Boosting Feature Extraction Performance on the aspect of Representation Learning efficiency

by

Haojin Deng
Lakehead University

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS

in the Department of Computer Science

Lakehead University

Boosting Feature Extraction Performance on the aspect of Representation Learning efficiency

by

Haojin Deng
Lakehead University

ABSTRACT

Machine learning is famous for its automatic data handling. While there is a slow growth in the performance of the state-of-the-art models in the most recent well-known learning frameworks, the number of parameters and training complexity rise unaware. Motivated by the present situation, we proposed two efficient methods to enhance the automation on some manual tasks and the efficiency of handling data, respectively. Emotion is one of the main psychological factors that affect human behaviour. A neural network model trained with Electroencephalography (EEG)-based frequency features have been widely used to recognize human emotions accurately. However, utilizing EEG-based spatial information with popular two-dimensional kernels of convolutional neural networks (CNN) has rarely been explored in the extant literature. We address these challenges by proposing an EEG-based Spatial-frequency-based framework for recognizing human emotion, resulting in fewer human-interaction parameters with better generalization performance. Specifically, we propose a two-stream hierarchical network framework that learns features from two networks, one trained from the frequency domain while another trained from the spatial domain. Our approach is extensively validated on the SEED, SEED-V, and DREAMER datasets. The experiments directly support that our motivation of utilizing the two-stream domain features significantly improves the final recognition performance. The experimental results show that the proposed spatial feature extraction method obtains valuable spatial features with less human interaction. Image classification is a classic problem in deep learning. As the state-of-the-art models became more profound and broader, fewer studies were devoted to utilizing data efficiently. Inspired by contrastive self-supervised learning frameworks, we proposed a supervised multi-label contrastive learning framework to improve the backbone model's performance further. We verified our procedure on CIFAR10 and CIFAR100 datasets. With similar hyperparameters and the number of parameters, our approach outperformed the backbone and self-supervised learning models.

Contents

List of Tables

List of Figures

DEDICATION

I would like to dedicate this thesis to,

My parents, who support me spiritually and financially. For always providing me advice from their experience. For letting me concentrate on research without any worries

My girlfriend, who always encourages me when I have difficulties, reminds me to cherish this opportunity and believes in my ability to overcome challenges.

I would also like to dedicate this thesis to my two cats, Owne and Pumpkin, and my chinchilla for being part of my family, always accompanying me while I am working from home and reminding me to eat on time.

Publications

Haojin Deng, Shiqi Wang, W. G. Will Zhao, Hui Zhang, Ruizhong Wei, Q. M. Jonathan Wu, Bao-Liang Lu, Yimin Yang. Classifying EEG Emotions: A Hierarchical Representation Learning Framework with both Frequency and Spatial Domains, under review of IEEE Transactions on Cognitive and Developmental Systems, 2022.

Wandong Zhang, Q. M. Jonathan Wu, W. G. Will Zhao, Haojin Deng, Yimin Yang. Hierarchical One-Class Model With Subnetwork for Representation Learning and Outlier Detection, in IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2022.3166349, 2022.

ACKNOWLEDGEMENTS

I would like to thank:

First of all, **Dr. Yimin Yang**, as my supervisor providing me with a chance to become familiar with academia and work under his supervision through all the tough times during COVID-19. His guidance on my work, passion for academics, and cautious attitude encourage me to learn and pursue being a researcher. It is a pleasure to work with him on the research.

Dr. Ruizhong Wei, as my co-supervisor guidance and help in overcoming some difficulties during my study.

Faculty of Graduate Studies & Faculty of Computer Science for their financial support. As an international student, this research would not have been possible without it.

Chapter 1

Introduction

1.1 Overview

Machine learning has been famous since the 1950s. IBM developed a computer program for playing checkers against human. In the same decades, Rosenblatt implemented the perception [56] by built a machine. However, due to the performance was not promising compared to Symbolic Artificial Intelligence at that time; the perception was not famous. In the late 1950s and 1960s, symbolic AI was able to solve algebra word problems or prove geometry theorems. In 1986, Rumelhart et al. described back-propagation learning procedure [57], which is the cornerstone of many modern machine learning models nowadays. Multi-layer perception(MLP) was a popular machine learning solution in the 1980s. However, the support vector machine (SVM) [5] was invented similarly to last time. As a competitor, SVM is simpler and more efficient. In recent years, because of the reduction of cost of data sample collection and the rapid growth of computing power, MLP and back-propagation became famous due to the success of deep learning [35]. Compared to SVM, neural networks can be composited by different numbers of layers and different types of structures such as CNN (Convolutional Neural Network) [36], LSTM (Long Short Term Memory) [26], Transformers, GAN (Generative Adversarial Networks) [67], etc. to handle various problems.

1.2 Background

This study aimed to efficiently use underlying features from one-dimensional electroencephalogram(EEG) data and image data to enhance the performance of classification tasks.

1.2.1 Machine Learning Categories

Nowadays, self-supervised learning became popular in machine learning. The relationship between self-supervised learning with unsupervised learning supervised learning and especially semi-supervised learning could be confusing. Since our thesis study involved self-supervised learning materials, we want to discuss their relationship before moving on to the next step.

Supervised Learning

Supervised learning is the traditional method used in specific tasks that involved labels such as classification and regression.

Unsupervised Learning

Unsupervised learning needs fewer human interactions compared to supervised learning. It does not require human labels. Unsupervised learning could be utilized to discover data patterns or analyze data by clustering or Principal component analysis (PCA). It can also use generative adversarial networks [67] to generate data or enhance resolution.

Semi-supervised Learning

Semi-supervised learning used a small amount of labelled data and a large amount of unlabelled data during training. The model is first trained by the labelled data, then uses the underlying distribution in data to label the unlabelled data. After that, the model was trained with both labelled data and pseudo-labelled data.

Self-supervised Learning

Similar to semi-supervised learning, self-supervised learning is combined with unsupervised learning and supervised learning. There are two steps in self-supervised learning: pretraining and finetuning. The pretraining step learns data representations without labels (unsupervised learning). The finetuning step is similar to supervised learning, training the backbone model by labels. Based on different researches [10,30], self-supervised learning with 10% of labelled data and 90% of unlabelled data could reach similar performance as supervised learning with 100% labelled data. We would introduce more about self-supervised learning in Chapter 3.

Difference between Self-supervised Learning and Semi-supervised Learning

These two categories are easy to get confused with because both of them involved unsupervised learning and supervised learning. However, that is the one common between these two categories. The Essential difference is that semi-supervised learning required a small amount of labels to train the model initially. Self-supervised learning pretraining tasks do not require any labels and they learn based on the underlying structures of data.

1.2.2 Electroencephalogram

Emotion recognition has been a popular research topic recently [13, 16, 38, 46, 70, 72]. The most common emotion recognition method is through human physical signals such as facial expressions, posture, etc., but also through physiological signals. Physiological signals are more reliable and largely involuntarily activated compared to physical signals [63]. Valenza et al. [68] adopted electrocardiogram (ECG) signals to recognize emotions. Wen et al. [71] used galvanic skin response (GSR), fingertip blood oxygen saturation (OXY) and heart rate (HR) to classify five emotions. Khadidja et al. [20] classify emotions based on electromyogram signal (EMG), respiratory volume (RV), skin temperature (SKT), skin conductance (SKC), blood volume pulse (BVP) and heart rate (HR). Different from physiological signals above, electroencephalogram (EEG) reflects the emotional stimulus in the brain [34].

Since 1985 [1], the EEG has been widely used because of its stability and easy accessibility than other biological signals [48]. Due to the rapid development of machine learning algorithms in recent years, the use of machine learning to analyze EEG signals for emotion analysis has achieved good results [29].

In the neuroscience field, EEG signals usually contain five frequency bands, such as delta (δ: 1–3 Hz), theta (θ: 4–7 Hz), alpha (α: 8–13 Hz), beta (β: 14–30 Hz), and gamma (γ: 31–50 Hz) [50] to examine their relationship with the emotional states. Alpha power asymmetry is a common indicator for evaluating emotional states, and there are spectral differences in asymmetric pairs of electrodes in the anterior regions of the brain [28]. Emotion states were also associated with spectral changes in other parts of the cerebral cortex, such as right parietal alpha changes, theta power changes in the right parietal [59], frontal-midline (FM) theta power [58], power asymmetry in the beta-parietal region, and gamma spectrum changes in the right parietal region [3].

1.2.3 Self-supervised learning on Image Classification

Before the contrastive learning framework came out in self-supervised learning, pretext tasks were the primary choices to learn image representations without any labels. For example, Zhang et al. input a grey-scaled image and use a model to restore the colour of the image [78]. Gidaris et al. rotate the input image and use the model to

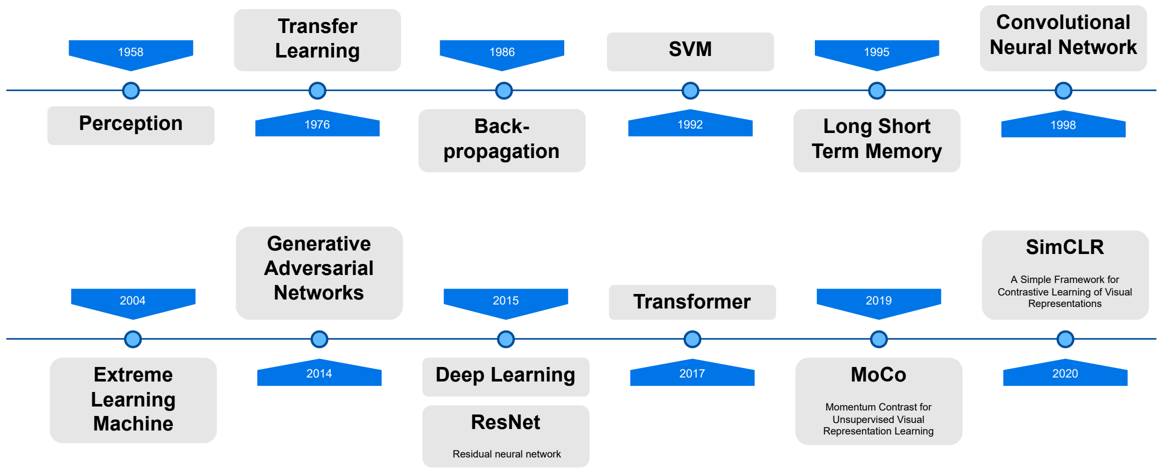


Figure 1.1: Timeline of machine learning related works.

predict the rotation angle [18]. Noroozi cropped the input image into several different blocks and trained the model to predict the correct location of each block(Jigsaw puzzles) [54]. To prevent the model learns the shortcut, the author removed the boundaries between the blocks. Pathak et al. removed pixels from the image and trained the model to predict the pixels. The contrastive learning framework is different from pretexts. It normally augmented one image with the stacking of different processing methods such as cropping, rotating, resizing, colour distorting, blurring etc. The contrastive learning model augmented the original image with different random augmentations and learnt the underlying similarities based on two augmentations.

1.2.4 Scope of the Study

Our studies explored the spatial feature and feature fusion of emotional recognition EEG datasets and contrastive learning in self-supervised learning area.

1.3 Related work

With the development of technology (data collection, computing power) nowadays, machine learning could be applied to different types of data, such as signal data, image data, and time series data with different structures effectively. This section introduces some famous structures related to our studies and different applications seniors in machine learning.

1.3.1 Multi-layer Perception

Multi-layer perception (MLP or "vanilla" neural networks) contains one or more hidden layers. Sometimes MLP layers in deep learning are also called fully-connected layers because each layer's neurons are connected to all the neurons in next layer. MLP model could be trained iteratively. The model updates the weight in each layer by back-propagation [57]. Due to the MLP being compatible with nonlinear activation, it could solve nonlinear separable problems. However, when researches start to explore the computer vision tasks, MLP efficiency appears to be very low. First, flatten image pixels as the input of MLP model would require a much larger number of parameters in the model. Second, the flatten step and the fully connection step could destroy the spatial representations in a image.

1.3.2 Support Vector Machine

Support Vector Machine(SVM) was first introduced in 1992 [25]. SVM has a higher speed and good generalization. Compared to MLP, SVM has better performance with limited training samples. SVM assumes there exists at least one hyperplane that could separate different category data with a linear or nonlinear decision boundary. The goal for SVM is to find that hyperplane and also find the decision boundary that maximizes the margins on both sides.

1.3.3 Convolutional Neural Network

To address the above challenges in computer vision tasks, CNN [36] was invented in 1998 to learn two-dimensional image semantics. Similar to MLP, CNN updates the parameter weights by back-propagation. However, CNN used a small kernel to learn local features by convolution. Each layer output could preserve the spatial information, and the number of the parameter is undersized compared to MLP. The convolutional neural network is famous and wins ImageNet competitions. As a trade-off for fewer parameters, convolutional layers need more calculations and training iteratively also slows down the training process.

1.3.4 Extreme Learning Machine

Huang et al. proposed ELM (Extreme Learning Machine) [27] in 2004. This method has good generalization performance and promising efficiency. It is a single-layer feedforward neural network(SLFN). However, there are some differences between ELM and vanilla neural network. The traditional vanilla neural network contains weight, bias and an activation function in each layer so this calculation step would process at least three times depending on the number of layers. After the model calculates the loss between predicted output and true labels in the vanilla neural network, it uses back-propagation to update weights and the process takes several iterations. ELM is much efficient. It only contains one hidden layer and the feedforward process is much faster. More significantly, ELM used Moore-Penrose generalized inverse to calculate the weight based on the true label and the predict output. Back-propagation and iterative training process is not required for ELM.

1.3.5 Deep Learning

Deep learning is not a new framework or machine learning structure, it is more likely a natural result or stage as our computing power increases to a certain point. Normally, if we add more layers in CNN, for example, 30 layers then it is called DCNN(deep convolutional neural network) just because it has more layers and more parameters. However, the gradients resemble white noise as the model became deeper, in other words, it's the shattered gradients problem [4]. ResNet [24] is one of the solutions to this problem. It kept the gradient more efficient during Back-propagating.

1.3.6 Generative Adversarial Networks

As more complex frameworks' training processes can be completed in an acceptable time, different types of the framework became popular. Benefiting from deep learning, GANs(Generative Adversarial Networks) could generate vivid images and videos as a generative model. This framework contains two models. One is a generator model and the other one is a discriminator model. In a training step, the generator model generates a sample and the discriminator model tries to distinguish the real one between this generated sample and the real data sample. This is a clever framework, the two models designed to be conflict with each other. The generator model could learn useful representations even without labels.

1.3.7 Transfer Learning

Transfer learning technique was founded dating back to 1976 [6]. It became popular as training a deep learning model from scratch takes a lot of time. If we pretrain a model in a large dataset, then finetune the model in our target dataset, the convergence speed would likely be faster. Usually, the large dataset and the target dataset should be similar, such as the ImageNet dataset and the CIFAR10 dataset. However, we believe transfer learning still works across different types of datasets (image and signal datasets). The detailed experiment would be given in Chapter 2.

1.4 Research Objective

In recent years, more studies proposed deeper neural networks. As the number of parameters and layers grows promptly, the model's performance increases insignificantly. Are deeper networks better? We do not know the answer yet. Nowadays, many researchers aim to increase the performance of the neural network by adding more parameters to the model. Although the large models such as GPT-3 [7] (175B), BERT [14] (110M), and Transformers [69](86M, Vision Transformer) have outstanding performance and generalization, the required computational power limited their usage on many applications such as smartphones and other smart devices. Zhang et al. proposed OPT [79] that only required 1/7 of GPT-3's training carbon footprint while preserving similar performance. However, due to the size and computation resources required to train these models, these technologies are difficult to apply on existed works.

Therefore, we aimed to propose new training patterns that could be compatible and boost existing technologies. Our training patterns could enhance representation learning efficiency without needing extensive computational resources or large amounts of training parameters. CNN replaced MLP in computer vision due to higher performance and fewer parameters. ResNet encouraged feature reuse and prevented the vanishing gradient problem with lower complexity than other studies. Based on the above facts, we reasonably hypothesize that the classification problem still has room to gain performance by optimizing the learning patterns of data representation.

1.5 Contribution

Based on the hypothesis, we proposed optimizing patterns to boost the feature extraction performance in one-dimensional EEG data and two-dimensional image data. Based on the characteristic of EEG data, we proposed DCNet to extract spatial features efficiently. Inspired by SimCLR [10], we proposed a multi-label bidirectional contrastive learning framework with a similar amount of parameters but significant higher generalization ability compared to SimCLR. Below are the main contributions of this thesis work.

1. We presented a two-stream hierarchical network framework (HNSN-DCNet) and reached state-of-the-art performance in the SEED dataset.
2. We applied DCNet for spatial feature extraction. This method is more efficient than traditional spatial feature extraction methods due to less human-made inference. Similar to the perception, DCNet has better adaptability and flexibility.
3. We proposed a supervised contrastive learning representation framework (SupCLR). SimCLR [10] claimed they have better performance than similar backbone supervised learning methods. With similar parameters and backbone, our proposed method performs better on CIFAR10 and CIFAR100 than SimCLR.
4. We analyzed SupCLR in Chapter III. Ablation Study section. Our experiment reflects projection head is not a necessary part during training compared to SimCLR. Even with fewer parameters, our framework has faster convergence speed and better performance compared to SimCLR.
5. Our proposed representation learning patterns could be used in different tasks to improve generalization and overall performance by gaining more comprehensive underlying representation features.

1.6 Organization of Thesis

This section was all about the introduction and the rest of the thesis proceeds as follows,

Chapter 2 provides the detail of the two-stream hierarchical network framework

(HNSN-DCNet), including the complexity analysis and our framework's experiments on the SEED, SEED-V and DREAMER Dataset.

Chapter 3 introduced the recent self-supervised learning frameworks and their differences. This chapter also presents the SupCLR framework, the proposed loss function and the theoretical analysis of the upper bound of the loss function compared to NT-Xent [10] loss function. After that, the experiment and ablation study sections explained the robustness and performance of our framework.

Chapter 4 is the last chapter in this thesis, which concludes the whole work done in this thesis and further explains the future prospects of our studies.

Chapter 2

Classifying EEG Emotions: A Hierarchical Representation Learning Framework with both Frequency and Spatial Domains

2.1 Introduction and Related Work

Machine learning methods has been used in emotion recognition and went well. Early machine learning methods methods (single-layer feed-forward network, fuzzy k-means and support vector machine (SVM)) achieved moderate results in the emotional classification of multiple emotional (two or more) states in experiments [22, 45, 53]. For example, Lin et al. [44] used the f-score index, based on the ratio of emotion recognition between and within classes. By classifying four emotion states at 26 electrodes, they achieved an average accuracy of 82.29%. Chanel et al. [9] used time-frequency data to identify three emotions, obtaining 63% accuracy, and fused different features and samples, achieving 80% mean accuracy. Zheng et al. [87] preprocessed EEG signals by the differential entropy (DE) [15] method, grouped all electrodes, selected the characteristic data of 12 electrodes and classified them by SVM. They obtained the best accuracy of 86.65%. Later, Yang et al. [76] proposed a neural network (NN)-based emotion recognition with subnetwork nodes and got mean accuracy of 91.37% on DE features from full channels.

