DeepCluster and SeLa, attempt to cluster the feature representations and use the cluster assignments as pseudo-labels for training. These methods offer a balance between instance discrimination and representation learning without the need for negative samples :

DeepCluster. Caron et al. introduced DeepCluster [1], which iteratively clusters the features extracted by a convolutional neural network (CNN) and uses these clusters to update the network parameters. This approach has shown to produce competitive representations but suffer from instability during training due to the use of k-means clustering.

SeLa (Self-Labeling). Asano et al. proposed a method that frames clustering as a self-labeling problem using Sinkhorn-Knopp optimization to ensure balanced assign-

ments. [2] This method demonstrated improve stability and performance over traditional clustering methods.

2.3. SwAV : Advanced in Clustering-Based SSL

SwAV (Swapping Assignments between Views) builds upon the strengths of these clustering-based methods while addressing their limitations :

Online Clustering with Sinkhorn-Knopp. Unlike DeepCluster, which relies on offline k-means clustering, SwAV performs online clustering using the Sinkhorn-Knopp algorithm. This ensures balanced and stable assignments, enhancing training efficiency and representation quality.

Multi-View Consistency. By leveraging multiple views (augmentations) of the same image, SwAV encourages consistency between different augmentations. This achieved through a swapping mechanism that aligns the assignments of different views, leading to more invariant and robust feature representations.

Prototype-Based Learning. SwAV uses prototype vectors to represent clusters, allowing for a more flexible and scalable approach to clustering compared to traditional methods. This prototype-based approach simplifies the assignment process and integrates seamlessly with the online clustering mechanism.

3. Method

3.1. Overview of the SwAV Algorithm

SwAV (Swapping Assignments between Views) is a self-supervised learning algorithm that combines online clustering and contrastive learning to learn robust and generalizable feature representations from unlabeled data. SwAV leverages the strengths of clustering by using prototype vectors to represent clusters and ensures balanced assignments through the Sinkhorn-Knopp algorithm. This section provides a detailed overview of the SwAV algorithm, including the architecture, the Sinkhorn-Knopp clustering mechanism, and the loss function used for training.

3.2. Backbone Architecture

The backbone architecture used in SwAV is based on ResNet-18, a convolutional neural network (CNN) known for its effectiveness in image classification tasks. The ResNet-18 backbone is composed of multiple residual blocks, which help in learning hierarchical feature representations. Each residual block can be described as follows:

$$\mathbf{y} = \text{ReLU}(\mathbf{W_2} \cdot \text{ReLU}(\mathbf{W_1} \cdot \mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) + \mathbf{x_s} \quad (1)$$

where $\mathbf{W_1}$ and $\mathbf{W_2}$ are convolutional filters, $\mathbf{b_1}$ and $\mathbf{b_2}$ are biases, $\mathbf{x}$ is the input feature map, and $\mathbf{x_s}$ is the shortcut connection.

The overall architecture of ResNet-18 used in SwAV consists of:

- An initial convolutional layer followed by batch normalization and ReLU activation.
- Four stages of residual blocks with increasing filter sizes.
- An adaptive average pooling layer.
- A final fully connected layer that is replaced by a projection layer in SwAV.

3.3. Projection and Normalization

In SwAV, the final fully connected layer of the ResNet-18 backbone is replaced by a projection layer that maps the high-dimensional features to a lower-dimensional space suitable for clustering. The projection can be described as:

$$\mathbf{z} = \text{Normalize}(\mathbf{W_p} \cdot \mathbf{f} + \mathbf{b_p}) \quad (2)$$

where $\mathbf{f}$ is the feature vector from the backbone, $\mathbf{W_p}$ and $\mathbf{b_p}$ are the weights and biases of the projection layer, and $\mathbf{z}$ is the normalized feature vector.

3.4. Online Clustering with Sinkhorn-Knopp

SwAV uses the Sinkhorn-Knopp algorithm to perform online clustering of the projected features. The goal is to obtain balanced assignments of features to prototype vectors. The Sinkhorn-Knopp algorithm is an iterative process that normalizes the rows and columns of the similarity matrix to enforce doubly stochastic constraints. The algorithm can be described as:

$$\mathbf{P} = \arg\min_{\mathbf{P}} \langle \mathbf{P}, \mathbf{Q} \rangle - \epsilon H(\mathbf{P}) \quad (3)$$

subject to $\mathbf{P1} = \mathbf{r}$ and $\mathbf{P}^\top \mathbf{1} = \mathbf{c}$, where ϵ is a regularization parameter, $H(\mathbf{P})$ is the entropy of $\mathbf{P}$, $\mathbf{r}$ and $\mathbf{c}$ are marginal constraints, and $\mathbf{1}$ is a vector of ones.

