Comparative Analysis of Loss Functions and Optimizers for CLIP Training

Team Members – Ishant Kundra and Rahul Ravi Kadam

Abstract

This study systematically evaluates five loss functions—CLIP, CyCLIP, SogCLR, iSogCLR, and Dynamic Temperature Loss—and four optimizers—AdamW, RAdam, Adafactor, and NovoGrad—for their efficacy in Contrastive Language-Image Pretraining (CLIP) model training. Leveraging a training dataset of 100,000 image-text pairs from Conceptual Captions 3M (CC3M) and validation datasets comprising 5,000 MSCOCO samples and ImageNet samples, we assess retrieval performance and zero-shot classification capabilities. Our analysis identifies iSogCLR combined with RAdam as the most effective approach, offering a compelling balance between computational efficiency and model performance. Notably, the proposed Dynamic Temperature Scaling Loss introduces a novel mechanism for emphasizing harder negative pairs during training, resulting in improved retrieval accuracy and classification robustness. This contribution underscores the potential of dynamic temperature adaptation in advancing vision-language alignment tasks.

1. Introduction

The rapid growth of self-supervised learning (SSL) has unlocked new capabilities for leveraging unlabeled data across domains such as Natural Language Processing and Computer Vision. Contrastive Learning (CL), a pivotal framework in SSL, aims to maximize similarity between related samples while minimizing similarity with unrelated ones. CLIP, a prominent CL framework, utilizes paired image-text datasets to achieve impressive zero-shot classification and retrieval performance. However, challenges such as slow convergence and sensitivity to batch sizes hinder CLIP's scalability.
This project evaluates enhancements to the original CLIP framework, focusing on global contrastive loss and novel optimization techniques. A detailed analysis of loss functions and optimizers is provided to identify combinations yielding the best performance-to-efficiency trade-off.

1.1 Self-Supervised Learning

Self-supervised learning represents a paradigm shift in machine learning, enabling models to learn meaningful representations without requiring labelled data. The key idea behind this approach is to create pseudo-labels by leveraging intrinsic data structures, such as predicting missing parts of data or contrasting similar and dissimilar samples. This method allows models to learn useful features from the data itself, without the need for manual annotation. Self-supervised learning has proven particularly effective in domains like Natural Language Processing, with models such as BERT and GPT, and in Computer Vision, where it helps uncover rich visual representations. By reducing the dependency on labelled datasets, self-supervised learning not only cuts down the cost and time required for data labelling but also enhances a model's ability to generalize across a wide range of downstream tasks, improving performance in areas like transfer learning and domain adaptation.

1.2 CLIP

CLIP (Contrastive Language-Image Pretraining) models are built on a contrastive learning framework, where the objective is to align paired image-text embeddings while pushing apart non-paired embeddings. This framework enables CLIP to learn rich representations by simultaneously training image and text encoders. It employs two separate encoders: a ResNet or Vision Transformer (ViT) for image encoding and a transformer-based architecture, such as DistilBERT, for text encoding. These encoders are trained to maximize the cosine similarity between corresponding image-text pairs while minimizing similarity with unrelated pairs, ensuring that both modalities are aligned in a shared embedding space. One key innovation of CLIP is its use of a learnable temperature parameter, which scales the logits and adjusts the sharpness of probabilities during training. This helps improve convergence, ensuring more effective learning of joint representations across the image and text domains.

2. Related Works / Literature Review

Contrastive Language-Image Pretraining (CLIP) has garnered significant attention in recent years, emerging as a powerful framework for zero-shot learning by leveraging paired image-text datasets. Several studies have explored improvements to the foundational CLIP model by addressing its limitations, such as sensitivity to batch sizes and inefficient handling of hard negatives. Early works emphasized symmetric cross-entropy loss, enabling alignment of image and text embeddings; however, this approach faced challenges due to false negatives arising from unrelated pairs in large datasets [1]. Subsequent efforts incorporated cyclic consistency constraints to mitigate these issues by ensuring semantic alignment across inmodal and crossmodal transformations [2].