Recently, deep neural networks have achieved good results in various fields, especially image and video processing [83]. Gong suggested deep learning has overall better performance compared to conventional methods because deep learning model do not require specific knowledge and expertise [19]. For example, Zheng et al. [86] used DE features to train a deep belief network (DBF) and achieved a mean accuracy of 87.62%. Unsupervised learning methods, such as auto-encoder, are also used for extracting deep features from EEG signals. Most recently, Li et al. [39] proposed the Fast Online Instance Transfer(FOIT) to avoid time consuming iterations and complex optimizations by training various non-iterative models. Li et al. [38] designed a 3-D Feature Representation and Dilated Fully Convolutional Networks (3DFR-DFCN) to fully capture prior knowledge into the 3-D array and use spectral norm regularization (SNR) to reduce the sensitivity for improvement of the generalization performance of DFCN. Their classification accuracy on the DREAMER dataset is 93.15%, 91.30% and 92.04% for valence, arousal and dominance, respectively. Wang et al. [70] used short-time Fourier transform (STFT) to convert 1 dimensional EEG signal to 2-dimensional signal and processed by CNN with residual block to avoid gradient disappearance and gradient explosion. Wu et al. [72] proposed an emotion-relevant critical subnetwork selection algorithm to remove the remaining noise and used Pearson's correlation coefficient and spectral coherence to find the correlation between signals. They got the state-of-the-art classification performance in several EEG datasets. On the other aspect, many studies reflect that pairwise learning could benefit EEG emotion recognition tasks [30, 40, 88]. Li proposed an effective joint distribution adaptation (JDA) method [40] by contrasting the similarity of pairs of data samples with adversarial training. Zhou et al. also proposed a novel pairwise learning method (PR-PL) [88] to learn the inherent relationship between different pairs of samples, and between sample features and prototype features. Kan et al. proposed a self-supervised contrastive learning framework (SGMC) to learn representations of EEG data samples [30]. They proposed the Meiosis augmentation method based on EEG 1D-signal data. The Meiosis method cross-exchanged part of a pair of data samples corresponding to the same stimuli. The result 89.65% on the SEED dataset is promising in the finetuning task even with 1% labelled data.

Yang et al. [76] selected signal data of special regions of the cerebral cortex and used these data for emotion recognition, and obtained positive results, further proving that the cerebral cortex has different reflection regions for different emotions, i.e., a spatial relationship exists between different electrode channels. However, in most existing studies [37, 43, 52], EEG spatial features are extracted manually based on specific dataset channels and the location according to the International 10-20 system. For example, the authors proposed the Pearson correlation coefficient (PCC), phase locking value (PLV), and transfer entropy (TE) to extract spatial features in [52]. Their framework required the geometry of electrodes for ordering, which restricted the number and position of channels in the application. This reliance on manual treatment makes it questionable if many existing spatial feature extraction methods could be applied to another dataset with guaranteed performance. Furthermore, low signal-to-noise ratio and difficulty improving accuracy are still significant difficulties with EEG datasets [82].

Motivated by above identified shortcomings in existing research, this paper proposed a novel spatial feature extraction framework (DCNet) with preprocessed 1-D EEG data. As Abdullah et al.'s found [2], convolutional layers as locally-adaptive thresholding procedures can prevent the noise between channels during training. We hypothesize that the deconvolution-convolution structure can focus on local features and extract spatial feature. We provide feature visualization experiments to verify this hypothesis. To further represent the advantage of our spatial feature extraction method, we combined it with Yang's frequency domain feature extraction method [76] (HNSN) as Fig. 2.1.

This framework is suitable for any EEG dataset with different channel numbers and does not require any spatial domain preprocessing step. From experiments result, the framework improves generalization performance from different stream features. With the fewer human-interaction parameters advantage, the backbone of DCNet layers could be used to transfer learning to other EEG datasets.

In particular, this paper makes the following contributions.

1. We proposed a spatial feature extraction method (DCNet) to convert 1-D EEG signal data into 2-D latent space and then use CNN structure to extract local spatial features. The feature visualization experiments on different datasets reflect that the performance of our spatial feature is promising.

2. We propose a two-stream hierarchical network framework (HNSN-DCNet) learning representations of features combined from multiple networks. The different streams can extract the frequency and spatial features, significantly improving accuracy. Evaluated by three widely selected datasets (SEED, SEED-V and DREAMER), the accuracy of our proposed framework outperforms other state-of-the-art methods. Furthermore, due to less human-made inference on the spatial feature extraction method, this framework has better robustness and stability when applied to other EEG datasets.

3. The high portability of DCNet allows our framework to enhance the generalize ability by transfer learning. Based on our experiment (Fig. 2.9), the performance improved significantly with transfer learning from the Imagenet dataset. This could reduce the high variance caused by small training samples in EEG datasets.

2.2 Method

2.2.1 Preprocess and Feature Extraction

Preprocess on SEED and SEED-V dataset

First, the sampling rate of SEED and SEED-V raw data was downsampled from 1000 Hz to 200Hz. Then both datasets were processed by a bandpass filter between 0- 75 Hz.

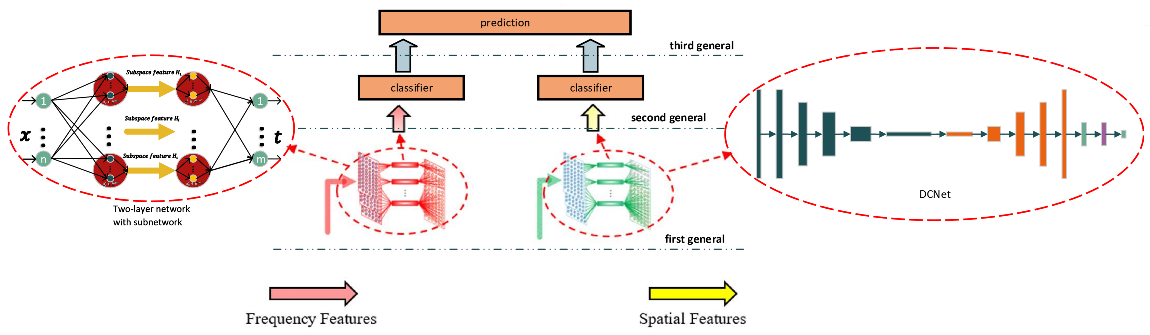


Figure 2.1: Proposed framework, 1) 1st general layer: feature extraction. 2) 2nd general layer: classifier with subnetwork nodes. 3) 3rd general layer: late fusion.

Differential Entropy Features on SEED and SEED-V dataset

Duan [15] proposed the effective features called differential entropy (DE) extend the concept of Shannon entropy and are used to measure the complexity of continuous random variables. Since EEG data has high low-frequency energy in high-frequency energy, DE can distinguish the EEG mode between low-frequency and high-frequency energy. Li [40] also suggests that using DE to extract features performs better than power spectral density (PSD).

The original calculation formula of differential entropy is defined as

$$H(x) = -\int_x f(x)\log(f(x))dx. \tag{2.1}$$

If the time series X obeys the Gauss distribution N(μ,δ), the DE features can be obtained by

$$\begin{aligned} H(x) &= -\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{(x-\mu)^2}{2\delta^2}} \log\left(\frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{(x-\mu)^2}{2\delta^2}}\right) dx \\ &= \frac{1}{2}\log 2\pi\delta^2, \end{aligned} \tag{2.2}$$

where the time series x obeys the Gauss distribution $N(\mu,\, \sigma^2)$. Experiments show that for fixed-length EEG signal sequences, DE is equivalent to the logarithm energy spectrum in a certain frequency band [61]. We use this method to extract the DE features of 5 corresponding frequency bands.

DE features can be converted to 5 different frequency bands (delta: 1-4 Hz, theta: 4-8 Hz, alpha: 8-13 Hz, beta: 13-30 Hz, gamma: 30-50 Hz) with time complexity

$O(KN \log N)$ where N is the size of samples, and K is the number of electrodes.

Then, we use the linear dynamic system (LDS) method to further filter out irrelevant components, and consider the temporal dynamics of emotional state [62].

Preprocessing on DREAMER dataset

For each participant, the DREAMER dataset contains baseline signals recorded without any stimulation and stimulated signals evoked by watching the video clips. Here is the preprocessing steps for each participant.

Step-1: Separate the current participant's baseline signal X_b into one-second trials $X_{b1}, X_{b2}, X_{b3} \cdots X_{bn}$.

Step-2: Calculate the mean $X_{b\mu}$ of the participant's all one-second trials to get the average baseline signal.

$$X_{b\mu} = \frac{\sum_{i=1}^{n} X_{bi}}{n}.$$

Step-3: Separate the current participant's stimulated signal X_s into one-second trials $X_{s1}, X_{s2}, X_{s3} \cdots X_{sn}$.

Step-4: Use each stimulated signal to subtract the baseline average trail to remove the mean and standardize the feature by scaling to unit variance σ.

$$X_{sj}^{'} = (X_{sj} - X_{b\mu})/\sigma.$$

Step-5:After all, we merge all the preprocessed one-second trails together to get $X_s^{'}$ which has the same shape as X_s.

2.2.2 First General Layer: Feature Extraction

Hierarchical Network With Subnetwork Nodes

A two-layer autoencoder has been introduced [75] where only the encoding layer weight has been generated randomly, based on which the decoding layer weight has been calculated. The autoencoder aims are to minimize the reconstruction loss, which is the squared error between the input X and the neural network output $\hat{Y}$.

In this paper, we use Hierarchical Network with Subnetwork Nodes (HNSN) [76] as our frequency feature extraction method. Zhang et al. [81] also proved the convergence of this algorithm. We briefly describe the Autoencoder algorithm in the algorithm 1.

Step-1: Given M arbitrary distinct training samples $(\mathbf{x}_k, \mathbf{y}_k)_{k=1}^{M}, \mathbf{x}_k \in \mathbf{R}^n$ are sampled from a continuous system. Randomly initialize the entrance layer subnetwork node:

$$\mathbf{H}_f^c = g(\hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c, \mathbf{x}), (\hat{\mathbf{a}}_f^c)^T \cdot \hat{\mathbf{a}}_f^c = 1, (\hat{\mathbf{b}}_f^c)^T \cdot \hat{\mathbf{a}}_f^c = 1, \tag{2.3}$$

where $\hat{\mathbf{a}}_f \in \mathbf{R}^{d \times n}, \hat{\mathbf{b}}_f \in \mathbf{R}$ is the orthogonal random weight and bias of the entrance mapping layer. $\mathbf{H}_f^c$ is the c-th subspace features. c represents subnetwork node index and initial index $c = 1$.

Step-2: Given an invertible activation function g, obtain the subnetwork node of the exit feature layer $(\hat{\mathbf{a}}_h^c, \hat{\mathbf{b}}_h^c)$ by

$$\begin{aligned} \hat{\mathbf{a}}_h^c &= g^{-1}(u_n(\mathbf{y})) \cdot (\mathbf{H}_c^c)^{-1}, \hat{\mathbf{a}}_h^c \in \mathbf{R}^{d \times m}, \\ \hat{\mathbf{b}}_h^c &= \sqrt{mse(\hat{\mathbf{a}}_h \cdot \mathbf{H}_f^c - g^{-1}(u_n(\mathbf{y})))}, \hat{\mathbf{b}}_h^c \in \mathbf{R}, \end{aligned} \tag{2.4}$$

where $\mathbf{H}^{-1} = \mathbf{H}^T(C1/I + +\mathbf{H}\mathbf{H}^T)^{-1}; C1 > 0$ is a regularization value; u_n is a normalized function $u_n(\mathbf{y}) : \mathbf{R} \to (0, 1]; g^{-1}$ and u_n^{-1} represent their reverse function.

Step-3: Update the output error $\mathbf{e}_c$ as

$$\mathbf{e}_c = \mathbf{y} - u_n^{-1} g(\mathbf{H}_f^c, \hat{\mathbf{a}}_h^c, \hat{\mathbf{b}}_h^c). \tag{2.5}$$

We can get error feedback data $\mathbf{P}_c = g^{-1}(u_n(\mathbf{e}_c)) \cdot (\hat{\mathbf{a}}_h^c)^{-1}$.

Step-4: Update the subnetwork node $\hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c$ in the entrance layer

$$\begin{aligned} \hat{\mathbf{a}}_f^c &= g^{-1}(u_j(\mathbf{P}_{c-1})) \cdot \mathbf{x}^{-1}, \hat{\mathbf{a}}_f^c \in \mathbf{R}^{n \times d}, \\ \hat{\mathbf{b}}_f^c &= \sqrt{mse(\hat{\mathbf{a}}_f^c \cdot \mathbf{x} - \mathbf{P}_{c-1})}, \hat{\mathbf{b}}_f^c \in \mathbf{R}. \end{aligned} \tag{2.6}$$

Step-5: obtain the c-th subspace feature data

$$\mathbf{H}_f^c = g(\mathbf{x}, \hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c). \tag{2.7}$$

Step-6: Set $c = c + 1$, add a new subnetwork node $\hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c$ in the feature mapping layer with orthogonal random initialization.

Step-7: Repeat steps 2 to 6 $L - 1$ times, then obtain the L subspace features $[\mathbf{H}_f^1, ..., \mathbf{H}_f^L]$.

Deconvolution and Convolution Network

To ensure that manual intervention is avoided, we chose to use DCNet(Deconvolution-and-convolution Network) [77] to extract spatial features from EEG signals. This end-to-end architecture can perfectly train any different channel size 1D signal data. The DCNet is an asymmetric framework. The encoding part of DCNet automatically converts the 1D input signal into a 2D matrix. Then the decoding part of DCNet uses 2D CNN for classification.

The input dimension is 1D EEG signal data. The encoder reshapes the input data (reducing channels and increasing dimensionality) into $32 \times 32 \times 3$ matrix by five transposed convolution layers. For the input Y^{l-1} (the output from the last layer) and the kernel k with size (M, N), we define the transposed convolution layer as follows:

$$T^{l}(i+a, j+b) = \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} k(a,b) * Y^{l-1}(i,j), \tag{2.8}$$

where i, j, a, b are the index of the input and the index of the kernel respectively. Then decoder converted the matrix back to 1D-shape data by four convolution layers with the max-pooling layer that extracts the spatial feature from the matrix. For the input Y^{l-1} (the output from the last layer) and the kernel k with size (M, N), we define the convolution layer as follows:

$$C^{l}(i,j) = \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} k(a,b) * Y^{l-1}(i+a, j+b). \tag{2.9}$$

Unlike other resize techniques, deconvolution or transposed convolution layer contains trainable parameters. Most existing processes need human-made inference to convert the 1D signal to spatial features. In contrast, the parameters in deconvolution layers could be updated during model training. The promising performance of this framework on 1D datasets has been proved by Dr. Yang [77] by 12 different datasets.

Normally the input data shape is (*sample, channelNumber*). To compatible with trasposeConv2d layer, we reshape the data input to (*minibatchSize, 1, 1, channelNumber*). The feature extraction output is the first dense layer(*minibatchSize, 1, 1, 80*) in Fig. 2.2. We have made some modifications to adapt the DCNet to our framework. In both deconvolution and convolution layers, we used Selu [32] activation function because it can effectively prevent vanishing and exploding gradient

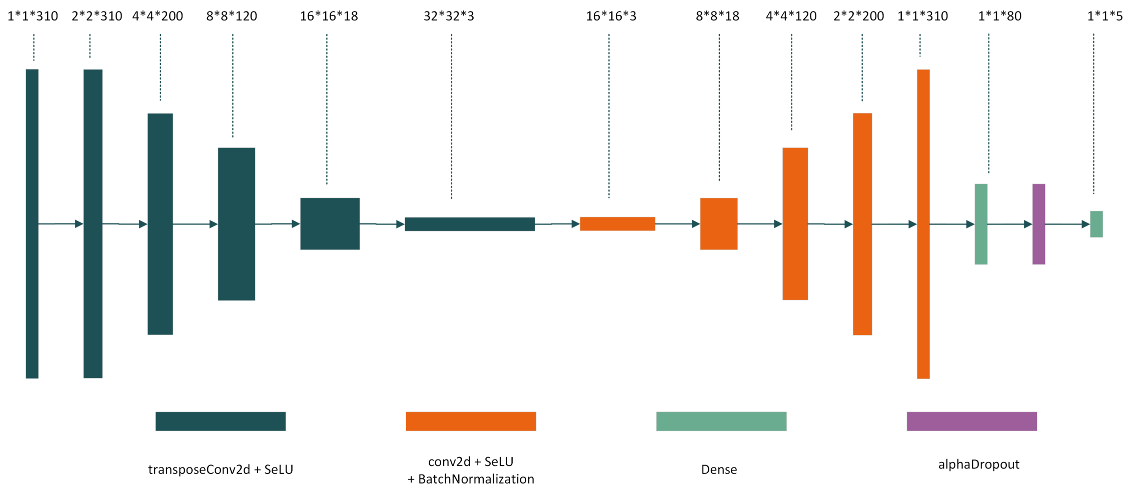


Figure 2.2: Our proposed DCNet structure. For the encoder part, we used unfolding transposed convolutional layers to prevent artifacts due to overlap. We used well-known 2D convolutional layers for the decoder part to extract the spatial feature. Both the encoder and decoder could preserve location information.

problems.

$$selu(x) = \begin{cases} x & x > 0, \\ ae^x - a & x \leq 0. \end{cases} \tag{2.10}$$

Then we also added an alpha-dropout layer against the overfitting problem while training the model. In addition, we made some changes to the DCNet framework. After several tests, we found unlapping subregions within the larger receptive fields has a faster convergence speed and best performance while training. Therefore, we set stride to 2 and kernel size to 2 in the encoder. We set stride to 2 and kernel size to 3 in the decoder.

