Supervised Contrastive Learning Review and Implementation on MRI Brain Image Classification

Aiden Winfield EZM3FG@VIRGINIA.EDU
Maria Cardei CBR8RU@VIRGINIA.EDU
Robert Bao CB5TH@VIRGINIA.EDU
All authors contributed equally to this work.

Abstract

Contrastive learning has experienced a resurgence in self-supervised representation learning, leading to state-of-the-art performance in unsupervised deep image model training. The work introducing supervised contrastive learning by Khosla et al. (2021) extends the self-supervised batch contrastive approach to the fully-supervised setting, enabling the effective use of label information. In this setting, samples from the same class are pulled closer together in the embedding space, while samples from different classes are pushed apart. Two versions of the supervised contrastive loss are introduced and analyzed. We provide a review of this work and implement the authors' provided code on a brain scan image dataset for a classification task. Our implementation and runnable code can be found at https://github.com/rcbao/cs-6501-mlia-supcon . The model performs well with an accuracy and F1 score of up to 0.90 each. We evaluate the effect of various hyperparameters including data augmentation, learning rate, and batch size, and analyze the best performing model configuration.

Keywords: Supervised contrastive learning, medical image classification, brain scan image classification

1. Introduction (Aiden Winfield)

Cross entropy was previously the most widely used loss function for supervised deep learning comparison models. However, many shortcomings have been found with cross entropy such as lack of robustness with regards to noisy labels and the possibility of poor margins which would result in reduced generalization performance (Zhang and Sabuncu, 2018; Elsayed et al., 2018). There was a need to find alternatives for use on large datasets such as ImageNet. The challenge addressed by Khosla et al. (2021) was to find a loss function that can perform as well as, or better than, cross entropy on large datasets while also accounting for the areas which cross entropy lacks. They introduce the idea of *supervised contrastive learning* for the first time.

A recent resurgence of contrastive learning has led to the improvement and development of self-supervised representation learning (Wu et al., 2018). This method pulls together an anchor and a positive sample in an embedding space and pushes away the negative samples. Since no labels are available in this self-supervised setting, the positive sample is usually an augmentation of the anchor, while the negative samples are randomly chosen from the batch. The work introducing supervised contrastive learning follows previous contrastive loss methods, but leveraging provided labels (Khosla et al., 2021). Similar to within the self-supervised setting, this loss function uses different class labels and pulls together embeddings

from the same class while pushing other classes away. By using labels, many positive pairs can be utilized rather than just using one positive sample and the anchor while maintaining the many negative samples (Khosla et al., 2021). The positives are drawn from the same class as the anchor rather than simply being an augmentation as seen in the self-supervised method (Khosla et al., 2021). The loss function is a generalization of the triplet and N-pair losses, which only uses one positive and one negative per anchor, and uses one positive with many negatives respectively (Chopra et al., 2005; Sohn, 2016). By using many positives and negatives for each anchor, this contrastive loss function is able to perform better than cross entropy without needing to use hard negative mining, which can be very difficult to tune well (Khosla et al., 2021). The loss function can also be used for either self-supervised or supervised contrastive learning depending on the learning method and available data.

In this paper, we present a comprehensive review of the original work on supervised contrastive learning and implement the technique on a brain scan image dataset. By applying this method to a medical image classification task, we aim to evaluate its performance and demonstrate its potential to generalize effectively to this domain. We also evaluate the effect of various hyperparameters including data augmentation, learning rate, and batch size, and analyze the best performing model configuration.

2. Background (Aiden Winfield, Maria Cardei)

The work in Khosla et al. (2021) builds on existing research in self-supervised representation learning, metric learning, and supervised learning. There are many papers addressing cross-entropy loss, divulging its drawbacks such as sensitivity to noisy labels and adversarial examples (Zhang and Sabuncu, 2018; Elsayed et al., 2018). Alternative approaches that change the referenced label distribution have been effective, like label smoothing (Szegedy et al., 2016), data augmentations (e.g., Mixup (Zhang et al., 2017), CutMix (Yun et al., 2019)), and knowledge distillation (Hinton et al., 2015).

In self-supervised learning, approaches based on contrastive learning (e.g., noise contrastive estimation (Gutmann and Hyvärinen, 2010) and N-pair losses (Sohn, 2016)) have become popular, with a focus on learning representations by comparing data points. Metric learning, using triplet losses is another related approach, where positive and negative pairs work together to guide the learning process (Chopra et al., 2005). In metric learning, the positive is always chosen from the same class while the negative is chosen from a different class (Chopra et al., 2005). Self-supervised contrastive methods typically use random negative pairs, whereas supervised metric learning often requires hard-negative mining for optimal performance.