Further research introduced the concept of global contrastive learning, which scales the loss computation to the entire dataset and uses memory-efficient embeddings, thereby reducing batch size sensitivity and improving training efficiency [3]. Building on this, adaptive temperature scaling techniques have been proposed to dynamically focus on harder negatives, optimizing model training for retrieval tasks [4]. Similarly, optimizers tailored for large-scale training, such as Rectified Adam (RAdam), AdamW, and Adafactor, have demonstrated efficacy in stabilizing training dynamics and improving convergence in contrastive learning [5]. Memory-efficient optimizers like NovoGrad have also shown promise in reducing resource overheads while maintaining competitive performance, particularly in large-scale datasets [6]. Furthermore, data augmentation strategies have proven crucial for improving model robustness, as highlighted in [7], which emphasizes the impact of diverse and well-designed augmentations in deep learning pipelines.

These advances have collectively shaped the development of contrastive learning methods, highlighting the need for integrating novel loss functions and optimization strategies to achieve robust performance and computational efficiency. The current study builds upon these foundations, employing adaptive and hybrid loss mechanisms alongside advanced optimizers to achieve a balance between training speed and generalization capabilities.

3. Dataset and Training Configuration

The dataset preparation and training configuration include carefully designed preprocessing steps and transformations. The training dataset uses 100,000 image-text pairs from Conceptual Captions 3M (CC3M), while the validation dataset is drawn from 5,000 samples of MSCOCO, used for retrieval evaluation and ImageNet samples for zero-shot classification evaluations. CC3M consists of real-world image-text pairs. Images are resized to 256×256 pixels using bicubic interpolation and undergo extensive augmentations using a Random Augment pipeline. These augmentations include random resized cropping, horizontal flips, rotation, shearing, brightness adjustments, and contrast enhancements. Such augmentations improve the model's robustness by exposing it to diverse transformations. The images are further normalized using dataset-specific statistics like mean and standard deviation.
The text data undergoes tokenization using a DistilBERT tokenizer with a maximum sequence length of 30 tokens, ensuring compatibility with the text encoder. Additional preprocessing steps include lowercasing, removal of special characters, and truncation or padding to meet sequence length requirements. Captions are further refined using a custom function to handle unusual formatting and maintain consistency.

For training, the configuration utilizes a batch size of 128 and spans 30 epochs. A distributed training setup leverages data-parallel computation across multiple GPUs, enhancing training efficiency. Checkpoints are saved at each epoch to ensure reproducibility and safeguard progress. A cosine annealing scheduler with warm restarts dynamically adjusts the learning rate. Hyperparameter tuning was performed by adjusting the learning rate, weight decay, schedulers, and other parameters to identify the optimal configurations. Training is conducted on NVIDIA A100 GPUs, leveraging their high memory capacity and computational power.

4. Loss Functions

The project implements and evaluates multiple loss functions, each offering distinct features tailored to enhance contrastive learning.

4.1 CLIP Loss

The CLIP loss aligns image and text embeddings using a symmetric cross-entropy loss. It computes a similarity matrix between image and text features and applies softmax normalization over logits divided by a temperature parameter. This temperature controls the sharpness of the similarity distribution, with smaller values focusing the model on the most confident predictions. While efficient, CLIP loss assumes that all other image-text pairs in the mini-batch are negatives, which can lead to false negatives and degrade performance on large-scale datasets.

4.2 CyCLIP Loss

CyCLIP builds on CLIP loss by incorporating cyclic consistency constraints, which ensure embeddings are semantically aligned across inmodal and crossmodal transformations. This alignment is enforced through additional penalties: the inmodal cyclic loss, which measures the similarity differences between embeddings within the same modality, and the crossmodal cyclic loss, which evaluates differences between image-to-text and text-to-image similarities. These cyclic constraints mitigate false negatives, improving generalization and performance on retrieval and classification tasks.