The iterative updates for the Sinkhorn-Knopp algorithm are:

$$\mathbf{u} = \mathbf{Q1}, \quad \mathbf{Q} \leftarrow \frac{\mathbf{Q}}{\mathbf{u1}^\top}, \quad \mathbf{v} = \mathbf{Q}^\top \mathbf{1}, \quad \mathbf{Q} \leftarrow \frac{\mathbf{Q}}{\mathbf{1v}^\top} \quad (4)$$

These updates are repeated for a fixed number of iterations to obtain the final assignment matrix $\mathbf{P}$.

3.5. SwAV Training Process

The training process of SwAV involves the following steps:

1. **Feature Extraction**: Two augmented views of the input images ($\mathbf{x_1}$ and $\mathbf{x_2}$) are passed through the ResNet-18 backbone to obtain feature representations ($\mathbf{z_1}$ and $\mathbf{z_2}$):

$$\mathbf{z_1} = \text{Normalize}(\text{Projection}(\text{Backbone}(\mathbf{x_1}))) \quad (5)$$

$$\mathbf{z_2} = \text{Normalize}(\text{Projection}(\text{Backbone}(\mathbf{x_2}))) \quad (6)$$

2. **Similarity Scores**: The normalized features are used to compute similarity scores with prototype vectors $\mathbf{C}$:

$$\mathbf{S} = [\mathbf{z_1}; \mathbf{z_2}]\mathbf{C}^\top \tag{7}$$

3. **Balanced Assignment**: The scores $\mathbf{S}$ are split into two parts $\mathbf{S_t}$ and $\mathbf{S_s}$, which are then processed by the Sinkhorn-Knopp algorithm to obtain the balanced assignments $\mathbf{Q_t}$ and $\mathbf{Q_s}$:

$$\mathbf{Q_t} = \text{Sinkhorn}(\mathbf{S_t}) \tag{8}$$

$$\mathbf{Q_s} = \text{Sinkhorn}(\mathbf{S_s}) \tag{9}$$

4. **Softmax Probabilities**: The softmax probabilities are computed for both parts:

$$\mathbf{P_t} = \text{Softmax}(\mathbf{S_t}/T) \tag{10}$$

$$\mathbf{P_s} = \text{Softmax}(\mathbf{S_s}/T) \tag{11}$$

5. **Loss Computation**: The loss function aligns the assignments of different views of the same image, encouraging consistent feature learning. The loss is computed as follows:

$$\mathcal{L} = -\frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbf{Q_t}_{ij}\log\mathbf{P_s}_{ij}\right) - \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbf{Q_s}_{ij}\log\mathbf{P_t}_{ij}\right) \tag{12}$$

where N is the number of samples, K is the number of prototypes, $\mathbf{Q_t}$ and $\mathbf{Q_s}$ are the balanced assignments, and $\mathbf{P_t}$ and $\mathbf{P_s}$ are the softmax probabilities.

4. Experiments and Results

4.1. Experimental Setup

For our experiments, we utilized the ResNet-18 backbone within the SwAV framework, performing self-supervised learning on the CIFAR-100 dataset. The following hyperparameters were used:

- **Number of epochs**: 100
- **Batch size**: 256
- **Feature dimension**: 128
- **Number of prototypes**: 3000
- **Learning rate**: 0.001

The choice of hyperparameters plays a critical role in the performance and efficiency of the SwAV model. Here, we discuss the rationale behind selecting specific values for the batch size, feature dimension, and number of prototypes.

Batch Size: 256 A batch size of 256 was chosen to balance between computational efficiency and the stability of the optimization process. Larger batch sizes generally provide more stable gradients, which can lead to more robust convergence. However, very large batch sizes may not fit into GPU memory and can slow down the training process due to increased computation per batch. A batch size of 256 is a commonly used value that provides a good trade-off between these factors.

Feature Dimension: 128 The feature dimension, which refers to the dimensionality of the embedding space where the input images are projected, was set to 128. This dimensionality is sufficient to capture the essential characteristics of the input images while keeping the computational complexity manageable. A higher dimensionality could lead to overfitting and increased computation, while a lower dimensionality might not capture enough information from the images. A dimension of 128 is a commonly used value that strikes a balance between these concerns.