The soft-nearest neighbors loss shares similarities to the supervised contrastive loss function (Salakhutdinov and Hinton, 2007). However, improvements were made by normalizing the embeddings and replacing euclidean distance with inner products (Wu et al., 2018). Stronger data augmentation is also used to improve the supervised contrastive loss function along with a disposable contrastive head and two stage training, but the greatest improvement comes from changing the form of the loss function itself (Wu et al., 2018). The Compart Clustering via Label Propagation regularizer also has similarities to supervised contrastive learning, but is used mostly for semi-supervised learning rather than fully supervised learning (Kamnitsas et al., 2018).

3. Methodology (Maria Cardei, Robert Bao, Aiden Winfield)

The methodology provided in Khosla et al. (2021) follows that from self-supervised contrastive learning, with modifications We briefly review their supervised contrastive loss methodology in this section. The derivation from self-supervised contrastive learning can be found in the original work (Khosla et al., 2021).

3.1. Representation Learning Framework

When given an input data batch, the first step is to **augment** the data twice to get two copies of each sample in the batch. These augmentations are random and generate two different views of the data and contain some of the information provided by the original sample. This creates a multiviewed batch. Then, both sets of samples are separately passed through the same **encoder** to produce a pair of normalized representation vectors of size 2048. The vector is then processed by a **projection network** to produce a lower-dimensional vector which is normalized and used for the contrastive loss calculation. The network projection is instantiated as a multilayer perception with a single hidden layer of size 2048 and output vector of size 128, as well as a single linear layer of size 128. To use the model for classification, after contrastive learning, a linear classifier is trained on top of the frozen representations using a cross entropy calculation. This can also be replaced with K-Nearest Neighbor classification or prototype classification.

3.2. Supervised Contrastive Loss Functions

Using this framework, the authors of Khosla et al. (2021) generalize a self-supervised contrastive loss function to two possible functions that incorporate supervision. These are shown below.

$$L_{\text{sup}}^{\text{out}} = \sum_{i \in I} L_{\text{sup},i}^{\text{out}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \tag{1}$$

$$L_{\text{sup}}^{\text{in}} = \sum_{i \in I} L_{\text{sup},i}^{\text{in}} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \right\} \tag{2}$$

The following is an explanation of variables:

- $i \in I \equiv \{1, \ldots, 2N\}$: The index of an arbitrary augmented sample in a multiviewed batch. Each sample i is associated with one or more positives $P(i)$, which are all samples in the batch that share the same class label $\tilde{y}_i$, and several negatives, which are samples with different class labels. $2N$ is the total number of augmented samples in the multiviewed batch, created from N original samples, with two augmentations per sample.
- $A(i) \equiv I \setminus \{i\}$: The set of all indices in the batch except for i, representing the anchors for contrastive comparisons.
- $P(i) \equiv \{p \in A(i) : \tilde{y}_p = \tilde{y}_i\}$: The set of all indices of positive samples in the batch that are distinct from i, sharing the same class label $\tilde{y}_i$.

- $|P(i)|$: The cardinality (size) of the set $P(i)$, representing the number of positive samples associated with i.
- $z_i \in \mathbb{R}^{D_P}$: The representation vector of the anchor sample i, obtained after encoding and projection using the encoder and projection networks. This vector is compared to:
  - z_p: The representation vectors of positive samples $p \in P(i)$, which share the same class label as i.
  - z_a: The representation vectors of all other samples $a \in A(i)$, including both positives and negatives. Negatives are those $a \notin P(i)$, meaning they have a different class label from i.
- $\cdot$: The inner (dot) product operator, used to compute the similarity between representation vectors z_i and z_p or z_a.
- $\tau \in \mathbb{R}^+$: A temperature scaling parameter that adjusts the smoothness of the softmax function in the loss calculation.

The denominator in the log term consists of $2N-1$ terms: $|P(i)|$ positives and $2N-1-|P(i)|$ negatives. The supervised losses push the encoder to more closely align all samples from the same class (positive samples) which improves the robustness of the function.

3.3. Choosing the Loss Function that Removes Bias in Positives

In Equation 1 ($L_{\text{sup}}^{\text{out}}$), the summation over positives is outside the logarithm, while for Equation 2 ($L_{\text{sup}}^{\text{in}}$) the summation over positives is inside the logarithm. The two loss formulations are different due to the concavity of the logarithmic function. Jensen's Inequality implies that $L_{\text{sup}}^{\text{in}} \leq L_{\text{sup}}^{\text{out}}$, making $L_{\text{sup}}^{\text{out}}$ the superior supervised loss function as it upper-bounds $L_{\text{sup}}^{\text{in}}$. Empirical results on ImageNet using ResNet-50 show that $L_{\text{sup}}^{\text{out}}$ achieves significantly higher performance, which is attributed to its better gradient structure and unbiased treatment of positive samples in the multiviewed batch (Khosla et al., 2021). The normalization factor $1/|P(i)|$ in $L_{\text{sup}}^{\text{out}}$ eliminates bias in the positives, improving gradient stability. Each positive sample contributes individually to the loss, so the model focuses more on hard positives and increasing similarity. In contrast, $L_{\text{sup}}^{\text{in}}$ includes the same normalization, but its placement inside the logarithm only contributes an additive constant, making its gradients more susceptible to bias. The contributions of individual positive samples are averaged before the logarithm, masking the influence of hard positives, and therefore reducing the model's ability to improve on challenging cases. Analytical gradient comparisons show that $L_{\text{sup}}^{\text{out}}$ benefits from stabilizing effects due to using the mean representation of positive samples. Consequently, $L_{\text{sup}}^{\text{out}}$ is used exclusively in Khosla et al. (2021), as well as in our experimentation.