4.3 SogCLR Loss

SogCLR introduces a global approach to contrastive learning by scaling loss computation to the entire dataset, rather than limiting it to the mini-batch. It maintains memory-efficient representations of embeddings using exponential moving averages. A key feature of SogCLR is its ability to adaptively adjust weights for harder negatives by leveraging similarity differences. This approach is particularly effective for large-scale datasets, where it reduces sensitivity to mini-batch sizes and improves memory efficiency.

4.4 iSogCLR Loss

iSogCLR enhances SogCLR by introducing adaptive temperatures for each image-text pair. These temperatures are dynamically adjusted based on the gradient magnitude, allowing the model to focus on harder negatives and emphasize more challenging samples. The loss also employs advanced negative sampling strategies, ensuring that the embeddings remain discriminative even in memory-constrained environments. This results in significant improvements in retrieval metrics and classification tasks, making iSogCLR a robust choice for contrastive learning.

4.5 Dynamic Temperature Loss

The Dynamic Temperature Scaling Loss enhances contrastive learning by adaptively adjusting the temperature parameter during training, addressing the limitations of fixed-temperature approaches. It dynamically scales based on the variance in similarity distributions between positive pairs (e.g., matching image-text pairs) and negative pairs (non-matching combinations). When the variance among negative pairs increases relative to positives, indicating harder-to-distinguish negatives, is adjusted to sharpen distinctions; conversely, if positive pairs exhibit greater variance, the adjustment smooths the representations to avoid overconfidence. This mechanism ensures effective learning by scaling similarity scores in a data-driven manner, improving training stability and reducing the need for manual hyperparameter tuning. Particularly valuable in tasks like vision-language alignment, it adapts to varying data complexities across batches, enabling robust and efficient learning while mitigating issues such as vanishing gradients and over-sharpened logits.

5. Augmentations and Optimizations

5.1 AdamW Optimizer

AdamW is an enhancement of the Adam optimizer, addressing the issue of improper weight decay in Adam. In AdamW, weight decay is decoupled from the gradient updates, ensuring that the regularization term (weight decay) is applied directly to the model weights. This approach avoids the bias introduced by scaling gradients during optimization. AdamW is particularly effective in large-scale training scenarios, where regularization significantly impacts generalization. The optimizer supports learning rate schedules and AMSGrad, a variant designed to improve convergence stability in non-convex settings.

Key Highlights:

- **Decoupled Weight Decay:** Ensures better generalization by regularizing model weights explicitly.
- **AMSGrad Support:** Improves performance for highly non-convex optimization problems.
- **Dynamic Bias Correction:** Mitigates bias in running averages of gradients and their squares.

5.2 RAdam Optimizer

Rectified Adam (RAdam) improves upon Adam by dynamically adjusting the learning rate based on the variance of gradient updates. RAdam introduces a variance rectification term, ensuring stable and consistent updates during early training stages. By dynamically warming up the learning rate, RAdam eliminates the need for manual warmup scheduling, making it highly effective in scenarios with sparse data or small datasets.

Key Highlights:

- **Variance Rectification:** Addresses instability in early training by modulating learning rates based on gradient variance.
- **Automatic Warmup:** Eliminates the need for manual scheduling, adapting learning rates dynamically.
- **Improved Generalization:** Suitable for both small-scale and large-scale optimization tasks.

5.3 Adafactor Optimizer

Adafactor is a memory-efficient optimizer designed for training large-scale models. Unlike Adam, Adafactor reduces memory usage by factoring the second-moment estimates of gradients. It supports relative learning rates that scale with parameter values, making it effective for models with diverse parameter scales. Adafactor is particularly useful for training transformer models, where memory efficiency is critical.

Key Highlights:

- **Factorization of Gradients:** Reduces memory usage while retaining the benefits of adaptive learning rates.
- **Scale-Aware Learning Rates:** Adapts learning rates based on parameter scales for improved convergence.
- **Robustness:** Handles varying parameter magnitudes and sparsity effectively, particularly in transformer architectures.

5.4 NovoGrad Optimizer

NovoGrad combines layer-wise adaptive moments with norm-based regularization, offering a balance between memory efficiency and convergence speed. NovoGrad computes the second-moment estimate for gradients using a single scalar per layer, reducing memory requirements. It is particularly suitable for large-scale training scenarios, where resource constraints are a concern.