2.2.3 Second General Layer: Classifier with Sub-network Nodes

We adopted our previous classifier [74] as our final classifier. Given $\mathbf{M}$ distinct feature samples combined from combination operator ($\mathbf{H}$,$\mathbf{t}$). $u(x) : \mathbf{R} \rightarrow (0, 1]$ is a normalized function; g is a sigmoid or sine activation function, and then for any continuous outputs t, we have $lim_{n\rightarrow+\infty}\|\mathbf{t} - ((g(\mathbf{a}_p^1, \mathbf{b}_p^1, \mathbf{H})) \cdot \boldsymbol{\beta}_p^1 + \cdots + (g(\mathbf{a}_p^c, \mathbf{b}_p^c, \mathbf{H})) \cdot \boldsymbol{\beta}_p^c)\| = 0$

Algorithm 1 The proposed method

```
Given a large training dataset $(\mathbf{x}_k, \mathbf{y}_k)_{k=1}^{M}$, $\mathbf{x}_k \in \mathbf{R}^n$, an invertible activation function
g , number of hidden nodes in each subnetwork node d (d equals number of targeted
dimensionality of the subspace features), regularization coefficient C, and the number
of subnetwork nodes L:
First general layer: Subspace feature extraction:
Step 1: Extract frequency features. Set $c = 1$, randomly generate the subnetwork
node for entrance feature layer by equation (3).
1: while c < L do
2:     Calculate the subnetwork node for exit feature layer by equation (4)
3:     Calculate the output error and error feedback data by equation (5)
4:     Update the subnetwork node $\hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c$ in the entrance layer by equation (6)
5:     Obtain the $c$-th subspace feature data by equation (7)
6:     Set $c = c + 1$, add a new subnetwork node $\hat{\mathbf{a}}_f^c, \hat{\mathbf{b}}_f^c$ in the feature mapping layer
       with orthogonal random initialization (equation (3)).
7:     Repeat $L - 1$ times, obtain the $L$ subspace features $\mathbf{H}_1$.
8: end while
Step 2: Train DCNet by EEG preprocessed data and extract spacial features $\mathbf{H}_2$.
Second general layer: Pattern learning:
Given combined feature $H$, set $c = 1, e_1 = t$.
1: while c ¡ L do
2:     Step 1: Calculate the $c$th subnetwork node $(a_p^c, b_p^c)$, and output weights $\beta_p^c$ as
       equation(2.11);
3:     Step 2: Calculate parameters as equation(2.12).
4: end while
Third general layer: Feature combination
Combined features $\hat{\mathbf{Y}}_{fre}$ and $\hat{\mathbf{Y}}_{spa}$ from equation(2.13)
```

holds with probability one if

$$\begin{aligned} \boldsymbol{a}_p^c &= g^{-1}((e_{c-1})) \cdot \mathbf{H}^T \left(\frac{C}{\mathbf{I}} + \mathbf{H}\mathbf{H}^T \right)^{-1}, \\ b_p^c &= \mathrm{sum}(\mathbf{a}_p^c \cdot \mathbf{H} - h^{-1}((\mathbf{e}_{c-1})))/N, b_p^c \in \mathbf{R}, \end{aligned} \tag{2.11}$$

$$\begin{aligned} & g^{-1}(\cdot) \begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = \sin(\cdot) \\ -\log\left(\frac{1}{(\cdot)} - 1\right) & \text{if } g(\cdot) = 1/(1 + e^{-(\cdot)}) \end{cases}, \\ & \mathbf{e}_c = \mathbf{t} - u_n^{-1} g(\mathbf{H}, \mathbf{a}_p^c, \mathbf{b}_p^c), \\ & \boldsymbol{\beta}_p^c = \frac{\langle \mathbf{e}_{c-1}, u^{-1}(g(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c)) \rangle}{\| u^{-1}(g(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c)) \|^2}, \end{aligned} \tag{2.12}$$

where $[\cdot]^{-1}$ represents its inverse function.

2.2.4 Third General Layer: Late fusion

In last layer, we extracted the predict feature from both frequency and spatial stream respectively.

$$\begin{aligned}\hat{\mathbf{Y}}_{fre} &= g(\mathbf{a}_p^1, \mathbf{b}_p^1, \mathbf{H}_{fre}) \cdot \boldsymbol{\beta}_p^1 + \cdots + g(\mathbf{a}_p^c, \mathbf{b}_p^c, \mathbf{H}_{fre}) \cdot \boldsymbol{\beta}_p^c, \\ \hat{\mathbf{Y}}_{spa} &= g(\mathbf{a}_p^1, \mathbf{b}_p^1, \mathbf{H}_{spa}) \cdot \boldsymbol{\beta}_p^1 + \cdots + g(\mathbf{a}_p^c, \mathbf{b}_p^c, \mathbf{H}_{spa}) \cdot \boldsymbol{\beta}_p^c.\end{aligned} \tag{2.13}$$

Then we fuse the $\hat{\mathbf{Y}}_{fre}$ and $\hat{\mathbf{Y}}_{spa}$ together and make the final prediction. Furthermore, different structures of autoencoder and DCNet can be used in our method. Other classifiers can be used, such as SVM or ELM. The proposed algorithm could be summarized in the following Algorithm 1.

2.2.5 Theoretical Analysis of DCNet Computational Complexity

VC-Dimension

Statistical learning theory established a relation to represent the lower bound of model generalization in terms of training error and testing error:

$$\epsilon_{true} = \epsilon_{emp} + \Phi(h, N), \tag{2.14}$$

where h is the VC-dimension of the model, N is the number of data samples and $\epsilon_{emp}, \epsilon_{true}$ are the empirical risk (training error) and true risk (true error) to some loss function respectively. Since true error is not predictable, minimize ϵ_{emp} and the function $\Phi(h, N)$ is the key for better performance. The function $\Phi(h, N)$ is referred as VC-confidence [66] and is defined as

$$\Phi(h, N) = \sqrt{\frac{h(ln(2N/h) + 1) - ln(\eta/4)}{N}}. \tag{2.15}$$

It represents the generalization of the model with probability $1 - \eta$. In other words, the difference between true and training errors would be more negligible as VC-confidence become smaller. As mentioned in [66, 74], this confidence is proportional to h/N. Goldberg and Jerrum [33] proved the upper bound of the neural network is $O(k^2)$ where $k = \theta(W)$ and W is the number of parameters. We calculated the upper bound of the number of parameters in the following sections to prove the generalization ability of DCNet.

Transposed convolutional part

For transposed convolution part of DCNet, we convert the 1D signal from $(1, 1, 310)$ to shape $(32, 32, 3)$. For each convolution or transposed convolution layer with (m, n) shape kernel(filter), d_p number of filters in the previous layer and d_c number of filters in the current layer, the number of trainable parameters can be represented as equation below:

$$((m \times n \times d_p) + 1) \times d_c. \tag{2.16}$$

The number of parameter DCNet transposed convolution part with layer L is

$$W_{trans} = \sum_{j=1}^{L} ((m \times n \times d_{j-1}) + 1) \times d_j, \tag{2.17}$$

where d_0 is the number of channel for the input signal. As we can see in Fig. 2.2, there are six layers in transposed convolution part as we want to reshape the input from $(2^0, 2^0, channel_{signal})$ to $(2^5, 2^5, 3)$. Since the number of filters $d_i \leq d_{i-1}$, the parameter number upper bound could be further expressed as

$$W_{trans} = O(L \times ((m \times n \times d_0) + 1) \times d_0), \tag{2.18}$$

where number of layer L, shape of the kernel (m, n) are constant, therefore

$$W_{trans} = O(d_0^2). \tag{2.19}$$

Convolutional part

In the convolutional part, the input shape is $(32, 32, 3)$ and the output is also in constant shape $(1, 1, 310)$.

$$\begin{aligned} W_{bn_i} &= d_i \times 4, \\ W_{conv} &= \sum_{j=1}^{L} ((m \times n \times d_{j-1}) + 1) \times d_j + d_j \times 4. \end{aligned} \tag{2.20}$$

Since the number of filters $d_{i-1} \leq d_i$, shape of the kernel (m, n) are constant and the number of filter in last layer $d_L = 310$, we can rewrite the upper bound to:

$$\begin{aligned} W_{conv} &= O(L \times ((m \times n \times d_L) + 1) \times d_L + L \times d_L \times 4), \\ W_{conv} &= O(L \times c + L \times c) = O(L). \end{aligned} \tag{2.21}$$

Generalization Analysis

Therefore, the upper bound of DCNet is $O(d + L)$ where d is the channel number of the input signal, and L is the number of layer in the convolutional part. Szymanski and McCane find a similar upper bound on their deep neutral network model [66]. Our original model proposed an asymmetric framework on the SEED and SEED-V datasets. Since there are only five layers in the convolutional part and the channel numbers are the same as the corresponded transposed convolutional layer, the parameters in the convolutional part are less than the transposed convolutional part. So our DCNet model has $O(d)$ parameters. As we mentioned in the VC dimension part, the VC-confidence is proportional to h/N where VC-dimension $h = O(W^2)$ only related to the number of trainable parameters, and N is the number of samples in the dataset. Since trainable parameters' upper bound in our model is only dependent on the channel of input, the generalization of our model is promising as long as the number of samples is much larger than the number of channels.

On the other hand, we still need a larger model with more parameters to fit a large dataset better to reduce ϵ_{emp}. For a larger dataset such as the DREAMER dataset, we prefer to increase the number of layers in the convolutional part to guarantee training accuracy while maintaining generalization performance.

2.2.6 Dataset

2.3 Experiments

Previous studies have shown that raising human emotions through movie clips is reliable [15] [86]. In this paper, we adopted the SEED [87], SEED-V [41] and DREAMER [31] dataset, one of the largest datasets, which has been popularly used for EEG-based emotion recognition. Video clips with audio were used to elicit specific emotions of the subjects' emotions. The SEED dataset experiment could prove the overall performance of our framework. In contrast, the SEED-V dataset has fewer

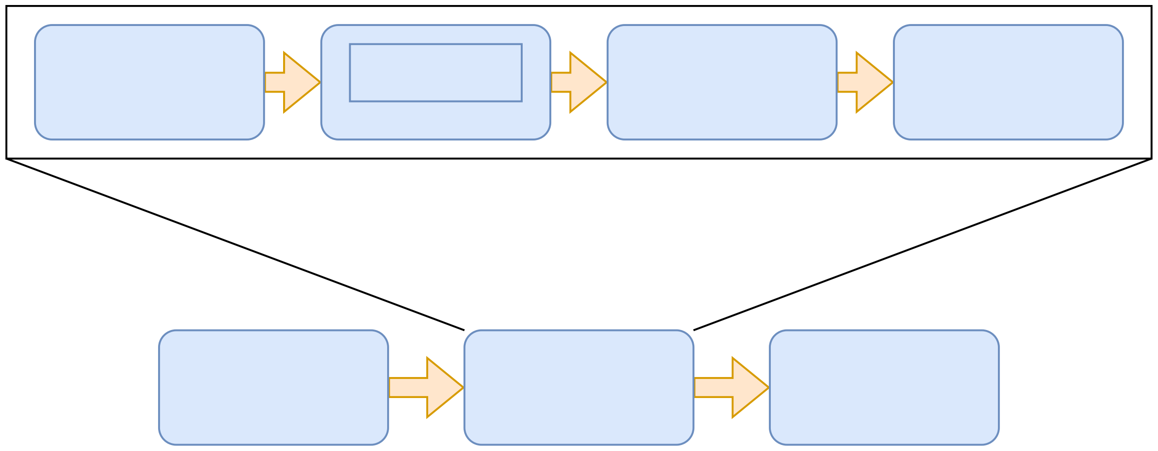


Figure 2.3: SEED Experiment protocol.

samples and two more emotion classes. The SEED-V dataset experiment was proposed to demonstrate the generalization of our framework. DREAMER dataset has relatively imbalanced data in each category and a different number of channels. We proposed a DREAMER dataset experiment to show the stability and robustness of our spatial feature extraction method.

SEED and SEED-V Dataset

Each film clip is about four minutes long, and carefully selected important clips enable it to create coherent emotions, which can well trigger the corresponding human emotions. There are 15 clips in both SEED dataset and SEED-V dataset. Each clip has 5 to 15 seconds of prompts before and 15 to 30 seconds of rest after each clip [see Fig. 2.3]. All movie clips are sorted according to different emotions to ensure that the same emotional movie clips are displayed discontinuously. To test the stability of EEG signals for sentiment analysis over time and the performance of cross-session emotion recognition, all experimental participants were required to conduct 3 trials, each trial being more than 3 days. To ensure the accuracy of emotional records, each subject was asked to complete an Eysenck Personality Questionnaire (EPQ) before the start of each trial. Extroverts who proved stable were selected as subjects. Therefore, subjects who reported themselves as normal participated in the experiment. According to the international 10-20 system, the EEG NeuroScan system was used to record EEG signals at a sampling rate of 1000 Hz. There are 62 active AgCl electrode channels for recording EEG signals. The impedance of each channel in the cap was controlled to less than 5 KΩ.

SEED dataset contains three different classes (positive, neutral and negative). SEED-V included five emotion classes (happy, sad, fear, disgust, and neutral). SEED-V also contains eye movement signals. However, since our model only focuses on EEG data, eye movement signals are not considered in this experiment. For the SEED Dataset, we used the first 9 trails for each subject in each session as a training dataset (84420 samples) and all the last 6 trails as a testing dataset (58128 samples). For the SEED-V Dataset, we proposed a 3-fold cross-validation method to split the training and testing dataset. Each fold is guaranteed to have five different trails representing five different emotions in random order. Since the trails' lengths are different, the number of training samples (19184; 18492; 20656) and testing samples (9984; 10672; 8512) in each fold are also different. From the above information, the SEED-V Dataset is a smaller dataset containing more emotion categories.

DREAMER Dataset

DREAMER dataset has 18 film clips. The length of these film clips was between 65 to 393s. There are 23 subjects in this dataset, 14 male and 9 female. Unlike the SEED dataset, DREAMER dataset EEG contains 14 channels at a sampling rate of 128 Hz. DREAMER dataset also contains ECG signals, but we only use EEG data in this experiment. DREAMER dataset is a multimodal dataset. The rating scales for the DREAMER dataset are valence, arousal and dominance. Twenty-three participants watched these film clips, and each participant conducted 477,184 data samples. After preprocessing, the data shape is $23 \times 477184 \times 14$ with 14 channels. The corresponding label shape is $23 \times 477184 \times 3$ with three different labels. We use 10-fold cross-validation to separate the training and validation data similar to Cheng et al. [13] dataset division method. The rating range for each class is 1 to 5. As the experiments from the baseline method [31], we use DREAMER dataset as three binary classification schemes. We set a threshold in the middle of rating scales and changed the rating value from 5 point scale to two classes(low and high). We consider rating value smaller or equal to 3 to low and greater than 3 to high.

2.3.1 Experiment Setup

We set up three different experiments for different datasets to test the robustness of our proposed method. In each experiment, we systematically compared the performance of our method and the most recent methods. All the experiments are done in

Python and the framework of DCNet is implemented using Tensorflow.

2.3.2 Experiment for SEED Dataset

Performance Comparison

Table 2.1: Network Configuration

Methods	parameter details
SVM	Linear Kernel, search space $2^{[-10,10]}$ with a step of one.
KNN	Baseline k equals 5.
ELM	1000 hidden neurons, search space $2^{[-10,10]}$ with a step of one.
H-ELM	N1=N2=300, N3=1000, search space for C1 and C2 is $2^{[-10,10]}$ with a step of one.
GELM	the number of hidden layer neurons is fixed as 10 times of the dimensions of input and we adopt a 5-fold cross-validation scheme.
DBN	first and second layer of DBN is selected from the ranges of [200, 200] and [150, 500], respectively.
HNSN	search space for C1 and C2 is $2^{[-10,10]}$ with a step of one. Three subnetwork nodes.In each subnetwork node, 500 hidden nodes are used.
OURS	DCNet part: we use 30 epochs, learning rate of 0.01. HNSN part: search space for C1 and C2 is $2^{[-10,10]}$ with a step of one. Three subnetwork nodes.In each subnetwork node, 500 hidden nodes are used.

For SEED dataset, we compared the performance of 13 different methods for emotion recognition: 1) Support Vector Machine (SVM); 2) Extreme Learning Machine (ELM); 3) Logistic Regression (LR); 4) Deep Belief Networks (DBNs) [84]; 5) Adaptive Subspace Feature Matching (ASFM) [8]; 6) Dynamical Convolutional Neural Networks (DGCNN) [65]; 7) Bimodal Deep AutoEncoder (BDAE) [47]; 8) Bi-hemispheres Domain Adversarial Neural Network (BiDANN) [42]; 9) Graph Regularized Extreme Learning Machine (GELM) [85]; 10) Hierarchical Network with Subnetwork Nodes (HNSN) [76]; 11) Electrode-frequency Distribution Maps (EFDMs) [70]; 12) Channel-fused Dense Convolutional Network (CDCN) [16]; 13) the proposed method.

These methods use DE feature as input. For SVM and ELM, parameters are set up from space [2^{-10}, 2^{-9},$\cdots$, 2^{10}] in each experiment. For GELM, we set the number of hidden layer neurons to 10 times the input data dimension, and adopted the cross-validation scheme [85]. We used 300 hidden neurons in the first and second layers and 1000 hidden neurons in the third layer for H-ELM($N1 = N2 = 300$, $N3 = 1000$). In

DBN, two hidden layers are used. We trained 1000 epochs with batch size of 201. We set unsupervised and supervised learning rates of 0.1 and 0.5. In each experiment, the number of neurons in the first layer of the DBN network was selected from [200, 500], and the number of neurons in the second layer was selected from the range of [150, 500]. For HNSN, to compare fairly with ELM/SVM, we selected the same values of C1 and C2, which all choose regularization parameters from space $[2^{-10}, 2^{-9}, \cdots, 2^{10}]$. For our method, in the DCNet part, we use 30 epochs with a learning rate of 0.01. For the HNSN part, in order to be consistent with other methods mentioned above, we choose the same parameters C1 and C2. Our proposed method includes late fusion. Table 2.1 shows the specific settings for different methods.

We used subject dependent test to evaluate our proposed method. Subject dependence means using a person's emotional responses, stimulated from different film clips, to predict this person's emotions. Table 2.2 shows the performance of each stream and our method. For emotion recognition using the DE feature, we obtain an average accuracy of 94.84%, which is almost 3% higher than [76] method.

From the confusion matrix in Fig. 2.4 we can obtain the following observations:

- Our method performs great on neutral and positive emotions. The accuracy almost achieves 99% on these two classes.
- From the related research, the negative class on the SEED dataset is likely to be more confused than other classes. Our proposed method ensured positive and neutral emotions recognizing accuracy and simultaneously boosted the negative class's accuracy.

We compared the accuracy of our proposed methods with other state-of-the-art methods to show the advantages of our proposed method. Table 2.2 shows the results of comparative experiments. From the table, we can see that the accuracy of our proposed method is significantly higher than that of other recent methods. In addition, our results are consistent with previous studies [84] [15] that our proposed method with DE features has the best performance in EEG-based emotional recognition.