4. Experiment (Robert Bao, Maria Cardei)

In this section we discuss our dataset, experimental setup, and results obtained.

4.1. Dataset

To evaluate the aforementioned supervised contrastive learning method, we performed experiments for a classification task utilizing an MRI brain scan image dataset. The dataset is provided by Professor Miaomiao Zhang in the Machine Learning for Image Analysis course at the University of Virginia (UVA). This dataset consists of 1950 total images: 720 healthy brain images, 561 brain images with mild cognitive impairment, and 669 brain images diagnosed with Alzheimer's disease. The images are each 100x76 grayscale images. It is already split into 1657 training images and 293 testing images. For the training images, there are 612 healthy brain images, 477 brain images with mild cognitive impairment, and 568 images of brains with Alzheimer's Disease. For the testing data, there are 108 healthy brain images, 84 brain images with mild cognitive impairment, and 101 images of brains with Alzheimer's Disease.

4.2. Experimental Setup

We performed our experiments on UVA's Rivanna HPC equipped with four NVIDIA V100 GPUs. We implemented the code from the original supervised contrastive learning paper Khosla et al. (2021), using both the TensorFlow and PyTorch implementations available on GitHub.

The **TensorFlow** implementation of their code utilizes TensorFlow v1.15, which is now deprecated and relies on Python v3.7. While we attempted to set up and run this implementation by installing the required dependencies, compatibility issues between software versions posed significant challenges. Consequently, we were unable to successfully execute the code with the appropriate dependencies while also leveraging GPU acceleration. Simultaneously, we implemented the **Pytorch** version, for which we were able to successfully run the code. Within their provided code on Github, we ran the `main_supcon.py` file and successfully ran their experiment on the CIFAR10 dataset, as described in their original work (Khosla et al., 2021). Afterwards, we altered the implementation to work on our dataset.

To obtain the best possible performance, we performed a systemic experimentation pipeline designed to evaluate different configurations of the supervised contrastive learning model on our brain scan classification task. We systemically tested various hyperparameter configurations and data augmentation strategies. The pipeline is described below.

Pre-processing First, we load the brain scan data from `.npy` files, representing the training and testing datasets. for simplicity, we use only the single image channel (i.e. grayscale), and discard any additional unnecessary channels. Additionally, we computed and applied normalization specific to grayscale MRI brain scans (mean = 0.00143, std = 0.00149).

Class Labels CIFAR10 involves ten classes, while our task focuses on three: Healthy, Mild Cognitive Impairment (MCI), and Alzheimer's Disease. The model's output layer and related configurations were updated to accommodate this three-class classification task.

Data Augmentation We define two types of data augmentation: a base transformation and an augmented transformation. The base transformation includes only normalization with a mean of 0.00143 and a standard deviation of 0.00149 that was described in the preprocessing stage. The augmented transformations consist of random cropping with resizing,

random horizontal flipping with a 50% probability, and random rotation with a range of -15 to +15 degrees. This augmentation aims to increase generalizability simulating real-world variations and to decrease overfitting. The goal is to make the model more robust to noise, alignment issues, and variability.

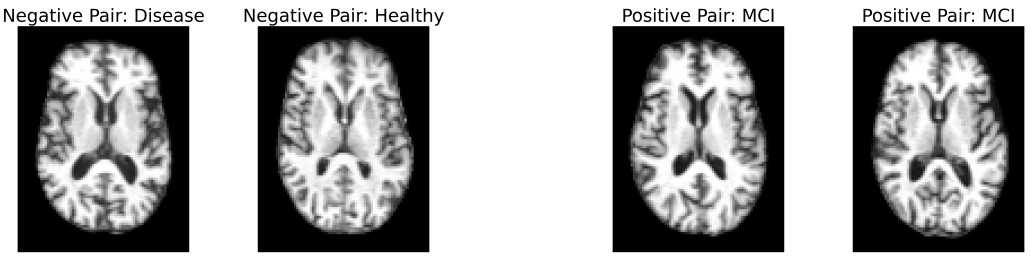
Notably, this transformation setup has subtle differences from the original implementation. The original needs to handle colored images, and as a result, it includes operations like `ColorJitter` and `GrayScale`. These operations, however, are irrelevant in our case since the brain scan images are already in gray scale. Thus, they are removed from the procedure, and we added random rotation in their place to further increase robustness.