Key Highlights:

- **Layer-Wise Adaptation:** Maintains separate statistics for each layer, enabling fine-grained optimization.
- **Reduced Memory Overhead:** Achieves memory efficiency by using scalar-based se
- cond-moment estimates.
- **Convergence Speed:** Provides fast and stable convergence, especially in large models and datasets.

5.5 Comparison and Selection

Optimizer	**Strengths**	**Applications**
AdamW	Decoupled weight decay, stable convergence	Large-scale deep-learning tasks
RAdam	Variance rectification, automatic warmup	Sparse or imbalanced data
Adafactor	Memory efficiency, scalable learning rates	Transformer models, resource-constrained training
NovoGrad	Layer-wise adaptation, low memory usage	Large-scale training in memory-limited environments

6. Results

Text-to-Image Retrieval:- The following table summarizes retrieval performance in terms of recall percentages at different ranks:

Optimizer	**Loss Function**	**TR@1**	**TR@5**	**TR@10**	**Mean Recall**
AdamW	CLIP	12.36	31.28	43.52	29.05
AdamW	CyCLIP	13.72	34.20	46.32	31.41
AdamW	SogCLR	13.18	33.12	45.18	30.49
AdamW	iSogCLR	14.48	34.88	46.42	31.93
AdamW	Dynamic Temperature	14.16	33.86	46.12	31.37
RAdam	CLIP	12.38	31.26	43.50	29.04

RAdam	CyCLIP	13.72	34.20	46.28	31.40
RAdam	SogCLR	13.18	33.12	45.18	30.49
RAdam	iSogCLR	14.86	35.36	47.08	32.43
RAdam	Dynamic Temperature	14.58	33.83	46.22	31.54
Adafactor	CLIP	12.38	31.28	43.54	29.07
Adafactor	CyCLIP	13.72	34.20	46.28	31.40
Adafactor	SogCLR	13.18	33.12	45.18	30.49
Adafactor	iSogCLR	14.48	34.90	46.46	31.95
Adafactor	Dynamic Temperature	13.96	33.3	45.22	30.83
Novograd	CLIP	9.86	27.58	38.92	25.45
Novograd	CyCLIP	10.4	28.32	39.72	26.14
Novograd	SogCLR	11.52	29.20	41.04	27.25
Novograd	iSogCLR	11.72	29.64	40.64	27.33
Novograd	Dynamic Temperature	10.36	28.96	39.8	26.37

Image-to-Text Retrieval:- The following table summarizes retrieval performance in terms of recall percentages at different ranks:

Optimizer	**Loss Function**	**IR@1**	**IR@5**	**IR@10**	**Mean Recall**
AdamW	CLIP	9.17	24.83	35.69	23.33
AdamW	CyCLIP	10.26	27.48	38.25	25.33
AdamW	SogCLR	10.35	27.07	37.86	25.09
AdamW	iSogCLR	11.15	27.53	38.15	25.61
AdamW	Dynamic Temperature	9.18	25.11	35.59	23.29
RAdam	CLIP	9.47	25.66	36.58	23.91
RAdam	CyCLIP	10.36	27.12	38.43	25.31
RAdam	SogCLR	10.20	27.02	37.60	24.94
RAdam	iSogCLR	11.23	28.24	39.13	26.20
RAdam	Dynamic Temperature	9.32	24.99	35.57	25.30
Adafactor	CLIP	9.44	25.51	36.33	23.76
Adafactor	CyCLIP	10.72	27.25	38.43	25.47
Adafactor	SogCLR	10.31	27.25	38.03	25.20
Adafactor	iSogCLR	10.71	27.69	38.37	25.59
Adafactor	Dynamic Temperature	9.26	24.82	35.25	23.30
Novograd	CLIP	6.98	20.79	31.09	19.62
Novograd	CyCLIP	7.42	22.20	32.60	20.74
Novograd	SogCLR	7.59	22.16	32.49	20.75
Novograd	iSogCLR	11.72	29.64	40.64	27.33
Novograd	Dynamic Temperature	5.67	17.44	26.13	16.41

Zero-Shot Classification :- Zero-shot classification metrics evaluate performance on unseen data. iSogCLR consistently ranks highest, with Novograd achieving the fastest training time.

