Self-Supervised Learning Approach for Jet Image Analysis Using SimCLR

Introduction

This document details my implementation of a self-supervised learning framework for jet image analysis. The project utilizes contrastive learning, precisely the SimCLR approach, to extract meaningful representations from unlabeled jet data. These representations are then used for classification and regression tasks. This documentation covers the complete methodology from pretraining to fine-tuning and evaluation.

Self-Supervised Learning Implementation

Data Preparation and Augmentation Strategy

Working with jet images presented unique challenges. I have also done a rough visualization to know about the data but have not attached its code to GitHub.The dataset consisted of unlabeled jet images stored in HDF5 format. I implemented a ContrastiveDataset class that reads this data and normalizes values to the [0,1] range to ensure consistent input scaling.

The augmentation pipeline, implemented through a custom SimCLR Augmentation class, forms a critical component of the contrastive learning approach. Each jet image is transformed into two different views using:

- Random horizontal flips (leveraging symmetry properties in jet physics)
- Random crops with resizing (maintaining at least 80% of the original image)
- Random brightness adjustments to increase variation

These augmentations create pairs of views that maintain content similarity while differing in appearance, which is essential for effective contrastive learning.

Custom Architecture Development

I developed a custom ResNet15 architecture specifically tailored for jet images. The implementation includes:

- Custom ResNetBlock modules with skip connections
- GroupNorm instead of BatchNorm (selected for better performance with smaller batch sizes)
- Strategic dropout layers for regularization
- A sequence of blocks that gradually increase the feature dimension to 512

This architecture provides an effective feature extractor that captures complex patterns in jet images while maintaining parameter efficiency.

Contrastive Learning Framework Selection

After researching several other self-supervised approaches mentioned in the task's description(MoCo, OBoW, Barlow Twins, VICReg), I selected SimCLR for this project due to its:

- Simplicity of implementation while maintaining strong performance

- Effectiveness with limited computational resources
- Strong performance on image data with geometric features similar to jet images
- Better convergence properties for the specific dataset size

The implementation includes:

- A ContrastiveModel class combining the ResNet encoder with a projection head
- A custom InfoNCELoss implementation that maximizes agreement between positive pairs while separating negative pairs
- A learnable temperature parameter to improve training stability

Training Strategy and Hyperparameters

The training implementation includes:

- Batch size: 256 (optimized for available GPU memory)
- Learning rate: 0.001 with cosine schedule and 10-epoch warmup
- AdamW optimizer with weight decay of 1e-4
- 100 total training epochs
- Temperature parameter initialized at 0.07
- Projection head dimension: 128

These hyperparameters were selected after experimental optimization to balance computational efficiency with representation quality.

Fine-Tuning Methodology for Downstream Tasks

Experimental Setup

To ensure reproducibility, I established consistent experimental conditions by:

- Setting fixed random seeds for Python, NumPy, and PyTorch
- Configuring appropriate warning filters
- Standardizing evaluation metrics across experiments

Data Handling for Supervised Learning

For the supervised phase, I developed specialized dataset classes:

FineTuneDataset for classification tasks:

- Reads data from HDF5 files
- Applies necessary transformations including channel permutation and normalization
- Supports optional data augmentations via the FineTuneAugmentation class

MassRegressionDataset for regression tasks:

- Processes image data efficiently
- Generates and normalizes mass values based on label correlations
- Provides properly scaled targets for the regression model

The data was split with 80% for training and 20% for evaluation to ensure reliable model assessment and was also mentioned in the description.

Architecture Adaptation

The backbone remained the custom ResNet-15 architecture, with specialized components added for specific tasks:

Classification Head:

- A feed-forward network mapping encoder outputs to class logits
- Dropout rate: 0.3 to prevent overfitting

Regression Head:

- A regressor with residual connections
- Deep supervision mechanism for more robust predictions
- Multiple output layers to enable gradient flow at different network depths

I implemented functionality for loading pre-trained weights from the self-supervised learning stage, including adaptations for weight dimension mismatches when necessary.

Classification Training Approach

For the classification task, I implemented:

- Cross-entropy loss function
- Adam optimizer with learning rate 0.0005
- ReduceLROnPlateau scheduler with factor 0.5 and patience 5
- Model checkpointing based on validation accuracy
- Training for 50 epochs with early stopping (patience: 10)

The fine-tuned model was compared against a model trained from scratch using identical hyperparameters to isolate the effect of pre-training.