Class Imbalance Handling To address the slight class imbalance in our dataset, we utilized a `WeightedRandomSampler` in the DataLoader to ensure balanced sampling during training.

Hyperparameter Search The pipeline then systematically tests combinations of hyperparameters, including a batch size of 16, 32, or 64, a learning rate of 0.01 or 0.05, and whether the base or augmented transformations are used for the augmentation stage. A table of the tests ran is shown in Table 1.

Table 1: Hyperparameter configurations tested during training

Run ID	Batch Size	Learning Rate	Augmentation Type
1	16	0.01	Base
2	16	0.01	Augmented
3	16	0.05	Base
4	16	0.05	Augmented
5	32	0.01	Base
6	32	0.01	Augmented
7	32	0.05	Base
8	32	0.05	Augmented
9	64	0.01	Base
10	64	0.01	Augmented
11	64	0.05	Base
12	64	0.05	Augmented



(*a*) Neg. Examples (Alzheimer, healthy) (*b*) Pos. Examples (MCI)

Figure 1: Example pair of positive and negative samples.

Model Architecture and Training The model used is a *Supervised Contrastive Learning ResNet-50*, which is specifically designed for feature representation learning. As described in Section 3, supervised contrastive learning aims to bring together the representations of images from the same class (positive pairs) while pushing apart representations of images from different classes (negative pairs). For each anchor image, the method utilizes multiple positive images (from the same class) and treats all other images in the batch as negative samples. An example of a positive and negative pair created is shown in Figure 1. The encoder of the ResNet-50 is fine-tuned during training, while a linear classifier is employed on top for final classification. This architecture is implemented in PyTorch.

Training and Validation We utilize four NVIDIA V100 GPUs to train the model efficiently. Training is conducted for up to 200 epochs, with an early stopping mechanism in place if validation loss does not improve for 5 consecutive epochs. The loss function used for the linear classifier stage is cross-entropy loss, optimized using stochastic gradient descent (SGD) with momentum set to 0.9. For each epoch, the training loss and validation loss are logged, and the model with the best validation loss is saved for further evaluation. Validation is performed at the end of each epoch to monitor performance and prevent overfitting.

Evaluation Metrics Model performance is evaluated using a weighted F1-score, which accounts for class imbalances in the dataset, and overall classification accuracy. A confusion matrix is generated to visualize the distribution of predicted versus true labels across all classes. Additionally, a classification report is generated to summarize precision, recall, and F1-score for each class. These metrics provide a comprehensive evaluation of the model's ability to generalize to unseen data.

4.3. Results

In this subsection, we present the outcomes of our experiments, which tested 12 different combinations of hyperparameters, including batch size, learning rate, and data augmentation strategies, on the provided dataset. Our evaluation metrics included precision, recall, F1-score, and overall accuracy.

4.3.1. Hyperparameter Effects

Table 2 shows a summary of performance of all trained models with different hyperparameters. The most consistent yet surprising finding is that models trained on data without data augmentation ("Base" transformations) consistently outperformed those trained with augmented data across all hyperparameter combinations. For example, models trained with Base transformations achieved a peak accuracy of 0.90 (F1 = 0.90) for a batch size of 32 and learning rate of 0.01, compared to only 0.48 accuracy (F1 = 0.44) for the best-performing augmented configuration (batch size = 64, learning rate = 0.01). In the worst case, training on augmented data led to accuracy dropping to as low as 0.29 (F1 = 0.15) for a batch size of 32 and learning rate of 0.05. These results indicate that the current augmentation strategy is ineffective, underscoring the importance of exploring more effective augmentation techniques in future work.

Table 2: Summary of all 12 training runs.

Batch Size	Learning Rate	Transform	Accuracy	F1-Score
16	0.01	Augmented	0.40	0.33
16	0.01	Base	0.84	0.84
16	0.05	Augmented	0.43	0.40
16	0.05	Base	0.89	0.89
32	0.01	Augmented	0.34	0.17
32	0.01	Base	**0.90**	**0.90**
32	0.05	Augmented	0.29	0.15
32	0.05	Base	0.88	0.88
64	0.01	Augmented	0.48	0.44
64	0.01	Base	0.85	0.85
64	0.05	Augmented	0.35	0.24
64	0.05	Base	0.88	0.88

The effect of the learning rate demonstrates mixed trends. For a batch size of 16 with augmented data, a learning rate of 0.05 (accuracy = 0.43, F1 = 0.40) slightly outperformed 0.01 (accuracy = 0.40, F1 = 0.33). A similar trend is observed for Base transformations, where the learning rate of 0.05 (accuracy = 0.89, F1 = 0.89) slightly exceeded 0.01 (accuracy = 0.84, F1 = 0.84). However, for a batch size of 32, the opposite is true: a learning rate of 0.01 led to higher performance for both Base (accuracy = 0.90, F1 = 0.90) and augmented cases (accuracy = 0.34, F1 = 0.17). For a batch size of 64, performance with augmented data decreased from 0.48 (F1 = 0.44) at a learning rate of 0.01 to 0.35 (F1 = 0.24) at a learning rate of 0.05. In contrast, for Base transformations, performance increased slightly from 0.85 (F1 = 0.85) at 0.01 to 0.88 (F1 = 0.88) at 0.05. These results highlight the complex and configuration-dependent relationship between learning rate and performance.