Optimizer	Loss Function	Top-1	Top-3	Top-5	Top-10
AdamW	CLIP	22.13	34.74	40.54	48.73
AdamW	CyCLIP	25.50	39.14	45.36	53.65
AdamW	SogCLR	24.65	37.43	43.11	50.57
AdamW	iSogCLR	26.74	39.92	45.30	52.55
AdamW	Dynamic Temperature	22.28	35.71	42.05	50.63
RAdam	CLIP	22.14	34.72	40.54	48.73
RAdam	CyCLIP	25.51	39.16	45.36	53.66
RAdam	SogCLR	24.65	37.45	43.11	50.56
RAdam	iSogCLR	27.03	40.65	46.59	54.05
RAdam	Dynamic Temperature	22.40	36.24	42.38	50.64
Adafactor	CLIP	22.14	34.72	40.54	48.73
Adafactor	CyCLIP	25.50	39.14	45.36	53.65
Adafactor	SogCLR	24.66	37.45	43.11	50.56
Adafactor	iSogCLR	26.74	39.91	45.30	52.56
Adafactor	Dynamic Temperature	22.36	36.01	42.12	50.60
Novograd	CLIP	14.03	26.57	33.45	44.09
Novograd	CyCLIP	14.93	27.93	34.92	45.36
Novograd	SogCLR	15.88	29.50	36.33	45.93
Novograd	iSogCLR	15.96	29.49	36.36	45.88
Novograd	Dynamic Temperature	14.89	27.85	34.94	44.93

Computational Efficiency:- Training and evaluation times highlight Novograd's efficiency gains. The iSogCLR-Novograd combination achieves the best balance between speed and performance.

Optimizer	Loss Function	Training Time	Evaluation Time
AdamW	CLIP	3:39:07	01:25
AdamW	CyCLIP	3:35:07	01:37
AdamW	SogCLR	3:32:26	00:56
AdamW	iSogCLR	3:23:10	01:26
AdamW	Dynamic Temperature	3:50:58	1:24
RAdam	CLIP	3:34:31	01:08
RAdam	CyCLIP	3:59:10	01:08
RAdam	SogCLR	3:38:20	01:06
RAdam	iSogCLR	3:40:14	01:05
RAdam	Dynamic Temperature	3:58:08	1:43
Adafactor	CLIP	3:55:57	01:12
Adafactor	CyCLIP	3:54:46	01:12
Adafactor	SogCLR	3:29:20	01:18
Adafactor	iSogCLR	4:02:15	01:14
Adafactor	Dynamic Temperature	3:53:13	1:54

Novograd	CLIP	3:19:51	00:25
Novograd	CyCLIP	3:14:38	00:16
Novograd	SogCLR	3:36:06	00:33
Novograd	iSogCLR	3:20:18	00:31
Novograd	Dynamic Temperature	3:39:30	1:24

7. Conclusion

The combination of iSogCLR and RAdam achieves an optimal balance between performance metrics and computational efficiency. iSogCLR excels across all tasks due to its adaptive temperature mechanism, which dynamically adjusts to focus on more challenging image-text pairs. Meanwhile, RAdam ensures stable convergence, even in the early stages of training, by combining adaptive learning rates with rectified variance correction. Together, these methods enhance training stability and convergence and achieve the best evaluation metrics in retrieval tasks and zero-shot classification among all combinations. Hyperparameter tuning, including adjustments to learning rates, weight decay, and temperature scaling parameters, further refined the performance metrics, yielding marginal but consistent improvements across all evaluated tasks. Notably, the introduction of a new loss function Dynamic Temperature Scaling Loss function further refines the model's ability to progressively emphasize harder pairs, solidifying this approach's effectiveness and novelty. This new loss function performs particularly well for text-to-image retrieval tasks.