DCNet Feature Analyze

We used the t-SNE algorithm the analyze the feature distribution and compared it to the original DE feature as Fig. 2.5. We used three different colours to represent different emotion classes. From the original DE feature, data are closer to their

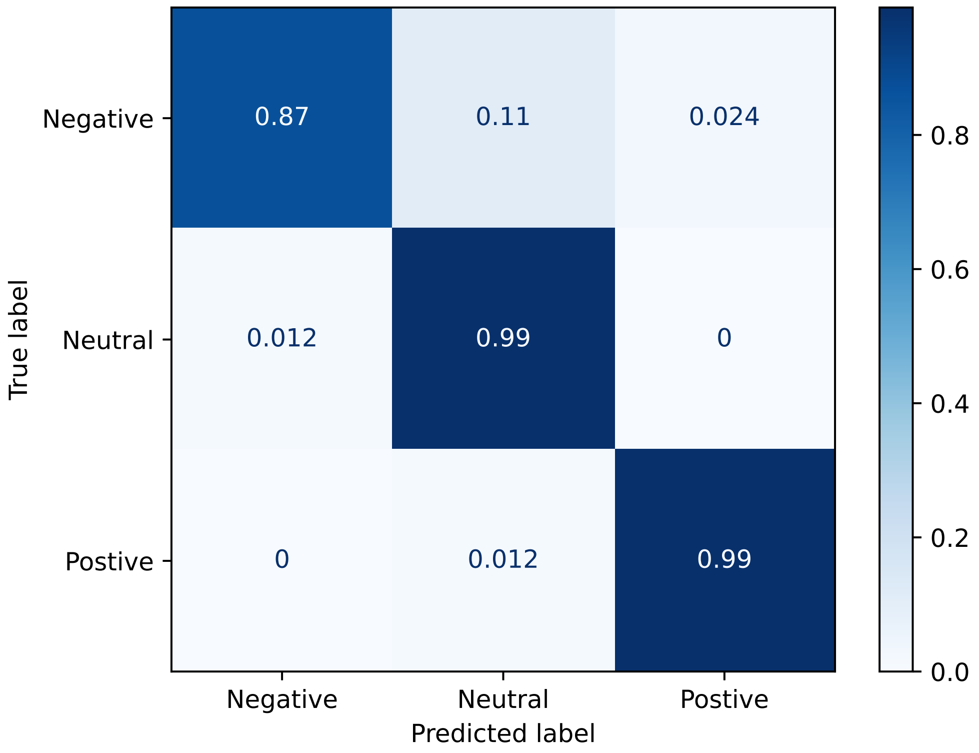


Figure 2.4: The confusion matrix of our proposed method on SEED dataset.

centroid, and data from three different classes are clustered together. Most data are lapping together rather than constructing distinct sequences. In contrast, the distribution of DCNet spatial features is more evenly. It also gradually separates different emotions and clusters the data by sequences of the same emotion. The comparison shows that the performance of our spatial feature extraction method is significant.

2.3.3 Experiment for SEED-V dataset

In order to show our framework generalization performance and test how our network performs on datasets with more emotional categories, we adopt the SEED-V dataset including five emotion statuses, disgust, fear, sad, neutral and happy. In the experiment, carefully selected film clips are used as the stimuli, which have been explored to have reliability in eliciting emotions [60].

Sixteen healthy subjects (6 males and ten females) participated in the experiments,

Table 2.2: Mean Accuracy of Subject Dependent Cross Session Test on SEED dataset

Methods	Mean	Std
SVM	89.99	-
ELM	82.92	-
LR	82.70	-
DBNs [84]	86.08	-
ASFM [8]	83.51	-
DGCNN [65]	90.40	8.5
BDAE [47]	91.01	8.9
BiDANN [42]	83.28	-
GELM [85]	91.06	-
HNSN [76]	91.67	9.91
EFDMs [70]	90.59	-
CDCN [16]	90.63	-
DCCA [46]	94.6	9.2
SGMC [30]	94.04	-
PR-PL [88]	93.18	6.55
JDA [46]	88.28	11.44
FOIT [39]	87.36	12.87
OURS	**94.84**	9.38

and each subject was required to perform the experiments for three sessions at an interval of one week or longer. EEG signal is collected simultaneously when the subjects are watching the film clips.

We use the cross-validation method to generate three train/test splits. We first compared each stream features approach accuracy [see Table 2.3]. As we can see, fusing multi-stream features achieves the best performance.

Table 2.3: Cross Session Mean Accuracy of Each Stream and Two-stream Fusion Approaches

Methods	DE
Frequency Stream	64.31
Spatial Stream	64.79
Two-Stream	**68.61**

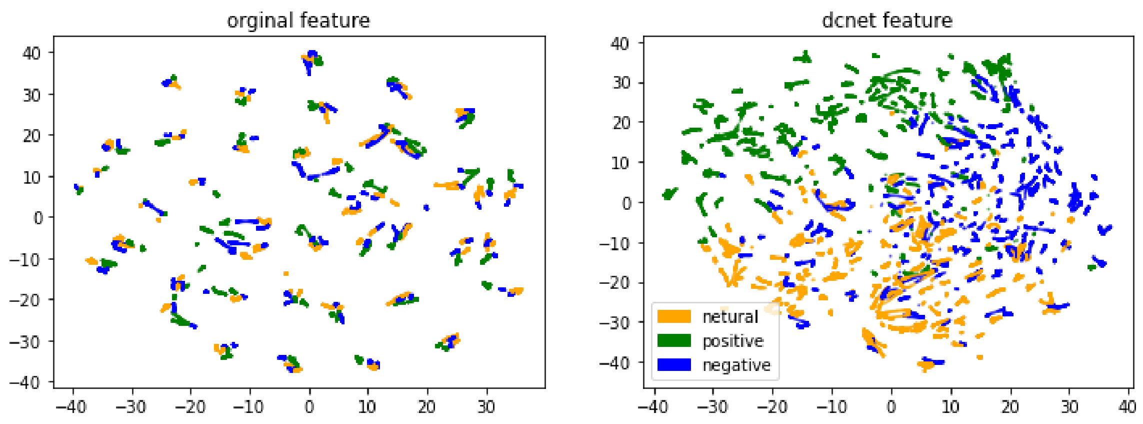


Figure 2.5: Feature visualization by the t-SNE algorithm. Different colors represent different emotion classes.

Performance Comparison

To further review our result, we depict the confusion matrices for each fold and the overall result of our proposed method. Fig. 2.6 shows the confusion matrices of five emotions on SEED-V dataset.

- From the overall confusion matrix, our method can ideally recognize fear and happy classes. The average accuracy is more than 75%.
- The sad and neutral classes get 68% and 65%, after by fear and happy tasks. It is worth noting that both these two classes have 15% misclassified samples with each other. This explains the sad and neutral emotions are easier to confuse on our method.
- Recognizing the disgust emotion was the most challenging task in the previous study [41]. We can observe that the disgust emotion performs very stable in all three folds, very easy to confuse with other emotions, especially with other negative emotions (fear and sad). Maybe disgust emotion stimulus cause resonance in participants and evoke other negative emotions.

Table 2.4 subject-dependent test compares the accuracy of our proposed methods with SVM, ELM, BDAE and HNSN

DCNet Feature Analyze

To see the benefits of deep features extracted by our proposed method, we run an at-distributed stochastic neighbouring embedding algorithm (t-SNE) to find two-

Table 2.4: Mean Accuracy of Subject Dependent Test on SEED-V dataset

Methods	Mean	Std
ELM	33.69	-
SVM	57.52	-
BDAE [41]	69.50	-
HNSN [76]	64.31	13.52
OURS	**68.61**	13.98

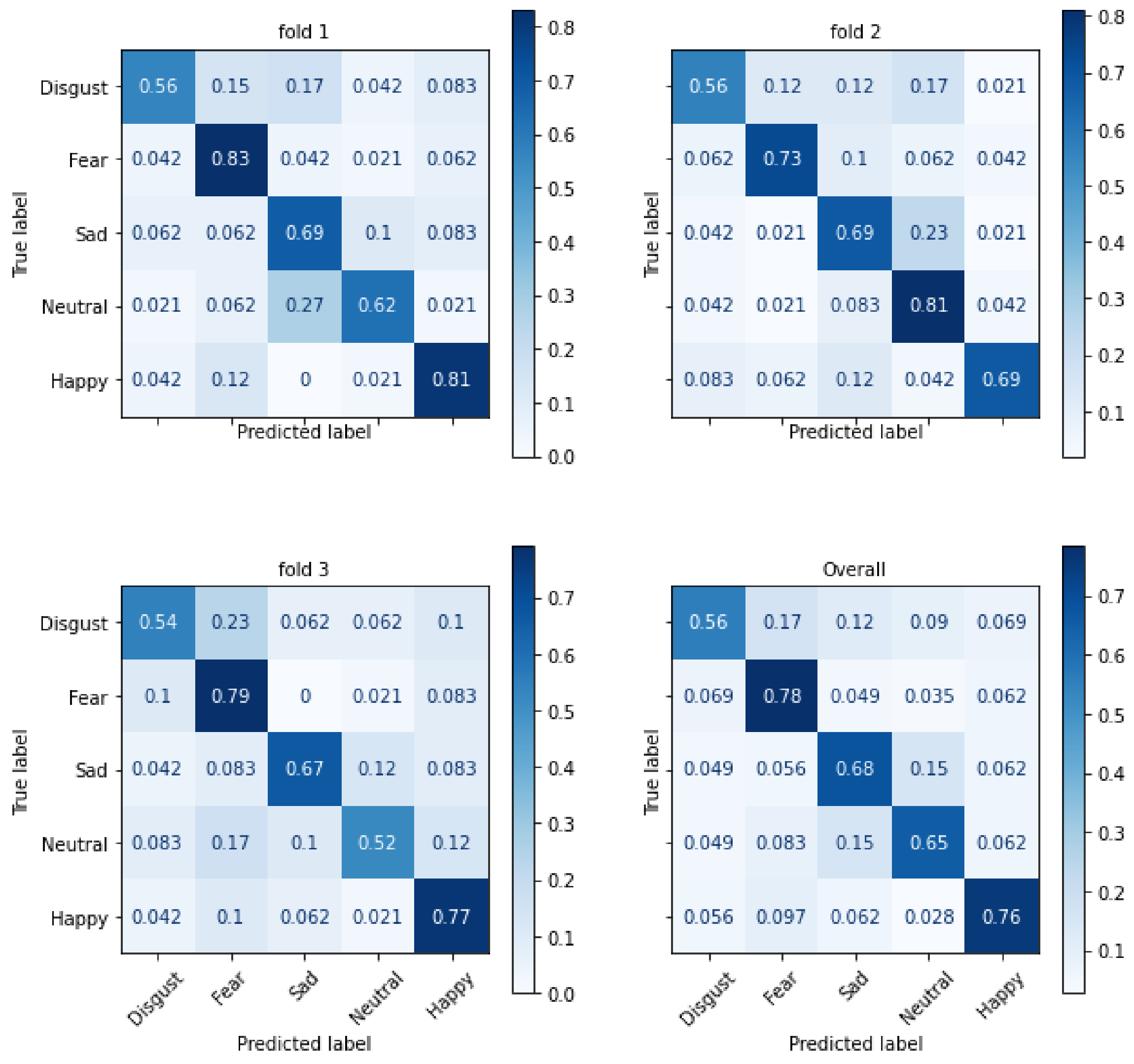


Figure 2.6: The confusion matrix of our proposed method on SEED-V dataset.

dimensional embeds of high-dimensional feature space and plot them as coloured points, according to semantic categories in a particular hierarchy. Fig. 2.7 shows the visualization of the DE original feature and the DCNet feature in three different folds. As we can see from Fig. 2.7, the disparity in different emotions classes is not obvious. The connections between the same class emotion are fragile, especially in this five-emotion classification task. After we extracted the spatial feature from DCNet, it is clearer to see different emotions cluster the samples than the original feature before we processed. Furthermore, the same class data samples clustered together with more robust connections. These differences are sufficient to demonstrate the superiority of the DCNet feature.

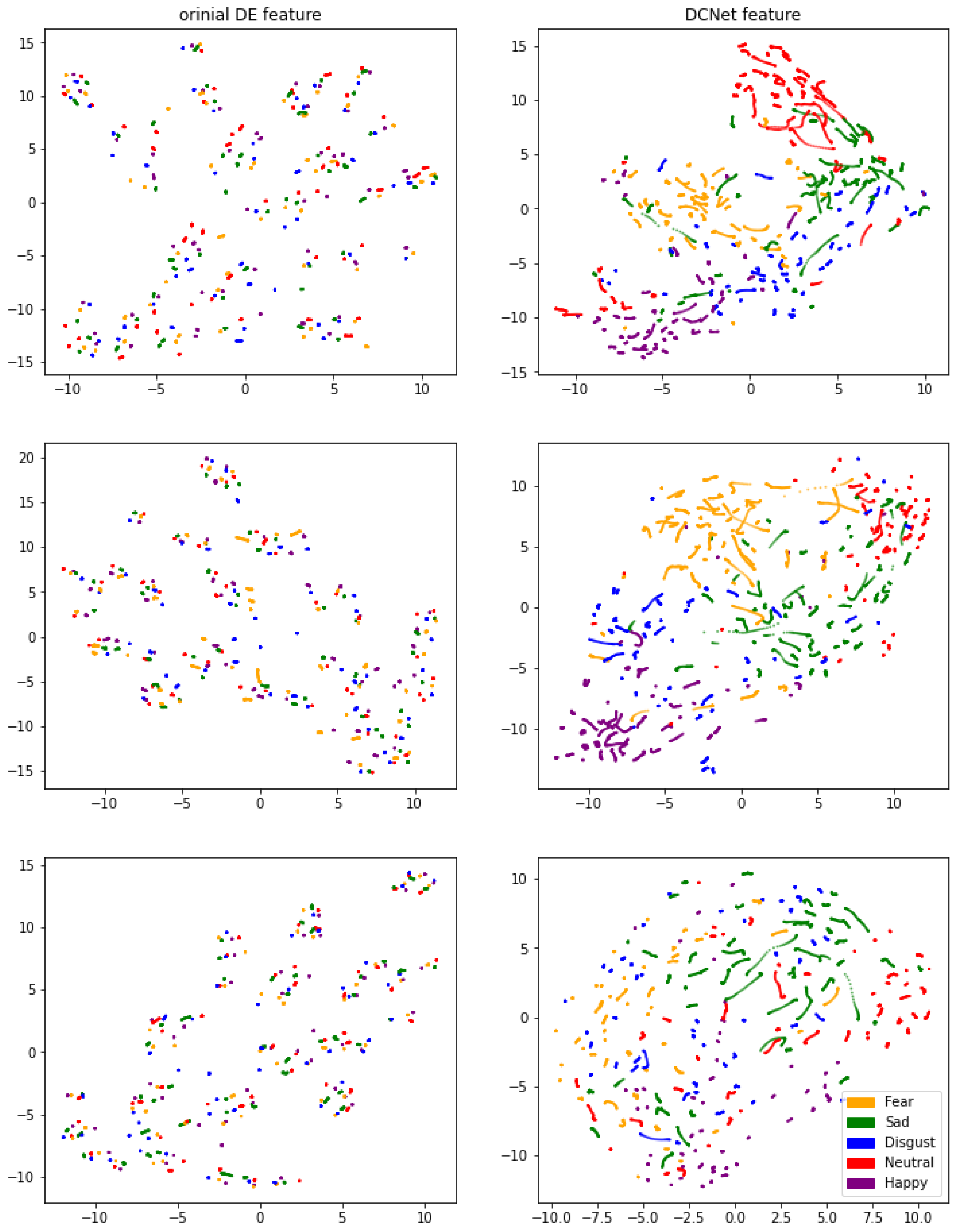


Figure 2.7: Feature visualization by the t-SNE algorithm. Different colours represent different emotion classes. Three columns represent different folds. The first row is the DE feature and the second row is the DCNet feature.

Table 2.5: DREAMER dataset Cross Session Test Mean Accuracy

Methods	Valence	Arousal	Dominance	Total Average
DT	75.53	75.74	76.40	75.89
SVM	87.14	87.03	87.18	87.12
VGG-16 [64]	78.99	79.23	54.83	71.02
DGCNN [65]	89.59	88.93	88.63	85.26
Deep Forest [13]	89.03	90.41	89.89	89.78
BDAE [47]	88.57	86.64	89.52	88.24
Deep CCA [46]	90.57	88.99	90.67	90.08
3DFR-DFCN [38]	**93.15**	91.30	**92.04**	92.16
Our proposed	93.01	**92.04**	91.74	**92.26**

2.3.4 Experiment for DREAMER dataset

DREAMER dataset is a larger multimodal dataset. Since the DREAMER dataset has three different labels (valence, arousal, dominance), it is a suitable dataset to test the generalization of our proposed method. Compared to 5×62 channels in SEED and SEED-V datasets, the DREAMER dataset only has 1×14 channels. Unlike the previous experiments, the DREAMER dataset has imbalanced data samples in each category. To further discuss the contribution and robustness of DCNet on spatial feature extraction, we keep the encoding structure precisely the same as the previous two experiments. On the decoder part, we proposed a DCResNet framework that combined DCNet with ResNet50 [24] using ImageNet pre-trained weight.

Due to the characteristic of our spatial feature extractor, we can connect any image classifier model after the mid-layer of DCNet (shape of $32 \times 32 \times 3$). This experiment explains the performance of DCNet without the symmetrical model structure.

For this time, we combine pre-trained ResNet50 on our DCNet as DCResNet. Since the output of the last transposed convolution layer in DCNet is $32 \times 32 \times 3$ as Fig. 2.2, we directly connect it as the input of the ResNet50. This experiment was completed on 4 RTX6000 GPUs in python, and the model was implemented using Tensorflow. The DCResNet used Adam as the optimizer and trained in 100 epochs and 4096 batch sizes.

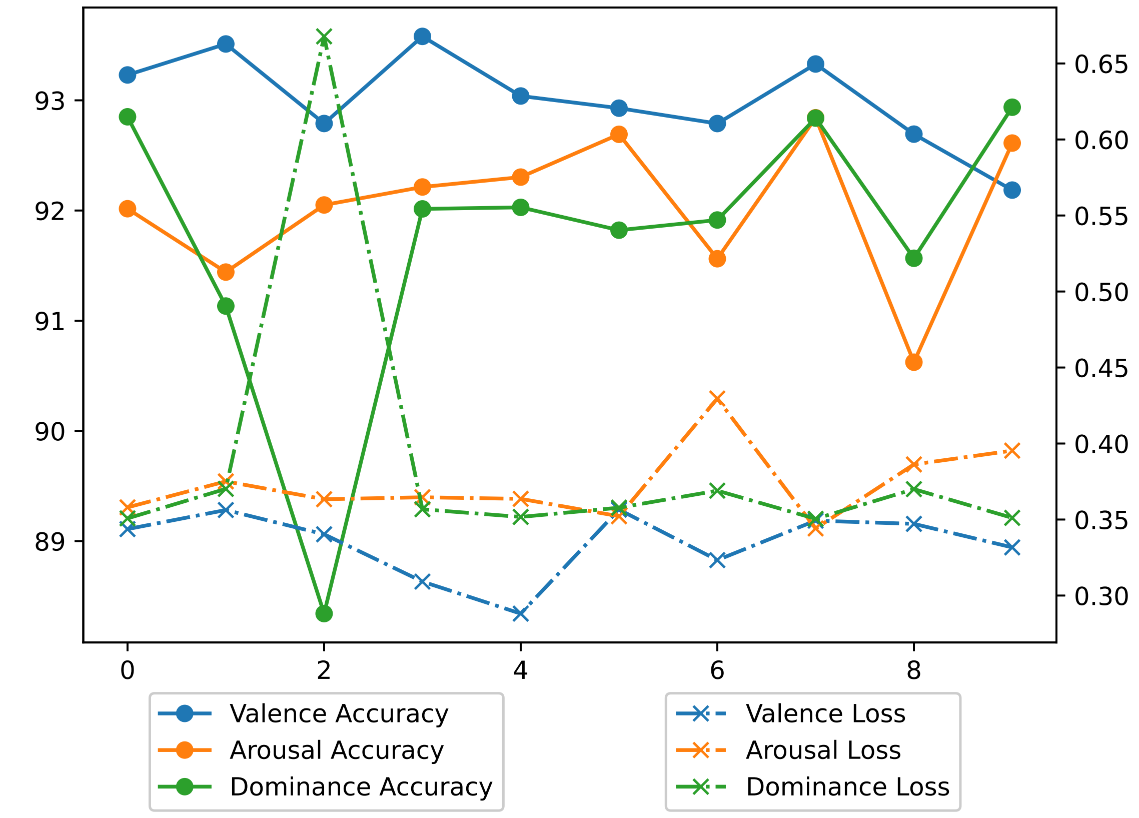


Figure 2.8: Ten-fold cross-validation accuracy and loss on the DREAMER dataset. **X-axis:** the number of fold; **y-axis(left):** validation accuracy; **y-asix(right):** validation loss.