The effect of batch size similarly shows varied results. For augmented data, a batch size of 64 (accuracy = 0.48, F1 = 0.44, learning rate = 0.01) performed better than a batch size of 16 (accuracy = 0.40, F1 = 0.33). However, at a learning rate of 0.05, batch size 16 (accuracy = 0.43, F1 = 0.40) outperformed 64 (accuracy = 0.35, F1 = 0.24). For Base transformations, batch size 32 consistently outperformed other batch sizes, achieving the best overall results (accuracy = 0.90, F1 = 0.90, learning rate = 0.01). These results suggest that while batch size affects performance, its impact varies depending on the learning rate and transform type.

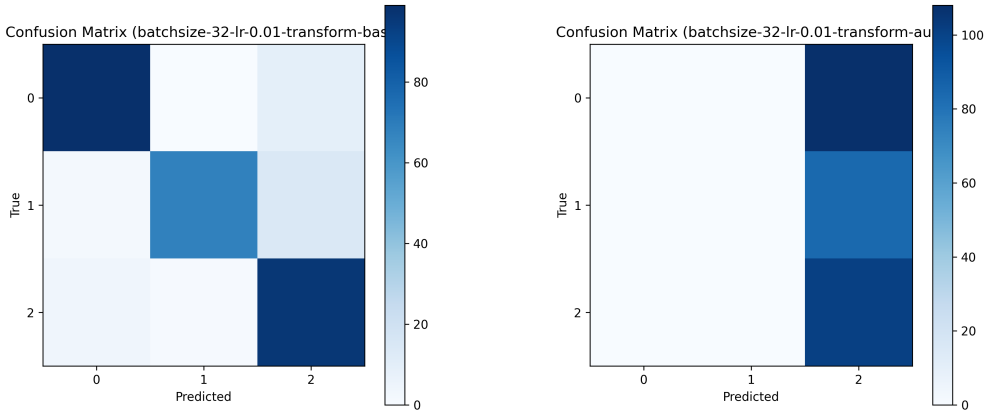
4.3.2. Optimal Configuration Analysis

The highest accuracy – 0.90 – was achieved with a batch size of 32, learning rate of 0.01, and with base transformation. We conduct an in-depth analysis of this model to better understand its behavior, particularly the impact of augmentation since this parameter resulted in consistent effects.

Table 3 shows in-depth classification metrics for this model variation and its counterpart trained to augmented data. As seen, the best performing model variation outperforms in all metrics, including precision and recall, which shows that it is capable of generalize

Table 3: Classification metrics for the supervised contrastive learning model with a batch size of 32, learning rate of 0.01, trained on original and augmented data respectively.

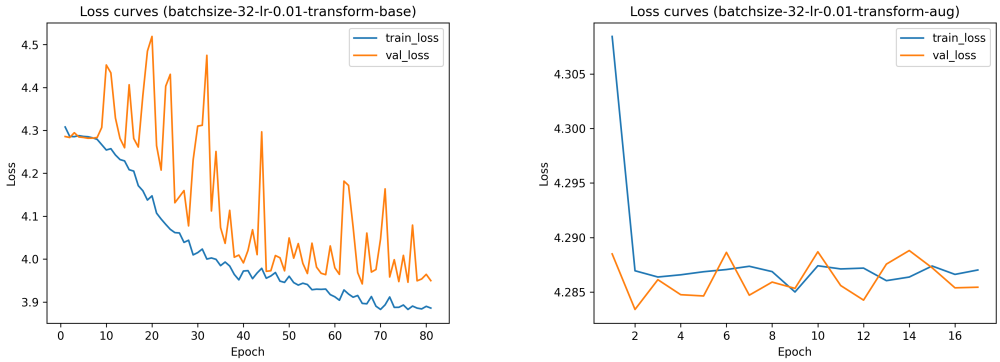
Metric	Base Transform	Augmented Transform
Precision	0.91	0.34
Recall	0.89	0.33
F1-Score	0.90	0.17
Accuracy	0.90	0.34



(a) Base Data (b) Augmented Data

Figure 2: Confusion matrices for the best performing supervised contrastive learning model with batch size of 32, learning rate of 0.01 comparing the case with base and augmented data. Class 0 represents healthy brain images, class 1 represents Alzheimer disease images, and class 2 represents MCI brain images.