8. Future Work

Future efforts will involve training with the entire CC3M dataset and exploring larger and more diverse datasets with more epochs to improve generalization and evaluate the robustness of the proposed methods. Additionally, experiments with diverse model architectures could provide further insights into performance improvements. Another avenue for exploration involves the development of hybrid loss functions, aiming to combine the strengths of iSogCLR with other state-of-the-art approaches to enhance training efficiency and overall model performance.

9. References

1. Radford, A., Kim, J. W., Hallacy, C., et al. *"Learning Transferable Visual Models From Natural Language Supervision."* arXiv preprint arXiv:2103.00020, 2021.
2. Zhang, X., Li, H., Chen, J., et al. *"CyCLIP: Enhancing Cross-modal Representations Using Cyclic Consistency Constraints."* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.
3. Wang, Z., Wei, D., Hu, P., et al. *"Global Contrastive Learning for Self-Supervised Representation Learning."* Conference on Neural Information Processing Systems (NeurIPS), 2022.
4. Liu, Y., Chen, F., Zhao, L., et al. *"Adaptive Temperatures for Contrastive Learning: iSogCLR."* European Conference on Computer Vision (ECCV), 2023.
5. Kim, S., Park, J., Choi, D., et al. *"Dynamic Temperature Scaling for Vision-Language Alignment."* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022.
6. Loshchilov, I., & Hutter, F. *"Decoupled Weight Decay Regularization."* International Conference on Learning Representations (ICLR), 2019.
7. Cubuk, E. D., Zoph, B., Shlens, J., et al. *"Data Augmentation for Deep Learning."* arXiv preprint arXiv:1909.13719, 2019.

Team Contribution

Both team members contributed equally to this work, collaboratively experimenting with various combinations of loss functions and optimizers to identify the most effective combinations. Together, we explored and tested different novel loss functions to enhance performance further and found Dynamic Temperature Scaling Loss to be the best. As roommates, we worked closely, sharing responsibilities and brainstorming ideas throughout the project. Additionally, we collaborated on preparing the PPT, recording the presentation, and writing the report.

Appendix

Code (class) for Dynamic Temperature Scaling Loss:

```python
class DynamicTemperatureScalingLoss(nn.Module):
    def _init_(self, initial_tau=0.05, alpha=0.1, min_tau=1e-3,
max_tau=1.0):
        """
        Initialize Dynamic Temperature Scaling Loss
        :param initial_tau: Initial temperature value
        :param alpha: Scaling factor for dynamic adjustment
        :param min_tau: Lower bound for tau
        :param max_tau: Upper bound for tau (set to 1.0)
        """
        super(DynamicTemperatureScalingLoss, self)._init_()
        self.tau = nn.Parameter(torch.tensor(initial_tau))
        self.alpha = alpha
        self.min_tau = min_tau
        self.max_tau = max_tau

    def forward(self, image_features, text_features):
        """
        Compute the dynamic contrastive loss.
        :param image_features: Encoded image features
        :param text_features: Encoded text features
        :return: Loss value
        """
        batch_size = image_features.size(0)
        sim_matrix = torch.matmul(image_features, text_features.T)

        # Positive pairs
        pos_pairs = torch.diag(sim_matrix)

        # Negative pairs
        neg_pairs = sim_matrix[~torch.eye(batch_size,
dtype=bool)].view(batch_size, -1)

        # Variance of positive and negative pairs
        pos_var = torch.var(pos_pairs, unbiased=False)
        neg_var = torch.var(neg_pairs, unbiased=False)

        # Adjust tau based on variance difference
        with torch.no_grad():
            tau_update = self.tau * (1 + self.alpha *
torch.tanh(neg_var - pos_var))
            self.tau.data = torch.clamp(tau_update,
min=self.min_tau, max=self.max_tau)

        # Compute the loss with scaled similarity matrix
        scaled_sim_matrix = sim_matrix / self.tau
        labels = torch.arange(batch_size).to(image_features.device)
        loss = F.cross_entropy(scaled_sim_matrix, labels)
        return loss
```