Subject dependent Experiment Results

After we perform the subject-dependent experiment, the result sufficiently proves the generalization, robustness and flexibility. From Fig. 2.8, most validation set has above 90% accuracy on all three labels. We also compared five recent related works and two baseline methods. We used Decision Tree Classifier (DT) and Support Vector Machine classifier (SVM) as baseline classifiers. We compared our proposed method result with VGG-16 [64], DGCNN [65], Deep Forest [13], Deep CCA [46] and 3DFR-DFCN [38].

From Table 2.5, the result shows we have comparable accuracy with one of the best performance state-of-the-art methods, 3DFR-DFCN [38]. It is worth noting that the advantage of DCNet is less human manual interaction and extracting of spatial features efficiently without additional information such as channel location.

As a comparison, although 3DFR-DFCN [38] has a similar performance, they manually converted the one-dimensional EEG signal into the two-dimensional matrix to preserve channel location information. Moreover, the total average accuracy of our proposed method is the best. The best overall accuracy proves the robustness of the DCNet structure in the multimodal dataset experiment.

Flexibility Analyze

We mentioned earlier that the DCNet structure could be compatible with different image classification models in terms of flexibility. The advantages of deep image classification models are not only the robustness of architecture but also the pre-trained weights of these models. To further discuss the usefulness of the 'flexibility,' we compared the performance of DCResNet with transfer learning from Imagenet weights and DCResNet with learning from scratch. From Fig. 2.9, the superiority of the transfer learning method is clear at a glance. The overall accuracy of the transfer learning method with the same training and testing set is 3% higher than learning from scratch. Furthermore, there is no apparent relation between one-dimension EEG signal data and the two-dimension Imagenet dataset. So we can conclude that various types of the one-dimension dataset can use this transfer learning framework to boost accuracy and efficiency further.

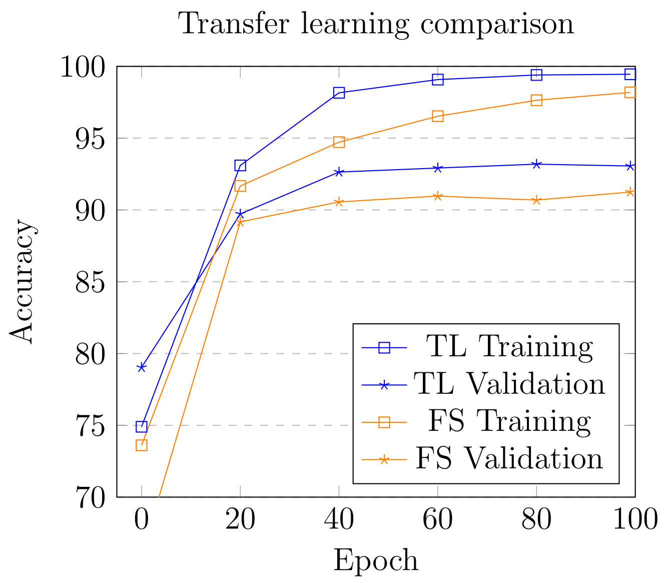


Figure 2.9: Comparison of the DREAMER dataset accuracy in each epoch between transfer learning from Imagenet (TF) and learning from scratch (FS).

2.4 Discussion

In this paper, we assumed that DCNet could extract spatial features. We provided feature visualization by the t-SNE algorithm in the experiment to prove our assumption. Zhang et al. introduced that combining or fusing spatial domain features with frequency domain features is more effective [82]. Our cross-session experiments on different datasets also evidence this argument. Compared to the original HNSN method, our combined framework boosted 3.17% and 7.11% on the SEED and SEED-V datasets, respectively.

As mentioned in the motivation section, most previous studies manually extract spatial features based on the specific EEG dataset channel orders and preprocessing method. However, different datasets have various sampling methods and could benefit from different preprocessing techniques. Ensuring the quality of extracted spatial domain features across different datasets has always been a hot topic. We aimed to propose an adaptive spatial feature extraction method to solve this issue. Using deconvolution(transposed convolution) as an encoder could reserve local information. The iterative training pattern automatically adapted the encoder to different preprocess methods and datasets. We experimented with the same deconvolution encoder structure across different preprocessing methods (downsampling and DE feature on the SEED and SEED-V dataset, removing the mean and standardized baseline on the DREAMER dataset) and a large span on the number of channels (62*5 on the SEED and SEED-V datasets, 14*1 on the DREAMER dataset). Compared to the recent studies, the performance confirmed our hypothesis and reflected the robustness of our framework in different application scenarios. As a trade-off for fewer human-interaction and higher compatibility, the complexity of DCNet is higher than the traditional spatial domain feature extraction methods due to more parameters and iterative training. The DCNet structure converting each EEG signal sample into a more extensive two-dimension latent space also needs more GPU memory. For example, the DCNet structure we used to train the SEED dataset contains 1,070,435 trainable parameters. Training the SEED dataset on Colab with Tesla P100 takes about 1 second per epoch(20ms/step). In our experiment, the time of training the model for 30 epochs is acceptable.

Based on the transfer learning in the last experiment, we unexpectedly found that transfer learning from ImageNet could converge faster and boost performance significantly in limited epochs, which gives us more flexibility in application. Here we briefly discuss the benefit of transfer learning in EEG datasets. Due to the limited number of subjects in EEG datasets, the model directly trained EEG data could contain high variance. Transfer learning from the image dataset may improve the generalization and reduce the variance.

2.5 Conclusion

Overall, we demonstrate 1) decomposing and transforming one-dimensional EEG signals into spatial and frequency features as more effective for emotion recognition, 2) the promising performance of our framework on small-scaled, medium-scaled and large-scaled signal datasets, and 3) the generalization and flexibility of DCNet compare to other feature extraction methods on extracting 1D signal data. The next chapter presents a contrasting framework that boosts representation learning efficiency.

Chapter 3

Bidirectional self-supervised learning representation

3.1 Introduction

3.1.1 background

The supervised learning method has made promising progress in recent years [24, 35, 36, 69]. However, the promising progress of supervised learning cannot leave high-quality data samples, human-labelled data labels, more parameters (more profound and wider) in the model, bigger batch size or more training epochs to train the model. Nowadays, many studies have started to pay attention to model performance efficiency, such as providing carbon footprint as an indispensable part of model performance [80] or evaluating performance with smaller batch size [12]. Compared to unsupervised learning and self-supervised learning(SSL), adequate resources are the bottleneck for supervised learning. As we expect models to have better robustness and generalization performance, one direction is not heavily dependent on human-labelled labels. Self-supervised learning has attracted attention lately for its generalization ability and efficiency in data utilization. Compared to unsupervised learning, the SSL framework learns representation from "labels" created by the content of the training samples. Based on this specific framework, SSL could be more appropriate for training low-quality data, less labelled or hard-to-label data or even unbalanced data.

In supervised learning, it is hard to tell which part of the neural network corresponds to learning the task's representation and to what extent the label could influence the training process rather than the semantic information of training samples. Unlike supervised learning, self-supervised learning further subdivided the training steps into upstream and downstream tasks. The upstream task is used to train the model to learn semantic information from data samples unsupervised, and the downstream finetunes the model with provided labels. The upstream task enables learning from data representations, and people can directly evaluate the performance, which gives us an efficient pipeline to learn the relationship between representation learning efficiency and different self-supervised learning framework. We will introduce these evaluation methods in the related work section. Self-supervised learning also needs fewer epochs of training in downstream tasks. Some frameworks also have outstanding performance with smaller batch size [12] [11]. In this study, we are focusing on the contrastive learning framework and contrastive loss function.

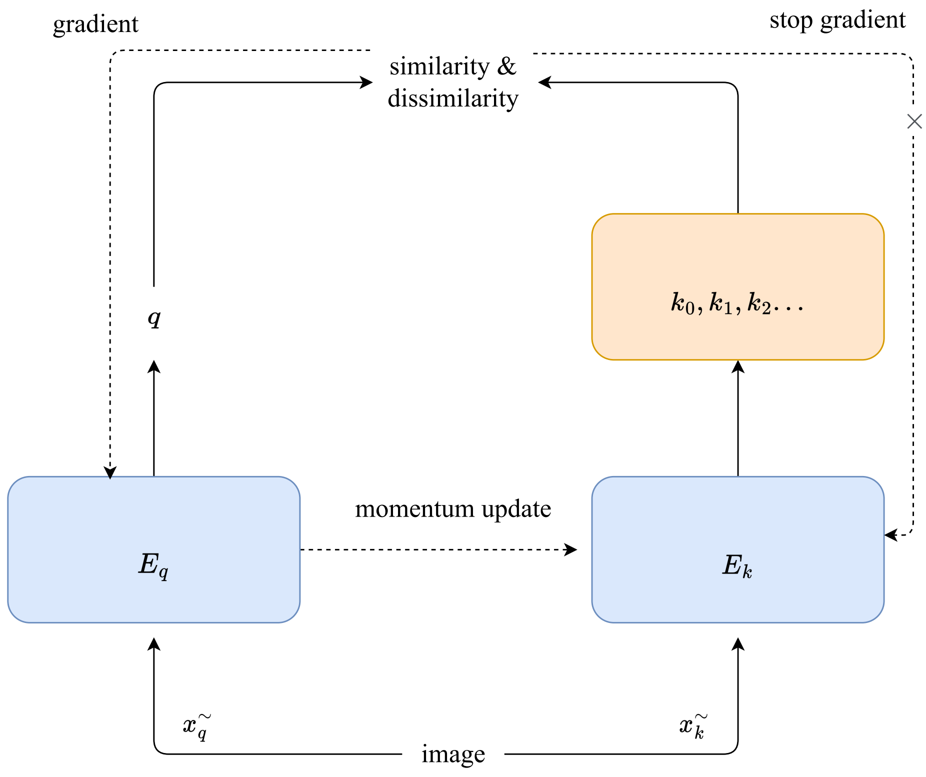


Figure 3.1: MoCo Framework. $x_q^{\sim}$ and $x_k^{\sim}$ are two different augmentations of the same image. The memory bank saves several mini-batch representations that output from the momentum encoder.

3.1.2 Related works

Nowadays, many self-supervised learning methods propose frameworks with contrastive learning. At first, people proposed an "end-to-end" framework to contrast two similar structures with two different inputs and compute their contrastive loss. At this stage, a contrastive learning framework needs a large batch size to keep enough positive and negative samples to obtain enough contrast information. In other words, the "end-to-end" framework requires large GPU memory as computation resources. Then the memory bank was proposed by [73] to reduce batch size and save each batch's output feature into a memory bank. The memory bank feature can be used as additional data samples to feed in contrastive loss function with current batch features. However, there is a disadvantage to this framework. The features stored in the memory bank are encoded during training, and each feature vector is generated with different parameters. Thus, the features in the memory bank lack consistency.

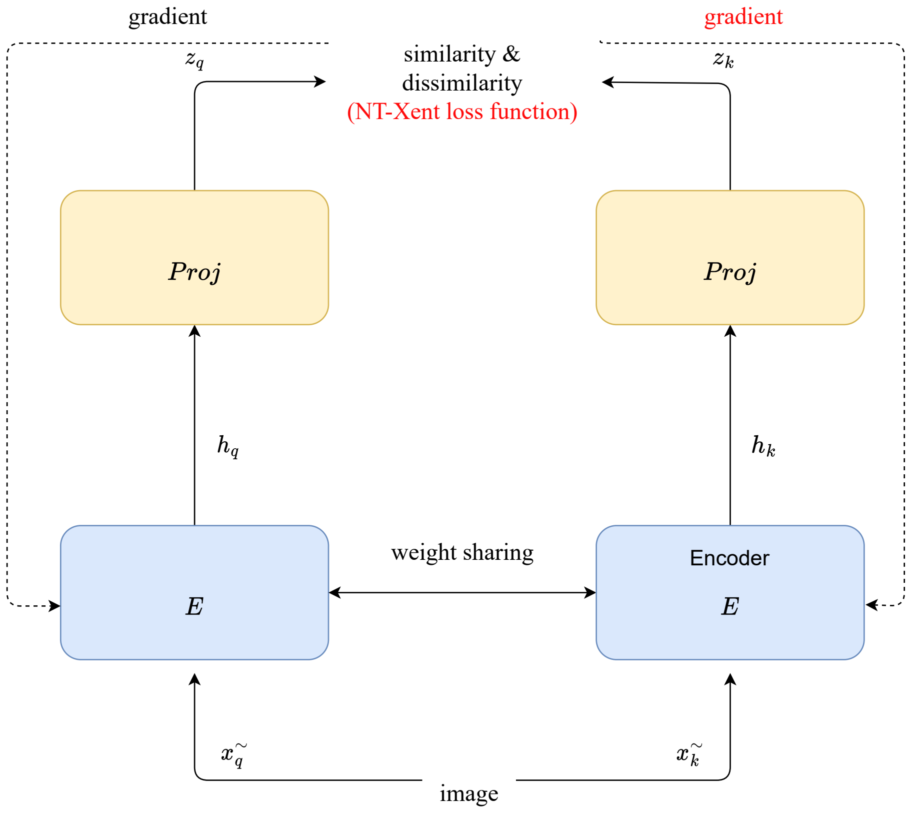


Figure 3.2: SimCLR Framework

MoCo

MoCo [23] applied unsupervised representation learning in computer vision and outperformed many datasets on supervised pretraining performance. Compared to the memory bank [73], MoCo proposed a momentum encoder to avoid feature inconsistency. The MoCo framework contains two encoders, a typical encoder and a momentum encoder. The loss function contrasts the similarity and dissimilarity between the output feature of both encoders (as Fig. 3.1). The typical encoder updates the parameters from the loss function gradient. To keep the consistency of momentum encoder E_k, E_k only updates parameters from the typical encoder E_q with a momentum as equation below:

$$\theta_q \leftarrow m\theta_k + (1 - m)\theta_q.$$

Moco saved the encoded keys from momentum encoder as queue with n batch. Then the keys could be represented as large amount negative samples to further boost the contrast loss function efficiency. Algorithm. 2 explains how MoCo works.

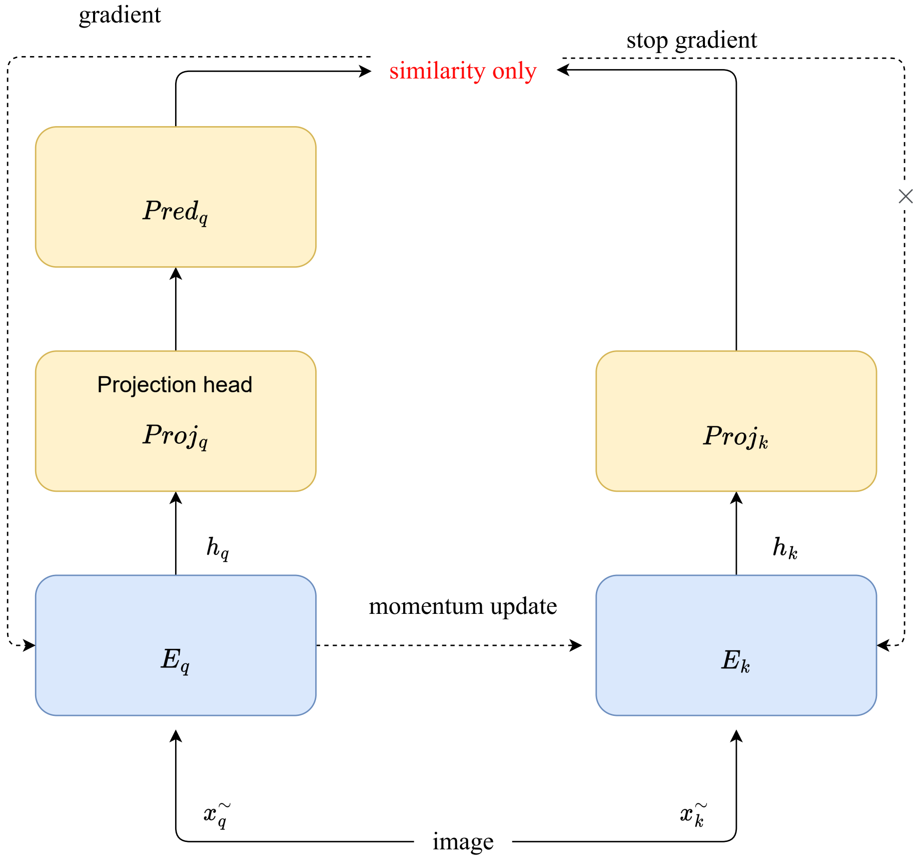


Figure 3.3: BYOL Framework

SimCLR

SimCLR [10] proposed a different type of framework as Fig. 3.2. Compared to MoCo, there is only one encoder in the framework. The encoder updates parameters by contrast loss function gradient, so there is no lack of consistency problem. Compared to the end-to-end framework, SimCLR adds a non-linear projection head after the encoder. The authors found that adding a non-linear projection head works better than a linear projection head or directly putting the encoded feature vectors into the contrastive loss function. The model needs to shorten the distance between two same images in the latent space with different augmentations. However, the encoder should not learn this information because it conflicts with learning image representation. Therefore Chen et al. use a non-linear projection head to "remove" the differences between different augmentations. The authors also proved that the non-linear projection head could improve the representation quality [10].