learning effectively. Figures 2(a) and 2(b) show the confusion matrices of the model configurations. For the base-data configuration, the diagonal cells of the confusion matrix are heavily highlighted, which shows that it is highly capable of predicting outcomes and made minimal errors. The case with augmented data gives us additional insight into performance, indicating that the model heavily predicted class 2, which were MCI brain images. This suggests that future work could involve inspecting this further with the goal to improve classification for healthy and Alzheimer disease brain images. We additionally investigate loss curves for training runs to analyze the convergence dynamics for model configurations. We compare convergence for our optimal model to the case with augmented data (Figure 3). Figure 3(a), the version with base data, shows steady reduction in training loss. The change in its validation loss is volatile initially, but it eventually stabilizes near the end of training. Therefore the model learned effectively from the base data. In comparison, Fig-



(*a*) Base Data (*b*) Augmented Data

Figure 3: Loss curves for the best performing supervised contrastive learning model with batch size of 32, learning rate of 0.01 comparing the case with base and augmented data.

ure 3(*b*) shows that the model with augmented data failed to converge; its validation loss stagnates early in training, which means that the model underfitted and did not effectively learn from training data. Our setup is designed to stop model training if the loss fails to decrease over more than 15 epochs; thus, the training for the model with augmented data was terminated early. Full evaluation data visualizations for other model configurations are listed in Section 6.

5. Discussion (Maria Cardei, Robert Bao)

In this work, we were able to successfully apply supervised contrastive learning to the brain scan image dataset. Although the model trained on augmented data had limited success, we achieved strong performance with models trained on the original dataset, small batch sizes, and medium to low learning rates. We also carefully addressed the class imbalance in the dataset to ensure it did not negatively impact the model's training or its ability to generalize effectively. Specifically, the model configuration involving a learning rate of 0.01, a batch size of 32, and utilizing the base data performed best with an accuracy and F1 score of 0.90. Future work could improve augmentation techniques, particularly tailoring them to the dataset. This could involve domain-specific augmentations such as variations in brain image intensity, shape distortions, or feature-mixing techniques that better preserve critical features. Additionally, potential measures could be taken to improve validation loss. Some techniques such as dropout and learning rate schedulers can be used to improve model robustness and generalization ability.

Supervised Contrastive Learning Review The technique of supervised contrastive learning is highly innovative, as it adapts contrastive learning—a method traditionally used in unsupervised contexts—to supervised tasks. This enables the model to leverage label information effectively, improving the separation of classes in the representation space. A key advantage of this approach is its ability to learn robust and generalizable features by

explicitly optimizing the similarity between samples of the same class while maximizing the dissimilarity between samples of different classes. Furthermore, supervised contrastive learning has been shown to outperform traditional cross-entropy-based methods, particularly in scenarios where the intra-class variability is high. Its comprehensive and thorough evaluation across different architectures and datasets highlights its versatility and practical relevance.

When examining the original work of supervised contrastive learning, we observe that while the approach demonstrates promising results, it relies heavily on the availability of high-quality labeled data, which may limit its applicability in domains where annotations are scarce or expensive to obtain. Additionally, the method's performance evaluation is often confined to well-curated datasets, leaving its robustness on noisier, real-world datasets underexplored. Future iterations of this work could benefit from strategies to mitigate these limitations, such as semi-supervised learning or leveraging synthetic data for augmentation. We appreciate that the authors include the original code accompanying their paper for replicatino. However, we found it could benefit from more extensive documentation and clearer instructions for running the model successfully on datasets of different sizes.

6. Conclusion (Maria Cardei, Aiden Winfield)

This work highlights the potential of supervised contrastive learning in medical image classification, demonstrating its ability to produce robust and generalizable representations. The method achieved its best performance with base transformations, underscoring the importance of carefully curated data and effective training setups in achieving high accuracy and F1-scores. However, the limitations of the current augmentation strategy were evident, with models trained on augmented data struggling to converge and significantly underperforming. These findings emphasize the need for augmentation techniques better tailored to the nuances of medical imaging.

The adaptability of supervised contrastive learning opens opportunities for future exploration, particularly in addressing class imbalances and optimizing training dynamics for challenging datasets. Extending this work to larger datasets, experimenting with diverse architectures, and integrating semi-supervised or unsupervised techniques could further expand its applicability. With these improvements, supervised contrastive learning has the potential to become a cornerstone methodology for solving complex classification tasks in medical imaging and beyond.

Contributions

All authors contributed equally to this work. AW worked on the introduction and background, and helped describe the methodology and conclusion sections as well. MC led the methodology description, and helped with the implementation and writing of the experimentation. She also worked on the discussion and conclusion sections. RB spearheaded the experimentation and wrote about it, and helped write the methodology and discussion sections.