Number of Prototypes: 3000 The number of prototypes (or clusters) was set to 3000 to ensure a diverse and comprehensive representation of the data in the feature space. Having a large number of prototypes allows the model to capture subtle variations in the data and provides more fine-grained clustering, which can improve the quality of the learned representations. However, too many prototypes can lead to increased computational overhead. The value of 3000 was chosen as it provides a good balance, capturing enough detail without excessive computation.

I conducted experiments with two different data augmentation strategies to evaluate their impact on the model's performance.

4.2. Data Augmentation Strategies

First Augmentation Strategy:

The first data augmentation strategy included:

- **Random Horizontal Flip**: Flips the image horizontally with a probability of 0.5.
- **Color Jittering**: Randomly changes the brightness, contrast, saturation, and hue of the image.
- **Normalization**: Normalizes the image with mean and standard deviation specific to the CIFAR-100 dataset.

Second Augmentation Strategy:

The second data augmentation strategy included:

- **Random Horizontal Flip**: Flips the image horizontally with a probability of 0.5.
- **Normalization**: Normalizes the image with mean and standard deviation specific to the CIFAR-100 dataset.

4.3. Evaluation Metrics

The models were evaluated based on Top-1 and Top-5 accuracy metrics on the CIFAR-100 dataset.

4.4. Results

Results with First Augmentation Strategy:

Fraction	Top-1 Accuracy	Top-5 Accuracy
1%	2.92%	11.79%
10%	15.42%	36.82%
30%	27.85%	51.95%
40%	31.15%	55.84%

Table 1. Performance with First Augmentation Strategy

Results with Second Augmentation Strategy:

Fraction	Top-1 Accuracy	Top-5 Accuracy
1%	4.72%	16.22%
10%	18.69%	41.44%
30%	29.44%	55.21%
40%	33.79%	58.73%

Table 2. Performance with Second Augmentation Strategy

4.5. Discussion

From the results, it is evident that the data augmentation strategy plays a significant role in the performance of the SwAV model. The first augmentation strategy, which included color jittering, resulted in slightly lower Top-1 and Top-5 accuracies compared to the second strategy, which only included horizontal flipping and normalization. Fine-tuning with varying fractions of labeled data consistently improved the model's performance, demonstrating the effectiveness of the SwAV pre-training in learning useful feature representations.

These findings highlight the importance of choosing appropriate data augmentation techniques in self-supervised learning tasks and underscore the robustness of the SwAV approach in leveraging unlabeled data to achieve competitive performance on downstream tasks.

5. Conclusion

In this study, I explored the application of the SwAV (Swapping Assignments between Views) algorithm for self-supervised learning on the CIFAR-100 dataset using a ResNet-18 backbone. The SwAV algorithm combines online clustering with contrastive learning to effectively leverage unlabeled data for learning robust and generalizable feature representations. My experimental setup involved two distinct data augmentation strategies, and I evaluated the model's performance based on Top-1 and Top-5 accuracy metrics.

The key findings from my experiments can be summarized as follows:

- **Impact of Data Augmentation**: The choice of data augmentation significantly affected the performance of the SwAV model. The first augmentation strategy, which included color jittering in addition to random horizontal flips and normalization, resulted in slightly lower accuracies compared to the second strategy that only used horizontal flips and normalization. This highlights the importance of selecting appropriate augmentation techniques to enhance the learning process in self-supervised frameworks.
- **Fine-Tuning with Labeled Data**: Fine-tuning the SwAV model with varying fractions of labeled data consistently improved its performance. This demonstrates the effectiveness of SwAV pre-training in providing a strong initialization for downstream tasks, allowing for better utilization of labeled data and achieving competitive performance even with limited labels.
- **Hyperparameter Choices**: The selected hyperparameters, including a batch size of 256, feature dimension of 128, and 3000 prototypes, were chosen to balance computational efficiency and model performance. These choices were justified by the stability they provided during training and their ability to capture essential characteristics of the data without overfitting.

Overall, my study confirms that the SwAV algorithm is a powerful tool for self-supervised learning, capable of learning high-quality representations from unlabeled data and performing well on downstream tasks with minimal labeled data. Future work could explore further optimization of data augmentation strategies, hyperparameter tuning, and the application of SwAV to other datasets and architectures to extend its applicability and robustness.

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. 1

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[4] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 1