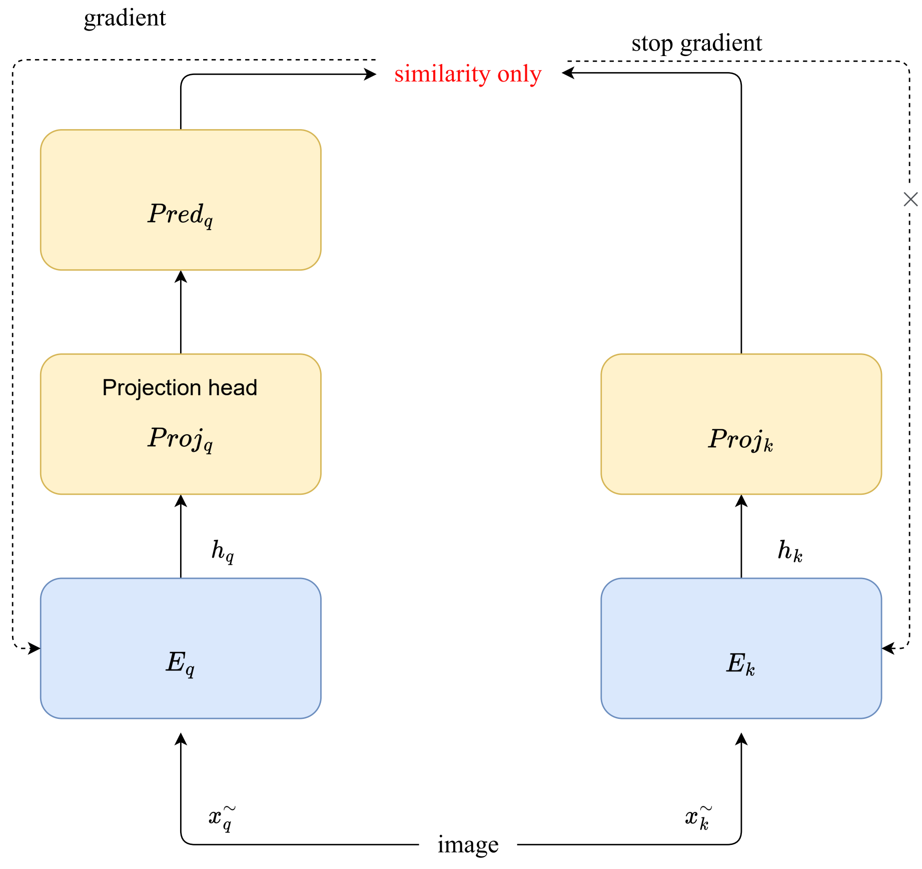


Figure 3.4: SimSiam Framework

The author proposed the NT-Xent loss as the contrastive loss. In a mini-batch with size n, the purpose of the NT-Xent loss function is to maximize the agreement of one of the input $\tilde{x_{0q}}$ with the only positive sample $\tilde{x_{0k}}$ and minimum the similarity from all other samples in the same batch $\tilde{x_{1k}}...\tilde{x_{nk}}$ as Fig. 3.5.

Although SimCLR and MoCo have different frameworks, they both use negative samples to prevent collapse in latent space. The following studies explore more possibilities to avoid collapse.

BYOL

BYOL [21] is different from previous methods. It does not require negative samples to keep enough distance in latent space. Similar to MoCo [23], BYOL has two networks (because BYOL networks contain a projection head and prediction head, we cannot call them "encoder") and one network update parameter with momentum. Based on

Algorithm 2 MoCo

As a new batch data x_n input to the framework

```
 1: while pretrain do
 2:     $\tilde{x_{ni}} \leftarrow augmentation(x_n)$
 3:     $\tilde{x_{nj}} \leftarrow augmentation'(x_n)$
 4:     $q \leftarrow E_q(\tilde{x_{ni}})$
 5:     $k_n \leftarrow E_k(\tilde{x_{nj}})$
 6:     remove the oldest batch $k_0$ from the queue $k$
 7:     add $k_n$ at the end of queue $k$
 8:     calculate contrastive loss of $q$ and $k$
 9:     update parameters of encoder $E_q$ by the gradient of loss function
10:     update parameters of encoder $E_k$ by above equation
11: end while
```

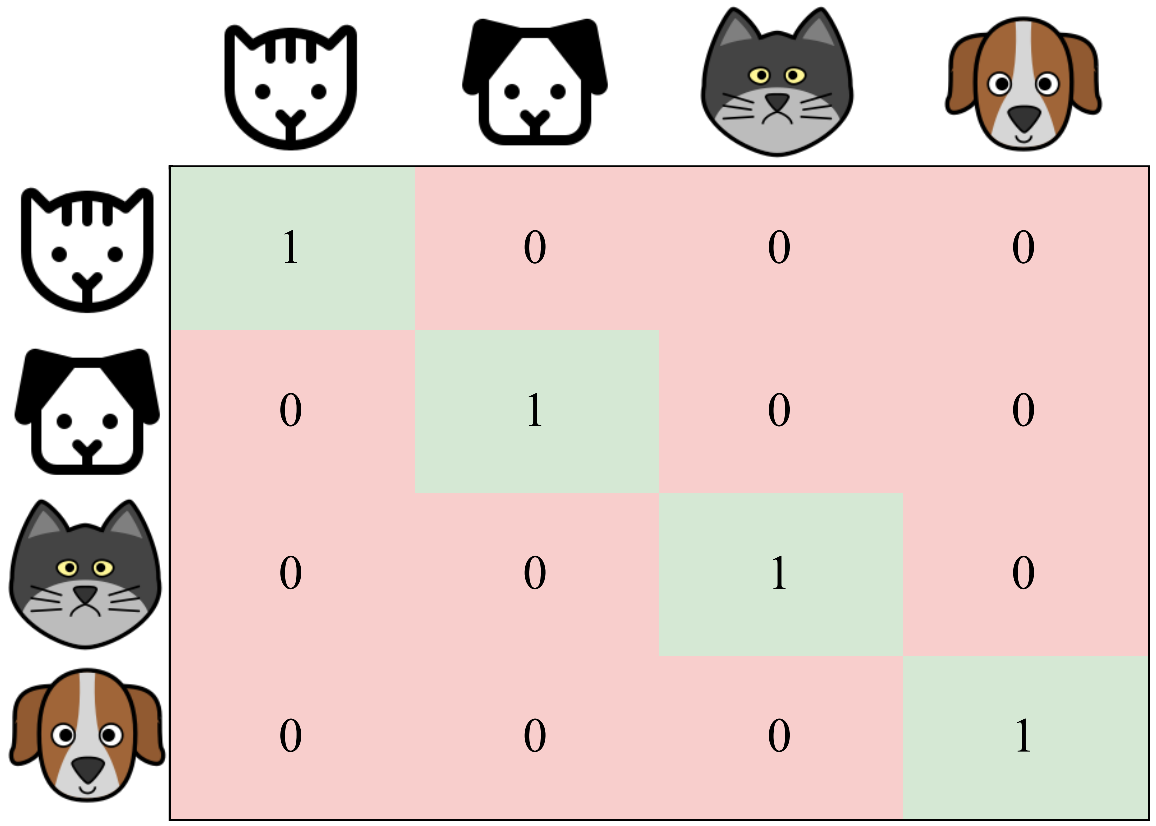


Figure 3.5: SimCLR loss function. From left to right, the labels are cat, dog, cat, dog. The x-axis and y-axis represent two different augmentations of a mini-batch image. The value in the table represents the pseudo-labels.

the practical results by Chen et al. [10], the authors added projection head into both networks. As Fig. 3.3, the network with gradient update is named online network and the network with momentum update is named target network.

$$online(x) = q_\theta(g_\theta(f_\theta(t(x)))),$$

$$target(x) = sg(g_\xi(f_\xi(t'(x)))).$$

The goal for BYOL framework is using online network to predict the output from target network. For this propose, the author add a prediction head to online network. The framework optimize as:

$$min(\mathcal{L}_{\theta,\xi}(online(x), target(x))).$$

The author claims that BYOL does not require negative pairs for contrasting because the framework is learnt from the previous version's output. They also proposed that the prediction header is essential to prevent representational collapse, and it needs to be near-optimal at all the time [21]. Some studies also mentioned that batch normalization is critical in the projection head and prediction head. As pseudo code that BYOL provided, they used batch normalization in the MLP(multilayer perceptron) blocks. The batch normalization layer would potentially involve information from negative samples and BYOL still using negative pairs to avoid collapse. In other words, they believe BYOL is an implicit contrastive learning method. However, Richemond et al. refute this hypothesis by proposing an alternative method without batch statistics [55]. They proved that batch normalization could be replaced by group normalization(GN) and weight standardization(WS), and the performance is almost the same as using batch normalization.

SimSiam

After BYOL proposed a framework without contrasting with negative pairs, Chen et al. further explored the method to avoid collapsing. They further simplified the self-supervised learning structure and proposed a Siamese network named SimSiam [12]. SimSiam removed the momentum encoder and the projection head. Two encoders directly share the weights, as Fig. 3.4. Compared to SimCLR, SimSiam does not need negative samples in the loss function, so a large batch size is also not necessary for SimSiam. SimSiam further shrunk the essential condition to prevent collapse. Similar as BYOL, SimSiam has a prediction head on the online encoder and stop gradient on the target encoder(the network without predictor). The online network update parameters by the gradient from loss function. The author proposed stop-gradient and the predictor are the essentials to avoid collapse on SimSiam.

3.1.3 Motivation and difficulties

There are still many areas worth exploring, especially the efficiency of learning representation and how to prevent collapsing during training. Both MoCo [23] and SimCLR [10] indicated that the increment of the number of the negative sample could improve the learning efficiency of sample representation. However, there is only one positive sample in the whole batch for each training sample. Even though these self-supervised learning frameworks already outperformed the supervised version, we believe the additional positive label information could be used to unrestricted the potential learning ability in latent space. This study will discuss the feasibility of increasing the number of positive samples in loss function to further improve the representation learning efficiency. Overall, our contributions are listed as follows:

- We proposed a supervised contrastive learning representation framework. Our proposed framework contrast multi-positive label at the same time.
- The convergence speed of our proposed multi-positive loss function is much more faster than SimCLR and supervised model and the upper bound of the proposed loss function is lower than the NT-Xent (SimCLR loss function).
- In the structure part, we challenged a common technique used in most SSL recent works – projection head. After experiments, we found that the projection head is unnecessary for our framework.

3.2 Method

3.2.1 Background

SimCLR [10] proposed a simple, efficient framework to learn representations by contrasting positive and negative samples. This framework outperformed the supervised version on many datasets by both linear evaluation and finetuned. We proposed a similar framework for learning representations based on similarities and dissimilarities. The significant difference is that our framework considers learning from multi-positive labels. SimCLR provides an idea to learn representations not only on self-supervised learning methods it can also improve the efficiency of supervised learning. This leads us to think that even SimCLR outperformed the same supervised model, can we properly use the label information to provide additional information for the framework to learn the representations only?

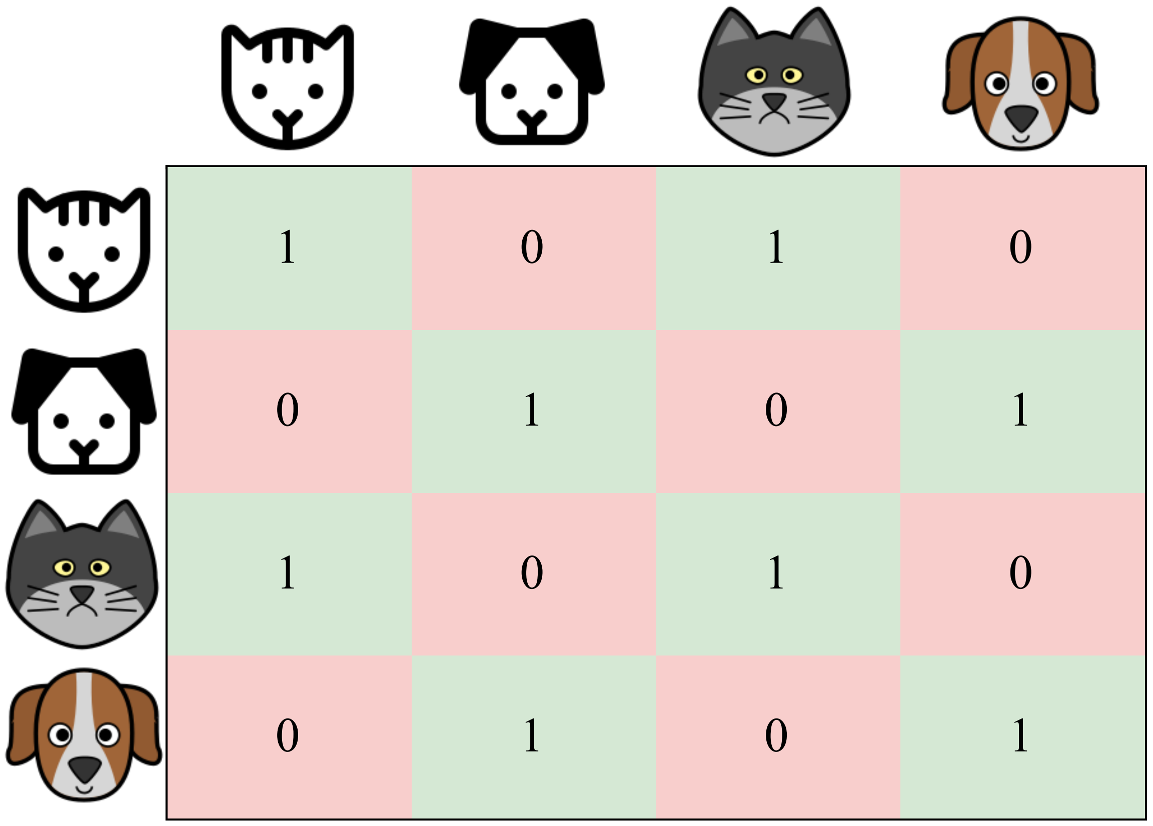


Figure 3.6: SupCLR loss function. From left to right, the labels are cat, dog, cat, dog. The x-axis and y-axis represent two different augmentations of a mini-batch image. The value in the table represents the pseudo-labels.

To provide useful information for contrastive learning, we only provide additional similar label information to the loss function. In other words, the NT-Xent loss(Fig. 3.5) only selects the image with different augmentation as the only pair of positive samples in the mini-batch. Our proposed method assigns all the images with the same true labels as positive samples as Fig. 3.6.

3.2.2 Framework

In our framework, we proposed a supervised contrastive learning method (SupCLR). Here are the training steps:

The augmentation function t generate two different augmented images $x_q^{\sim}$ and $x_k^{\sim}$ by:

$$x_q^{\sim} = t(x),$$

$$x_k^{\sim} = t'(x).$$

Then encode the augmented images by weight sharing encoder E to generate representations h_q and h_k:

$$h_q = E(x_q^{\sim}),$$

$$h_k = E(x_k^{\sim}).$$

SimCLR [10] found the projection head is beneficial to preventing the encoder from learning unrelated features based on the contrastive loss function, such as the framework needing to resume the similarities from the same image with different augmentations $x_q^{\sim}$ and $x_k^{\sim}$. The projection head will learn these features, and the projection head will be removed in downstream tasks. We also added the weight-sharing projection head in our framework named $Proj$. We use 3-layer MLP as a non-linear projection head, which comes with ReLU as an activation function.

$$z_q = Proj(h_q),$$

$$z_k = Proj(h_k).$$

Finally, we want to minimize the contrastive loss function $\mathcal{L}$. The next section will introduce the loss function in detail.

3.2.3 Loss Functions

The contrastive loss function will compare the final output z_q and z_k. The purpose of the NT-Xent (SimCLR contrastive loss function) with an input sample a is:

$$max(sim(z_{qa}, z_{ka})), \tag{3.1}$$

$$min(\sum_{b}^{n} \mathbb{1}_{[b \neq a]} sim(z_a, z_b)), \tag{3.2}$$

where $\mathbb{1}_{[b \neq a]}$ is an indicator function equals to 1 only if $b \neq a$ and n is the batch size. To simplify the expressions, we will omit the q and k for the similarity between different original images(index). For example, $sim(z_{qa}, z_{kb}) + sim(z_{qb}, z_{ka}) \equiv sim(z_a, z_b)$.

Inspired by the NT-Xent we proposed a multi-positive label contrastive loss function. Given m classes label corresponding to the input data with batch size n. In the loss function, all samples with index $1...n$ belong to one of the label sets $L_1...L_m$. The objective of our loss function corresponding to input sample a is:

$$max(\sum_{c\in L_i} sim(z_a, z_c)\ a \in L_i), \tag{3.3}$$

$$min(\sum_{b}^{n} \mathbb{1}_{[b\notin L_i]} sim(z_a, z_b)\ a \in L_i), \tag{3.4}$$

where $\mathbb{1}_{[b\notin L_i]}$ indicate z_a and z_b have different labels and there are in total $|L_i|$ elements in the class set L_i. In the loss function, we want to maximize the similarity between all samples belonging to the same label and minimize the similarity of all samples between different labels. Overall, the loss function for an input sample a is defined as:

$$\mathcal{L}(a) = -\log \frac{\sum_{c\in L_i} \exp(sim(z_a, z_c)/\tau)}{\sum_{b}^{2n} \mathbb{1}_{[b\notin L_i]} \exp(sim(z_a, z_b)/\tau)}\quad a \in L_i. \tag{3.5}$$

In the loss function L_i is the label set of sample a. There are two steps in the contrastive loss function. First, we need to convert labels into relationship matrix Mat between different sample labels as Fig. 3.6. This matrix define as

$$Mat_{ij} = Mat_{ji} = \begin{cases} 0 & label_i \neq label_j \\ 1 & label_i = label_j \end{cases}. \tag{3.6}$$

with shape $batchsize * batchsize$. Then we use z_q, z_k and the matrix Mat to calculate the loss as Algorithm 3.

We used two different label matrices in the algorithm to calculate the loss. One is for the positive part, and the other one is for the negative part. Note that the positive and negative labels do not represent the similarity or dissimilarity between samples. The positive label matrix (Mat) is the label for $logit_{ab} and logit_{ab}$. We expect only the contrast of positive sample pairs to be 1. The negative label matrix ($negLabel$) is the label for $logit_{aa} and logit_{bb}$. After removing all positive pairs, we expect all the rest of the sample pairs in $logit_{aa} and logit_{bb}$ to be 0.

Algorithm 3 Multilabel Contrastive Loss

```
Input latent space features $z_q$, $z_k$ and matrix similarity label $Mat$
 1: $logit_{aa} = z_q \circ z_q^T / \tau$   ▷ calculate logit from same augmentation
 2: $logit_{aa} = logit_{aa} - Mat * 10^9$   ▷ remove positive samples
 3: $logit_{bb} = z_k \circ z_k^T / \tau$
 4: $logit_{bb} = logit_{bb} - Mat * 10^9$
 5: $logit_{ab} = z_q \circ z_k^T / \tau$   ▷ calculate logit from different augmentation
 6: $logit_{ba} = z_k \circ z_q^T / \tau$
 7: $sim_a = [logit_{ab}, logit_{aa}]$
 8: $sim_b = [logit_{ba}, logit_{bb}]$
 9: $negLabel = zerosLike(masks)$   ▷ all-zero negative label, same shape as $Mat$
10: $multilabel = [Mat, negLabel]$   ▷ combine positive and negative labels together
11: $loss_a = softmaxCrossEntropy(multilabel, sim_a)$   ▷ calculate contrastive loss
12: $loss_b = softmaxCrossEntropy(multilabel, sim_b)$
13: $loss_{total} = mean(loss_a, loss_b)$
14: return $loss_{total}$
```

3.2.4 Theoretical Analysis of the Loss Function

This section discusses how well our loss function could learn from multi-positive contrastive learning compared to the NT-Xent (SimCLR loss function).

Discussion

SimCLR proposed NT-Xent as their contrastive loss function. NT-Xent used one pair of samples as positive and the rest samples from batch size as negative. The advantage is this loss function could benefit from a larger batch size because it would have more negative samples to contrast. However, the efficiency is acceptable but not promising enough because the loss function could arrange all same downstream-label samples as negative and extend the distance on latent space. To address the following issue, we systematically analyzed the upper bound of the NT- Xent loss function and our proposed loss function. We hypothesize our proposed loss function could benefit from multi-positive pairs and hence gain more information in larger batch size compare to SimCLR.