References

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In *Advances in Neural Information Processing Systems*, pages 842–852, 2018.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Konstantinos Kamnitsas, Daniel C. Castro, Loïc Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya V. Nori. Semi-supervised learning via compact latent space clustering. In *International Conference on Machine Learning*, 2018.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. URL https://arxiv.org/abs/2004.11362.

Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419, 2007.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

Zhirong Wu, Alexei A Efros, and Stella Yu. Improving generalization via scalable neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

Appendix A. Full Evaluation Data Visualizations

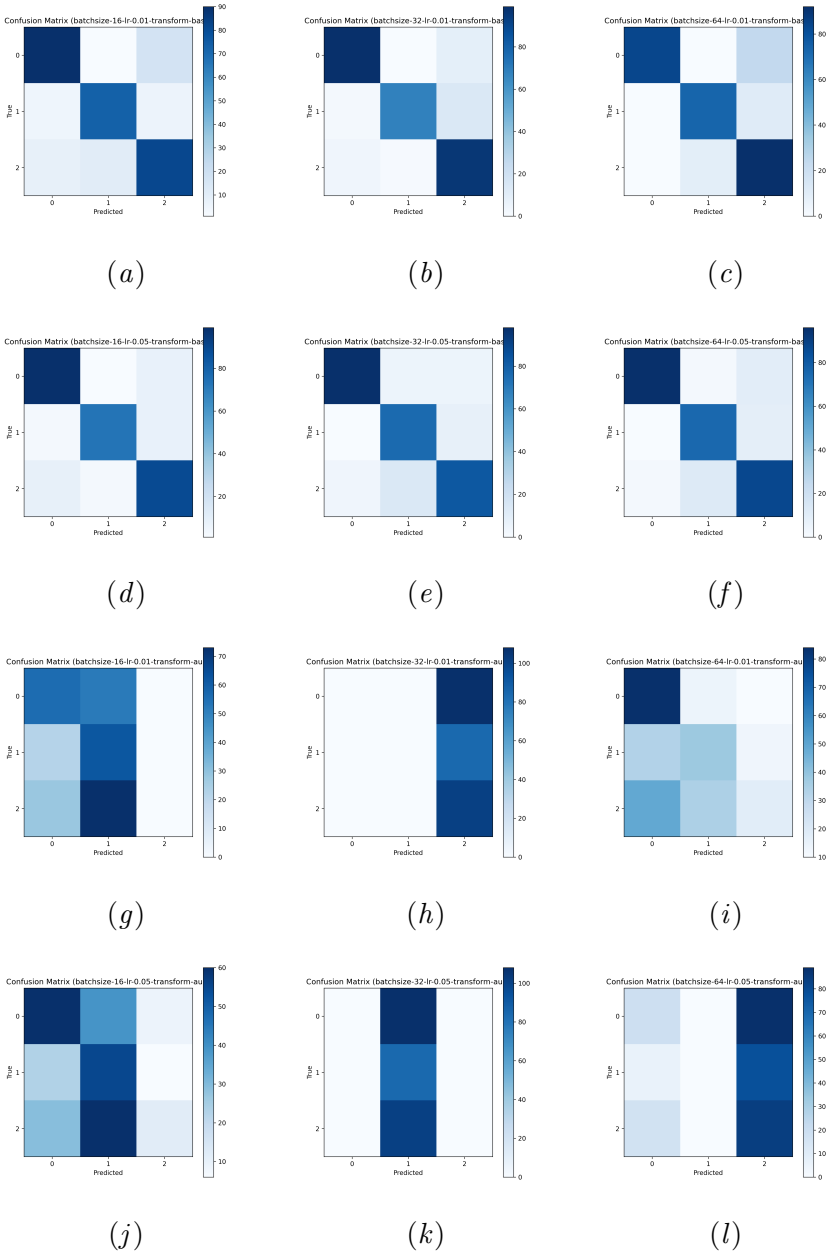


Figure 4: Confusion matrices for all 12 training runs. Each subfigure represents a unique configuration of batch size, learning rate, and data transformation. The subfigures are referenced as follows: 4(a) Batch size = 16, LR = 0.01, Base; 4(b) Batch size = 32, LR = 0.01, Base; 4(c) Batch size = 64, LR = 0.01, Base; 4(d) Batch size = 16, LR = 0.05, Base; 4(e) Batch size = 32, LR = 0.05, Base; 4(f) Batch size = 64, LR = 0.05, Base; 4(g) Batch size = 16, LR = 0.01, Augmented; 4(h) Batch size = 32, LR = 0.01, Augmented; 4(i) Batch size = 64, LR = 0.01, Augmented; 4(j) Batch size = 16, LR = 0.05, Augmented; 4(k) Batch size = 32, LR = 0.05, Augmented; 4(l) Batch size = 64, LR = 0.05, Augmented.