Regression Training Implementation

For the regression task, I used:

- SmoothL1Loss (Huber loss) to handle outliers effectively
- AdamW optimizer with a learning rate of 0.001
- Cosine annealing scheduler with warm restarts (T_0=10, T_mult=2)
- Gradient clipping at 1.0 to prevent gradient explosion
- Output denormalization for proper error metric calculations

This approach was selected to maximize stability during training while effectively handling the characteristics of the regression task.

Results and Analysis

Performance Evaluation

The evaluation demonstrated clear benefits from the self-supervised pre-training approach:

Classification performance:

- The fine-tuned model achieved 87.3% accuracy compared to 82.1% for the model trained from scratch
- F1 scores showed similar improvements (0.86 vs. 0.80)
- The performance gap widened when using limited labeled data (25% and 50% of available labels)

Regression performance:

- The fine-tuned model achieved consistently lower error metrics
- MSE reduction of 15.7% compared to training from scratch
- RMSE and MAE showed similar improvements (12.3% and 14.1% respectively)

These results confirm that the representations learned during self-supervised pre-training capture meaningful physics features that transfer effectively to downstream tasks.

Analysis and Visualization

The project includes several analytical visualizations:

- Learning curves tracking training and validation metrics
- Comparative performance charts between fine-tuned and from-scratch models
- Activation map visualizations using forward hooks to inspect network behavior
- Regression prediction scatter plots to assess prediction quality across the mass range

These visualizations provide insight into model behavior and validate the effectiveness of the approach.

Comparison with VGG Supervised Approach

Architectural Differences

While the ResNet15 with SimCLR approach leverages self-supervised learning, the VGG architecture implements a fully supervised approach with notable differences:

- The VGG model uses a deeper sequential architecture with smaller kernels compared to ResNet's residual blocks
- The VGG implementation includes Squeeze-and-Excitation (SE) blocks for attention mechanisms, which are not present in the base ResNet15
- The VGG model employs multi-head classifiers for ensemble effect, while the ResNet approach uses a single classification head after fine-tuning

Training Methodology Comparison

- The VGG model used a weighted BCE loss to handle class imbalance, showing a more targeted approach to the dataset characteristics
- The VGG implementation employed a OneCycleLR scheduler compared to ReduceLROnPlateau in the ResNet fine-tuning
- Mixup augmentation was specifically implemented in the VGG model to enhance generalization

Performance Comparison

Based on the evaluation metrics:

- Classification accuracy: The VGG model achieved approximately 84% accuracy on the test set, which is lower than the fine-tuned ResNet (87.3%) but higher than the ResNet trained from scratch (82.1%)
- AUC: The VGG model reached an AUC of 0.90, comparable to the fine-tuned ResNet but with higher computational requirements
- Convergence speed: The VGG model required significantly more training time to reach optimal performance compared to the fine-tuned ResNet

Resource Efficiency

- The ResNet15 with SimCLR approach demonstrated better parameter efficiency with fewer trainable parameters compared to VGG
- Self-supervised pretraining showed superior performance on smaller labeled datasets, requiring only 50% of labels to match VGG's performance with 100% of labels
- The VGG model showed higher memory consumption during training due to its larger parameter count

Visual Feature Learning Assessment

The qualitative analysis shows:

- The VGG model demonstrates strong discrimination between classes but exhibits more confusion in edge cases
- The confusion matrix from VGG shows similar patterns to ResNet but with slightly higher false positive rates
- Visualization of example predictions indicates that the VGG model and fine-tuned ResNet focus on similar regions in the jet images, but the self-supervised approach shows more consistent attention to physics-relevant features

Conclusion and Future Work

The self-supervised learning approach implemented in this project demonstrates significant advantages over training from scratch, particularly when labeled data is limited. The model effectively captures meaningful physics features during the self-supervised phase, creating a strong foundation for fine-tuning.

Compared to the VGG-supervised approach, the SimCLR methodology offers better performance with lower computational requirements and shows superior ability to generalize from limited labeled data. While the VGG model provides competitive performance, the self-supervised approach presents a more efficient path to high-quality representations, especially in scenarios where labeled data acquisition is expensive or limited.

Potential directions for future work include:

- Applying this methodology to other types of physics data
- Comparative analysis with other self-supervised methods (like Barlow Twins, VICReg etc)
- Implementing more specialized augmentation techniques for jet physics
- Exploring deeper architectures while maintaining computational efficiency
- Investigating hybrid approaches that combine the strengths of both VGG's attention mechanisms and ResNet's self-supervised learning

This task successfully combines self-supervised learning with robust fine-tuning strategies for both classification and regression tasks in jet image analysis. The results demonstrate the potential of this approach for physics applications where labeled data may be limited.