Lemma 1. *At least two positive pairs correspond to one representation in a mini-batch exist if the batch size n is larger than the number of label classes m.*

Proof. If the batch size $n = m$, the worst case is that all label $L_1, \dots L_m$ exist in one sample. However, if $n = m + 1$ one subset in $\{L_1, \dots L_m\}$ must exists more than one sample which means in at least one subset L_i there exists two positive pairs $\{(x_{aq}, x_{ak}), (x_{aq}, x_{bk})\}$ correspond to a single representation. $\square$

Upper Bound

According to Lemma 1, we will analyze the upper bound of our proposed loss function when the mini-batch size is larger than the number of label classes. Based on our loss function equation (3.5), the total loss for all samples in batch size n could be present as:

$$\mathcal{L} = -\frac{1}{n}\sum_{a}^{n} \log \frac{\sum_{c\in L_i} \exp(sim(z_a, z_c)/\tau)}{\sum_{b}^{2n} \mathbb{1}_{[b\notin L_i]} \exp(sim(z_a, z_b)/\tau)} \quad a \in L_i, i \in m, \tag{3.7}$$

where L_i represents one of the label sets in m classes that always contains the index of the same label samples as a. Based on the logarithmic rules we can rewrite Equation (3.7) to

$$\begin{aligned} \mathcal{L} = &-\frac{1}{n}\sum_{a}^{n} \log(\sum_{c\in L_i} \exp(sim(z_a, z_c)/\tau)) \\ &+\frac{1}{n}\sum_{a}^{n} \log(\sum_{b}^{2n} \mathbb{1}_{[b\notin L_i]} \exp(sim(z_a, z_b)/\tau)) \quad a \in L_i, i \in m, \end{aligned} \tag{3.8}$$

and based on the previous studies on the NT-Xent loss [21, 89], the parts inside the *log* could be convert to the LogSumExp(LSE) term:

$$LSE(x_1, \ldots, x_n) = \log(\exp(x_1), \ldots, \exp(x_n)), \tag{3.9}$$

where $x_{a,b}$ could be denoted as $sim(z_a, z_b)/\tau$. Then we can rewrite the loss function as

$$\begin{aligned} \mathcal{L} = &-\frac{1}{n}\sum_{a}^{n} \log(\sum_{c\in L_i} \exp(x_{a,c})) \\ &+\frac{1}{n}\sum_{a}^{n} \log(\sum_{b}^{2n} \mathbb{1}_{[b\notin L_i]} \exp(x_{a,b})) \quad a \in L_i, i \in m. \end{aligned} \tag{3.10}$$

Since there exists at least one positive pair of sample, and only the negative pairs

($\mathbb{1}_{[b \notin L_i]}$) exist in the second term in loss function

$$\log(\sum_{b}^{2n} \mathbb{1}_{[b \notin L_i]} \exp(x_{a,b})) < LSE(x_{a,1}, \ldots, x_{a,N}). \tag{3.11}$$

Based on the properties of the LogSumExp function

$$max\{x_1, \ldots, x_n\} \leq LSE(x_1, \ldots, x_n) \leq max\{x_1, \ldots, x_n\} + \log(n), \tag{3.12}$$

we can simplify the upper bound of our proposed loss function by

$$\begin{aligned} \mathcal{L} <& -\frac{1}{n}\sum_{a}^{n} LSE(x_{a,L_{i1}}, \ldots, x_{a,L_{il}}) \\ &+ \frac{1}{n}\sum_{a}^{n} LSE(x_{a,1}, \ldots, x_{a,N}) \quad a \in L_i, i \in m, \\ <& -\frac{1}{n}\sum_{a}^{n}(max(x_{a,L_{i1}}, \ldots, x_{a,L_{il}}) + \log(|L_i|)) \\ &+ \frac{1}{n}\sum_{a}^{n}(max(x_{a,1}, \ldots, x_{a,N}) + \log(N)) \quad a \in L_i, i \in m, \\ =& -\frac{1}{n}\sum_{a}^{n}(max(x_{a,L_{i1}}, \ldots, x_{a,L_{il}})) - \log(|L_i|) \\ &+ \frac{1}{n}\sum_{a}^{n}(max(x_{a,1}, \ldots, x_{a,N})) + \log(N) \quad a \in L_i, i \in m, \end{aligned} \tag{3.13}$$

where $l = |L_i|$ is the number of postive pairs related to x_a. Based on Agren's study [89], the upper bound of the NT-Xent loss function could be represent as

$$\mathcal{L}^{NT-Xent} = -\frac{1}{n}\sum_{i,j \in MB} x_{i,j} + \log(N) + \frac{1}{n}\sum_{i}^{n} max(x_{i,1}, \ldots, x_{i,N}), \tag{3.14}$$

where MB refers to the current mini-batch of data and $x_{i,j}$ represents the only positive pair in the NT-Xent. We will compare our loss function expression with the NT-Xent term by term. In the first term of equation(3.13), $max(x_{a,L_{i1}}, \ldots, x_{a,L_{il}})$ already included the only positive pair from NT-Xent $x_{i,j}$. Therefore,

$$\begin{aligned} max(x_{a,L_{i1}}, \ldots, x_{a,L_{il}}) &\geq x_{i,j}, \\ -\frac{1}{n}\sum_{a}^{n}(max(x_{a,L_{i1}}, \ldots, x_{a,L_{il}})) &\leq -\frac{1}{n}\sum_{i,j\in MB} x_{i,j}. \end{aligned} \tag{3.15}$$

Furthermore, since the number of positive pairs in the set L_i must be positive. Based on Lemma 1, when there exists more than one positive pair, $|L_i| > 0$ and

$$\begin{aligned} &-\log(|L_i|) + \frac{1}{n}\sum_{a}^{n}(max(x_{a,1}, \ldots, x_{a,N})) + \log(N) \\ &\leq \log(N) + \frac{1}{n}\sum_{i}^{n} max(x_{i,1}, \ldots, x_{i,N}). \end{aligned} \tag{3.16}$$

Based on the two parts of proof(3.15, 3.16), our loss function has lower upper bound compare to the NT-Xent if and only if there exists more than one positive sample pair($|L_i|$) in the current mini batch. More information is gained in contrasting if there are more positive pairs in a mini-batch. Based on equation (3.15), if there exists a pair x_{a,L_i} larger than the original positive pair $x_{i,j}$ in the NT-Xent, the upper bound could be further reduced.

3.3 Experiment

In this section, we focus on study the innovation of SupCLR and the contributions on multi-positive labels loss function.

3.3.1 Experiment Setup

We access the performance of our proposed methods by CIFAR10, CIFAR100 and ImageNet datasets, respectively. All the experiments are running on 4 RTX6000 GPUs. All our experiments were implemented using Tensorflow 2.8.0 with ResNet-50(1X) architecture as our backbone encoder. We used 3 MLP-layers as nonlinear projection head and 1 MLP layer as linear evaluation output layer. In both the pretraining and finetuning step, the hyperparameters of our methods are the same as the SimCLR baseline method. On CIFAR10 and CIFAR100, we first evaluate the performance on pretraining linear evaluation with mini-batch sizes as 1024 and

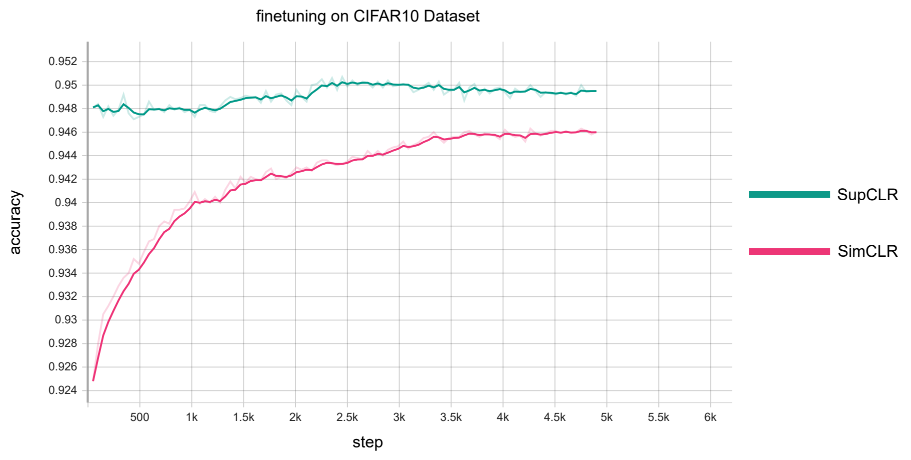
1000 training epochs. Then we evaluate the finetuning result with 100% labels with mini-batch size as 1024 and 1000 training epoch.

Table 3.1: Accuracy comparison. The SupCLR w/o positive and SupCLR w/ sigmoid will be introduced in Ablation Study 3.4.1 and 3.4.2

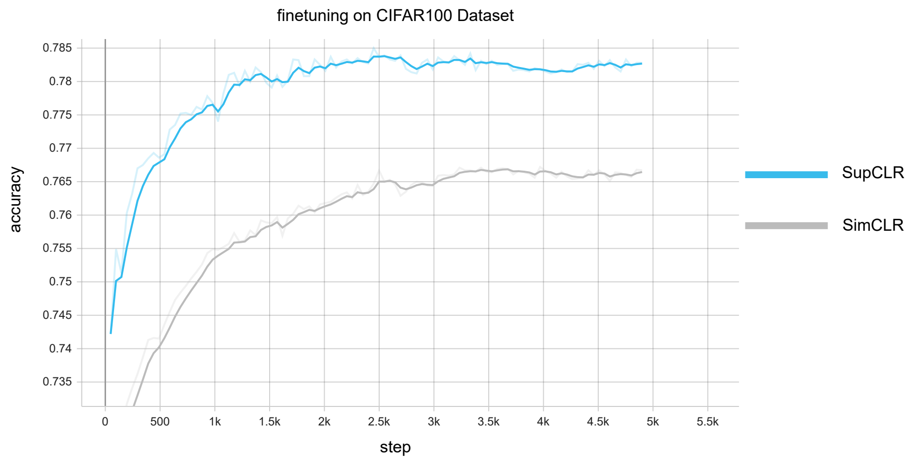
Methods	CIFAR10		CIFAR100	
-	Pretrain	Finetune	Pretrain	Finetune
ResNet-56 (result from paper [24])	-	93.03	-	-
ResNet-50 backbone	-	83.81	-	52.32
SimCLR [10]	92.89	94.6	71.56	76.64
SupCLR w/o positive	93.89	94.82	72.81	77.08
SupCLR w/ sigmoid	94.61	95.08	73.36	77.64
Our proposed (SupCLR)	**95.09**	**95.1**	**74.15**	**78.27**

3.3.2 Performance Comparison

In this section, we compared the performance of our proposed model with SimCLR. Similar to SimCLR's method, we used a linear layer to evaluate the pretraining performance in the pretraining step. The parameters in this layer would be trained by the training data labels. However, the label information would stop the gradient on this layer and not affect the model pretraining. Our proposed framework significantly outperformed the original SimCLR framework with a similar training environment and hyperparameters (Table 3.1). In the pretraining part, our proposed method performs better because we involved similar labels as additional information in the loss function. The notable gap in the finetuning part reflects the differences in representation learning efficiency between the two methods. To further illustrate the comprehensive learning ability of our proposed framework, we analyzed the finetuning process on the CIFAR10 experiment. From Fig.3.7a, our proposed method's evaluation accuracy has inconspicuously increased during 100 epochs, which also shows our proposed method learnt the majority of meaningful representations without categorical labels in the pretraining step.



(a) CIFAR10 finetune comparison between SimCLR and SupCLR



(b) CIFAR100 finetune comparison between SimCLR and SupCLR

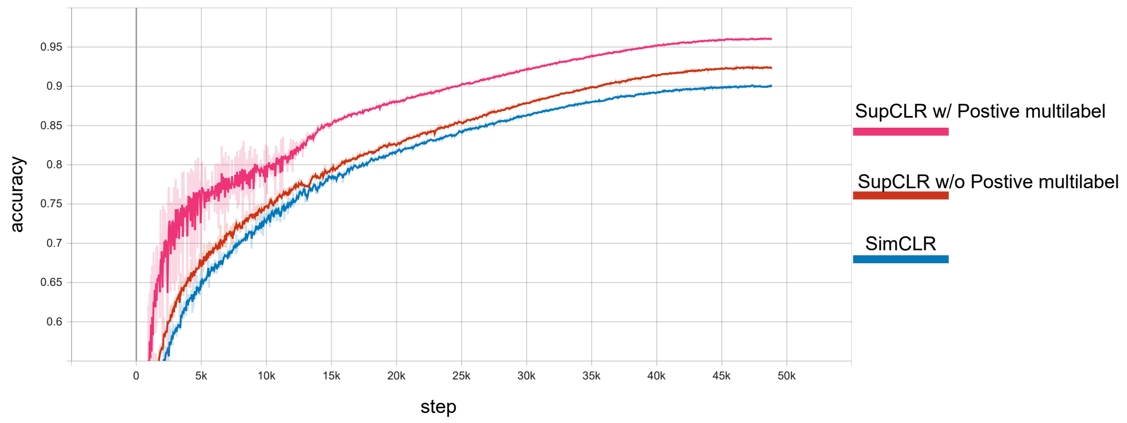
Figure 3.7: Batch size experiment

3.4 Ablation Study

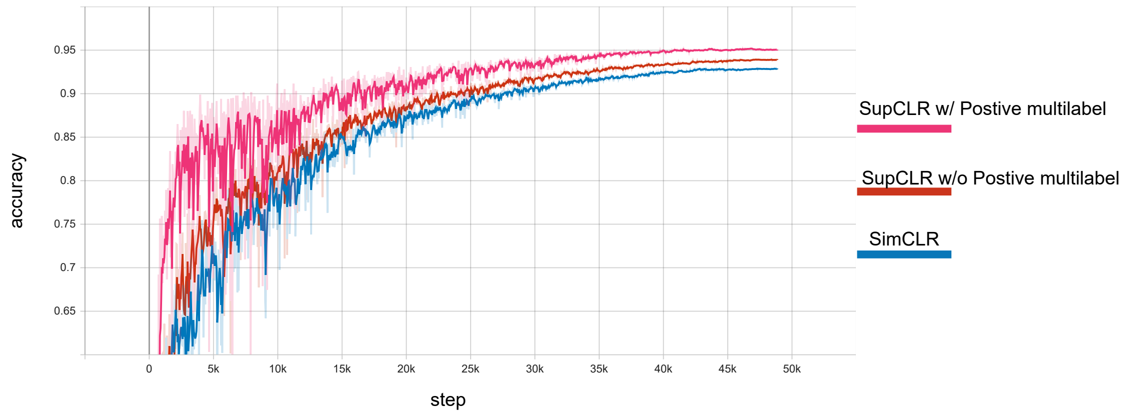
We present ablations on SupCLR to provide a deeper understanding of its performance.

3.4.1 Remove multi-positive loss

As we mentioned in the method part (Algorithm 3), our loss function contains a positive and negative part. To estimate if the multi-positive labels could help our framework to gain representation learning efficiency, we removed the multi-positive labels and used only one pair of positive labels, similar to SimCLR (Fig. 3.5). To keep all other conditions the same as our proposed method, we preserved multilabel information in the negative part.



(a) training accuracy comparison



(b) linear evaluation accuracy comparison

Figure 3.8: Comparison between operating multi-positive labels in contrastive loss function and not using multi-positive labels in the contrastive loss function

Fig.3.8 presents the result of pretraining and linear evaluation accuracy. From both the training and linear evaluation process, the SupCLR without positive labels and SimCLR have similar trends during the training process. In comparison, SupCLR with a multi-positive label achieves better accuracy, especially in the first 5k steps. The result displayed that multilabel loss could boost the convergence speed. Furthermore, it is worth noting that even if we involved multi-positive labels in the loss function, there is no overfitting problem in the whole pretraining stage (Fig.3.8a, Fig.3.8b).

3.4.2 Softmax versus Sigmoid for multilabel loss

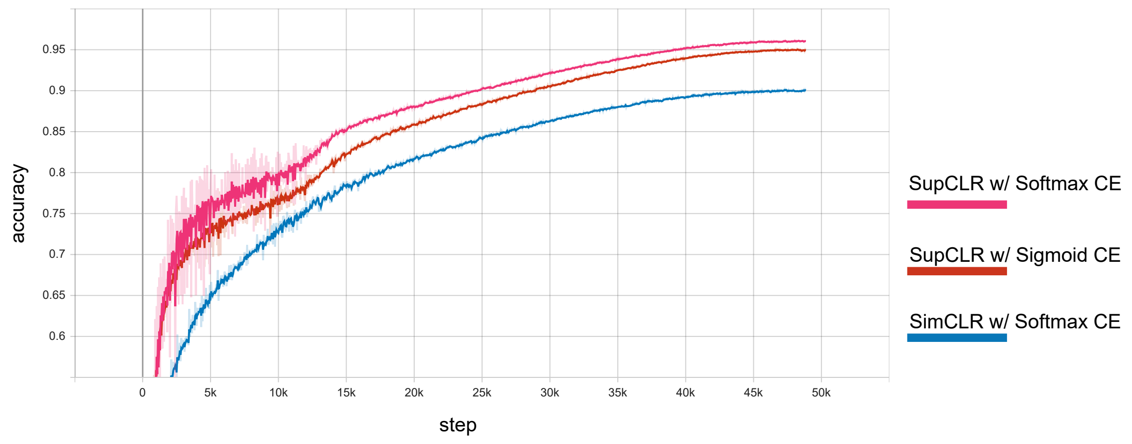
Using softmax cross entropy in multilabel classification tasks has always been counter-intuitive compared to the sigmoid binary cross entropy loss function. The reason is that people encourage each output prediction to be an independent possibility, such as the sigmoid binary cross entropy loss function. The softmax cross entropy is precisely the opposite of what we expect. If the possibility of one label is higher, the others would be lower. However, as Mahajan et al.'s study [49], the softmax cross entropy loss performs much better than the sigmoid binary cross entropy loss. We also involved multi-positive labels in our loss function (Fig. 3.6). In this experiment, we compared the performance of these two functions in the same environment.

Similar to the last experiment, we first compared the pretraining process between softmax, sigmoid and softmax SimCLR (Fig.3.9a). Unlike removing the positive labels, the sigmoid and softmax versions keep similar trends in this comparison. The accuracy is also remarkably close. In this situation, we performed two runs on finetuning stage to see if there were any apparent differences. In Fig.3.9b, the performance looks unstable, but the difference between the four runs is only around 0.001%. Therefore, there is no obvious difference between softmax and sigmoid cross entropy loss with a batch size of 1024.