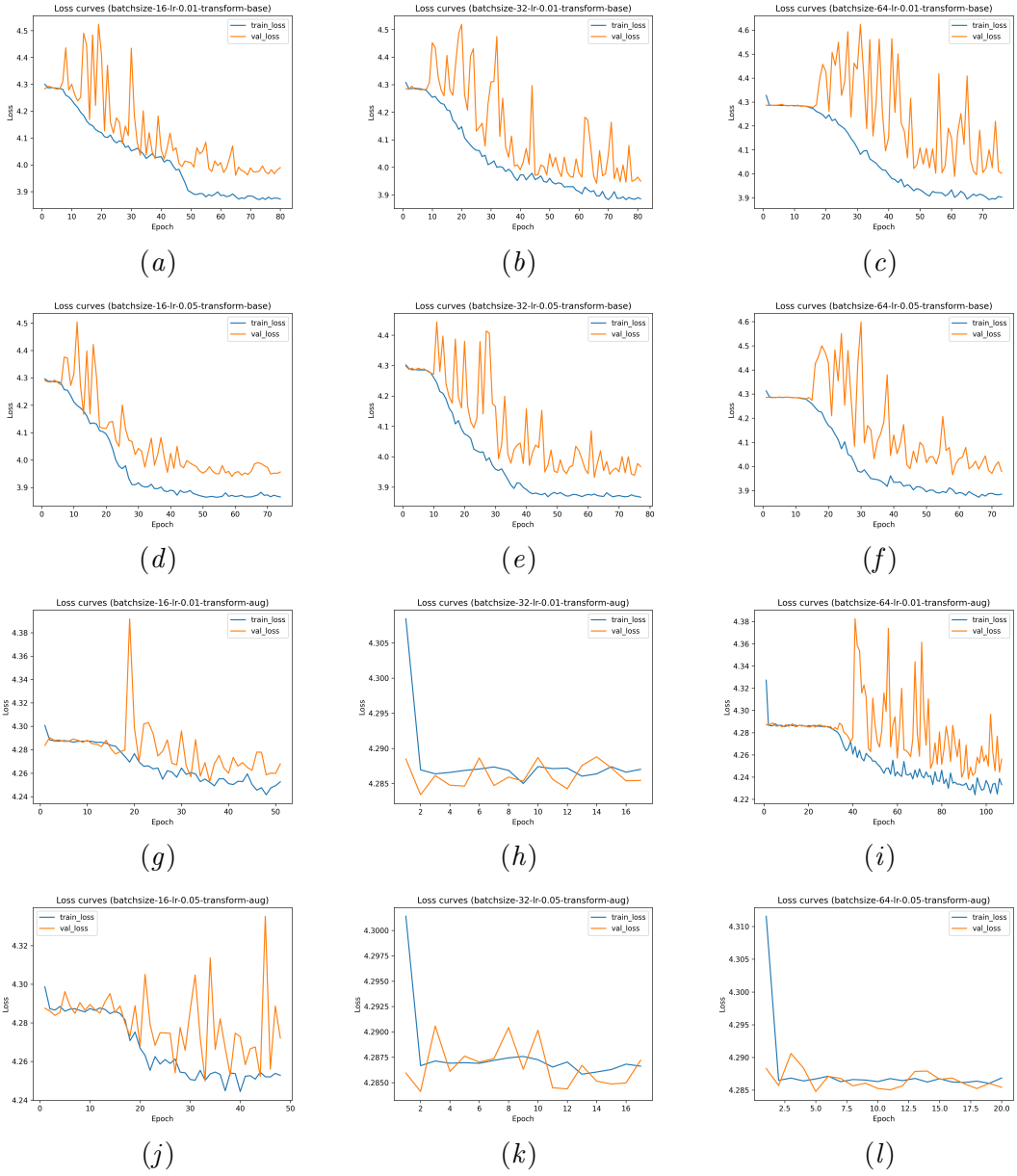


Figure 5: Loss curves for all 12 training runs. Each subfigure represents a unique configuration of batch size, learning rate, and data transformation. The subfigures are referenced as follows: 5(a) Batch size = 16, LR = 0.01, Base; 5(b) Batch size = 32, LR = 0.01, Base; 5(c) Batch size = 64, LR = 0.01, Base; 5(d) Batch size = 16, LR = 0.05, Base; 5(e) Batch size = 32, LR = 0.05, Base; 5(f) Batch size = 64, LR = 0.05, Base; 5(g) Batch size = 16, LR = 0.01, Augmented; 5(h) Batch size = 32, LR = 0.01, Augmented; 5(i) Batch size = 64, LR = 0.01, Augmented; 5(j) Batch size = 16, LR = 0.05, Augmented; 5(k) Batch size = 32, LR = 0.05, Augmented; 5(l) Batch size = 64, LR = 0.05, Augmented.