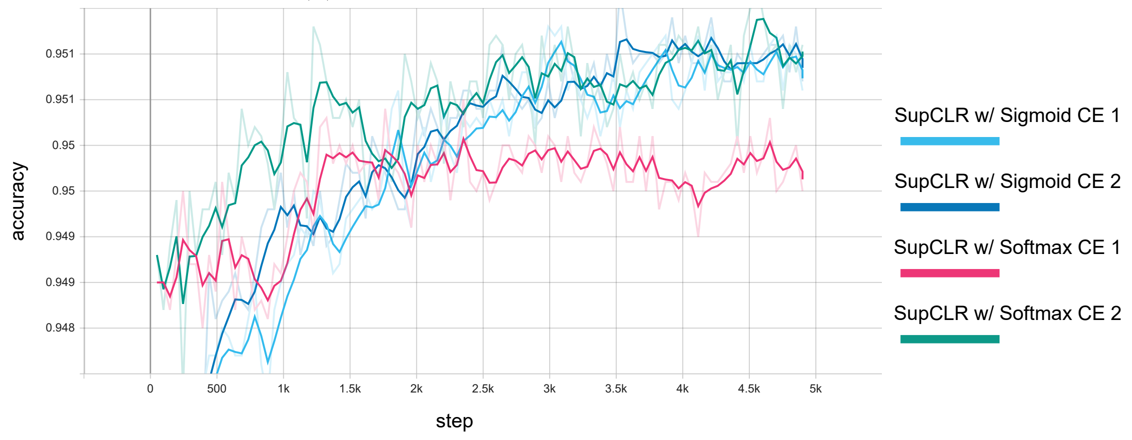
3.4.3 Projection head and Shallower Backbone

Most previous self-supervised learning studies [10–12, 21] used projection head as part of the pretraining framework and withdrew it during finetuning stage. The reason is that the non-linear MLP projection head learns to handle different augmentations, and a similar image will be projected to a shorter distance in latent space. However, handling augmentation tasks became a counterproductive part during finetuning.

Different from previous studies, we hypothesize the projection head could benefit the classification task during the finetuning stage based on the multilabel loss function. Here is the reason why we made this assumption. The normal contrastive loss function only involves two feature inputs into the loss function. As the loss decreases, the projection head is trained to ignore the augmentations. The essential difference between our loss function is that we involved two features and the true label (Mat) as the input. Thus, the projection head could be trained to ignore augmentation and learn the true label information synchronously. In addition, we want to compare both methods with a shallower encoder and fewer pretraining epochs. Unlike



(a) pretrain training accuracy comparison

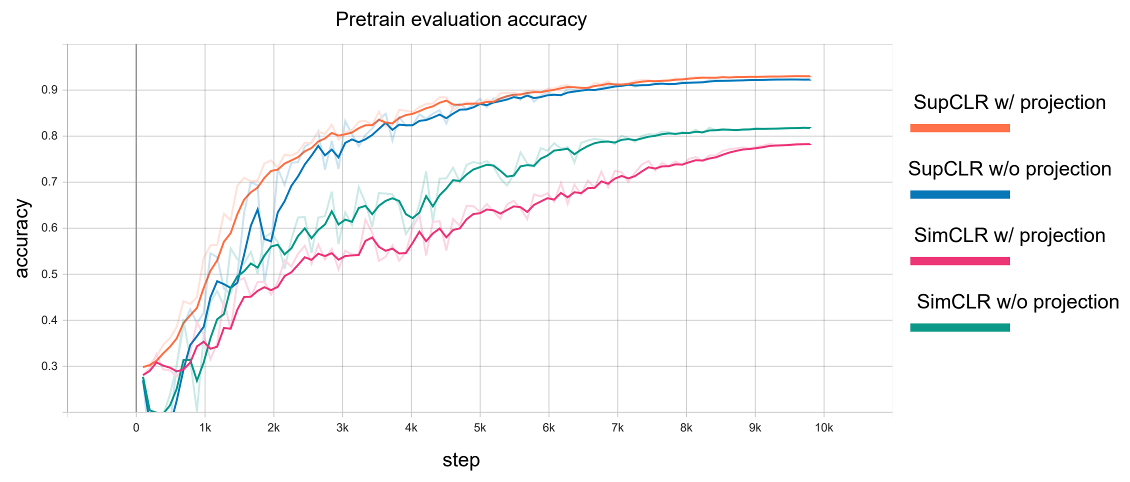


(b) finetuning evaluation accuracy comparison

Figure 3.9: Comparison between Softmax Cross-Entropy and Sigmoid Cross-Entropy in the multilabel loss function

previous experiments, this used ResNet-18(1X) as the backbone. Both pretraining and finetuning stages are 100 epochs with batch size of 512. This experiment trained with one RTX 3080 Ti GPU (12 GB Memory). From Fig.3.10, the 'SupCLR w/o projection' and 'SimCLR w/o projection' are the original frameworks we used in previous experiments. The linear evaluation layer is directly connected to the backbone encoder. The 'SupCLR w/ projection' and 'SimCLR w/ projection' preserved the projection head in both the pretraining linear evaluation and the finetuning stage.

Based on the result(Fig.3.10), we can verify that the projection head is redundant in both SimCLR finetuning and pretraining evaluation stages. However, the SupCLR has the opposite result as we hypothesized. In both stages, the projection head



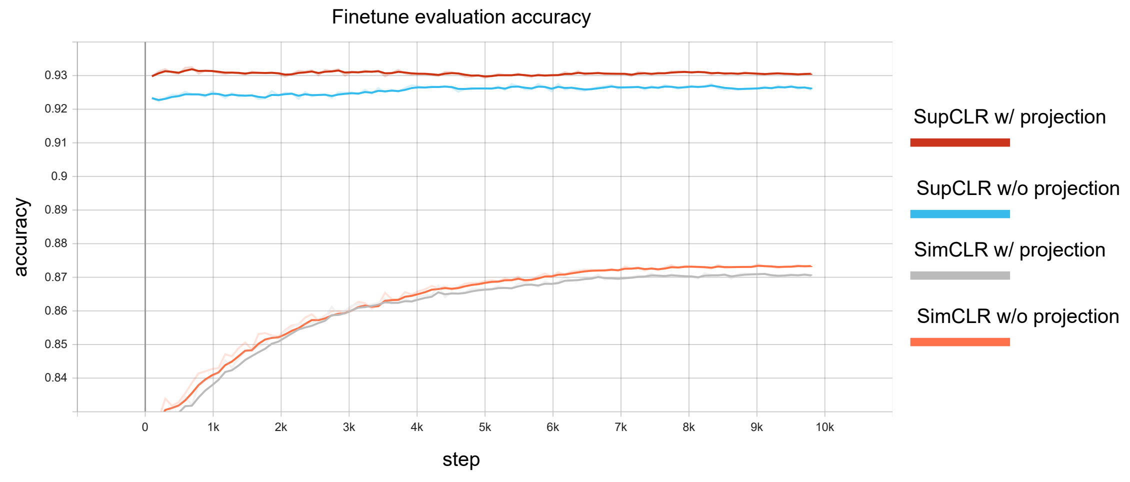


Figure 3.10: Pretrain and finetune performance on projection head experiment. SupCLR w/o projection is the original method. In SupCLR w/ projection, the finetuning process preserved the projection head as part of the framework.

involved leads to better performance. The difference in finetuning stage is more significant than SimCLR. It is worth noting that there is a 10% accuracy gap between SupCLR and SimCLR under similar conditions.

Overall, we proved that preserving the projection head could benefit our framework. In other words, the property of the projection head does not exist anymore since it is a part of the encoder. Our framework also has a more promising convergence speed than SimCLR.

3.4.4 Batch size

SimCLR [10] reported that the model could benefit from a larger batch size due to larger negative sample size in one iteration. Similarly, we proved that our method could also profit from a larger batch size due to the increase of positive pairs. In this experiment, we would compare both pretraining and finetuning evaluation accuracy on both methods with different batch sizes (128, 256, 512, 1024) on CIFAR100. Due to computational resource limitations, our largest batch size is 1024.

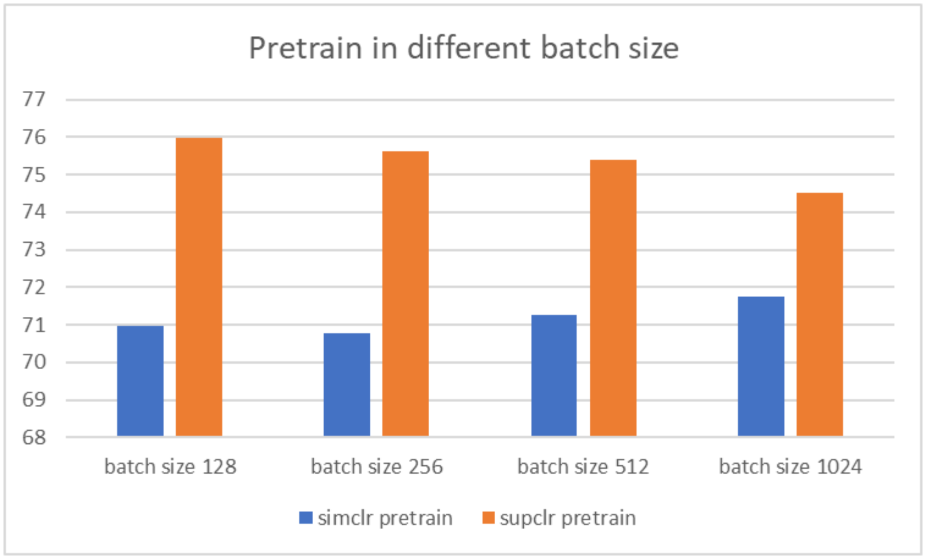
In this experiment, our proposed method outperformed SimCLR in all different batch size (Fig.3.11b). Based on the finetuning result, the accuracy of SimCLR boosts when the batch size grows. In addition, the gap between the SimCLR finetuning result and the SupCLR finetuning result also expands as the batch size grows, which confirms our theoretical analysis of the multilabel loss function. An unexpected finding is that the pretrained linear evaluation accuracy on SupCLR decreases as the batch size increase. The specific batch size finetuning stage is trained based on the same batch size pretrained model. For example, batch size 256 finetuning result trained based on batch size 256 pretrained model. It denotes that the downtrend in the pretraining stage does not affect the uptrend in finetuning stage.

3.4.5 Compare with Supervised Backbone

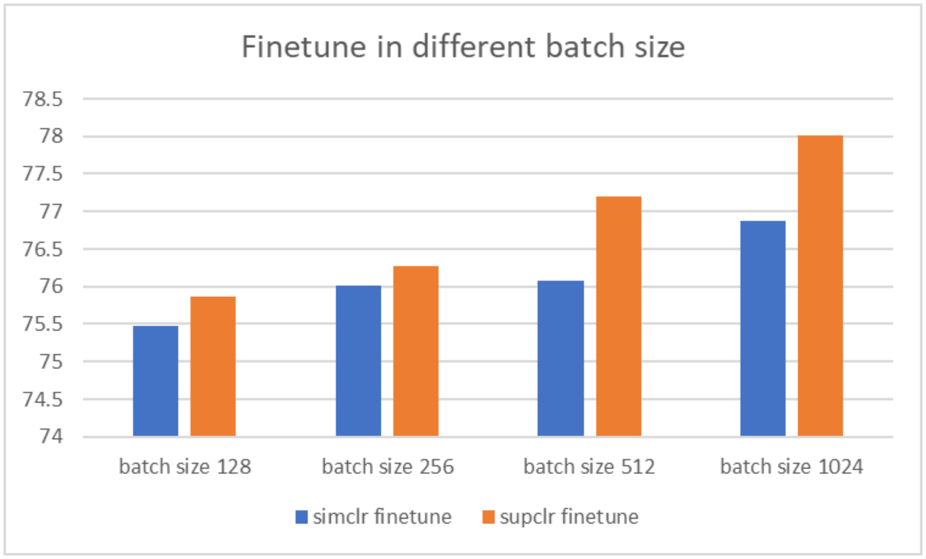
We used similar finetuning parameters to run the ResNet-50 supervised learning experiment from scratch. To ensure relative fairness, we trained the supervised ResNet-50 for 1100 epochs because we ran 1000 epochs for pretraining and 100 epochs for finetuning on other models. However, the accuracy is very low and severely overfitting was found on CIFAR100(Fig.3.12). We conclude that the self-supervised learning parameter may not suitable for supervised training. Therefore we cited the CIFAR10 performance in Table 3.1. The result came from [24]. The result from original paper still cannot keep up with contrastive learning result.

3.5 Discussion

In our experiments, we illustrate the advantage of our framework from several aspects. With similar parameters and environments, our proposed method outperformed SimCLR in both CIFAR10 and CIFAR100 datasets. The performance gap after removing multi-positive label loss reflected the significance of our multilabel contrastive



(a) Pretraining comparison between SimCLR and SupCLR



(b) Finetuning comparison between SimCLR and SupCLR

Figure 3.11: Batch size experiment

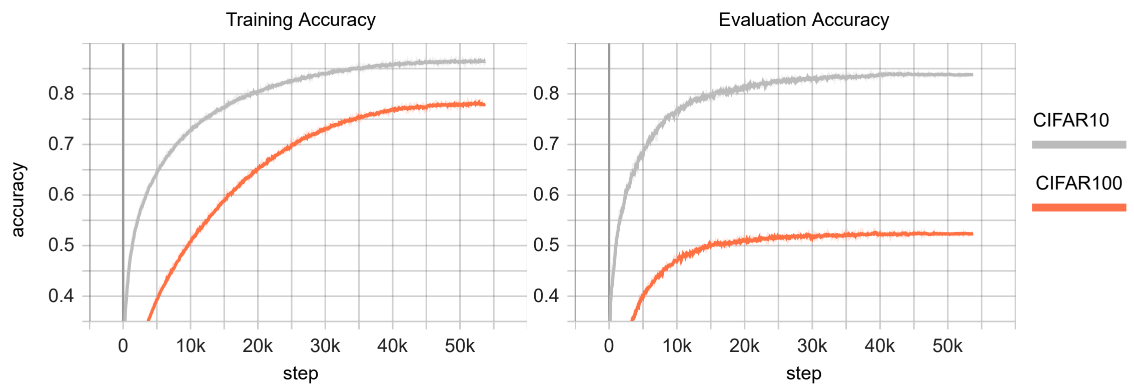


Figure 3.12: Supervised training from scratch performance on ResNet-50

loss function. Furthermore, the batch size experiment proved our hypothesis and theoretical analysis that our framework could gain more information as batch size increases compared to the NT-Xent loss function. The batch size experiment did not include the projection head in the evaluation. There is one reasonable suspect based on the conclusion of the projection head experiment. As the batch size grows, more label-based information could be learnt by the projection head, and the encoder would learn more feature representation-based information. So in the linear evaluation result is decreasing, but as long as the encoder is finetuned with true labels, the evaluation performance still increases. The projection head experiment is also an essential innovation in our method. Withdrawn the projection head in finetuning stage could cause information loss because the projection head and encoder are pretrained together based on contrastive function. This challenge existed in most recent self-supervised learning studies. Benefiting from our loss function, the projection head could be trained with label-related features, further boosting our performance. One unexpected finding is that the softmax cross-entropy loss could lead to unstable pretraining in the earlier stage (Fig.3.8a), especially if the batch size is larger than 1000. We did not test if the sigmoid cross-entropy loss could overcome this issue due to computational resource limitation.

3.6 Conclusion

The chapter concludes by arguing the efficiency and differences of the multilabel contrastive loss function based on traditional methods. Our experiments and theoretical analysis systematically analyzed the characteristic of our loss function. We conclude that multilabel contrast loss function is feasible and worth exploring. In application scenarios, this framework could be applied on different backbones to replace any supervised learning task. More work could be conducted on stabilizing the softmax cross-entropy on the multilabel task.

Chapter 4

Conclusion & Future Work

4.1 Overview

We introduced our proposed frameworks of this thesis in Chapter 2 and Chapter 3. In both these chapters, we designed many experiments to prove our hypothesises and test the robustness of the frameworks.

4.2 Main Contributions

As Menghani suggested in the efficient deep learning project [17] and efficient deep learning survey [51], training efficiency and interface efficiency are two main aspects that we can boost in a task. From the interface perspective, we illustrated a less human interface spatial feature extraction method for EEG emotional classification task in Chapter 2. We innovated a multilabel contrastive loss function from the training efficiency perspective to reach similar performance in previous studies with less training time.

There are three key contributions of this thesis:

- **First** Most EEG spatial features manually select channel locations or use the 1D-CNN layers for extraction. However, manually selecting channel methods are not durable across different datasets. The result on 1D-CNN is also not promising. We presented a DCNet structure to solve this challenge. Our framework converts 1D signal data into a 2D structure that preserves spatial information. After that, we used well-known 2D CNN layers to extract the spatial features.

- **Second** DCNet has outstanding transfer learning flexibility due to the application of the 2D-CNN part. We proved that even ImageNet pretrained 2d-CNN structure could benefit on EEG classification task.

- **Third** We proposed a multilabel contrastive loss function to explore the representation learning area. Previous self-learning structures only learn similarities from the same samples. Although the SSL methods perform better than supervised learning, we applied multi-positive contrasting to guide the framework learning similarities between small categories but different samples. This innovation further intensified the representation learning efficiency.

4.3 Conclusion

In this thesis, we proposed two deep learning frameworks for improving efficiency on two tasks (EEG emotional classification and image classification).

Chapter 2 presents an idea to extract spatial features without additional preprocessing techniques. Even though our promising performance is essential to prove the efficiency of our spatial and frequency features combination framework, the high scalability(transfer learning) and compatibility(fewer human-interaction) of our framework are more significant in the future application scenario. Although we dealt with emotional classification tasks in this study, we believe that applying the proposed framework to other EEG signal data or multimodal data could be successful.

Chapter 3 provided a new contrastive learning structure. It presents that even with a similar structure and the same hyperparameters, the backbone encoder still has the potential to learn more representation features and boost performance. This framework could be applied with other supervised frameworks as the backbone and boost their performance.

4.4 Future Work

Future research can be conducted to verify the effectiveness of the DCNet on the one-dimension dataset from more areas. As a next step, researchers could also consider constructing a new 3D-DCNet framework to include both spatial and temporal features from EEG signal data. We expect that the 3D model would significantly reduce the time required for training and preserve as much temporal information as possible. It would be interesting to apply our DCNet to transfer learning between different EEG datasets or the DCNet framework to self-supervised learning.

In the SupCLR framework, more experiments on softmax cross-entropy multilabel loss for larger batch size (4096) and larger categorical datasets is necessary. Another interesting direction is applying multilabel loss function with no labels(self-supervised learning). Next step, we will use DCNet to convert the 1D signal to 2D feature space and then use SupCLR to train 1D data.

Appendix A

List of Abbreviations

SSL	self-supervised Learning
LSE	log sum exponent
MLP	multilayer perceptron
t-SNE	t-distributed stochastic neighbor embedding
EEG	electroencephalogram
ECG	electrocardiogram
GSR	alvanic skin response
OXY	oxygen saturation
HR	heart rate
EMG	electromyogram
RV	respiratory volume
SKT	skin temperature
SKC	skin conductance
BVP	blood volume pulse
FM	frontal-midline
PCC	Pearson correlation coefficient
PLV	phase locking value
TE	transfer entropy
CNN	convolutional neural network

SVM	support vector machine
DE	differential entropy
PSD	power spectral density
LDS	linear dynamic system
DCNet	deconvolution-and-convolution Network
ELM	extreme learning machine

Appendix B

Softwares Used

B.1 Overview

This section presents the python modules and essential tools used to perform this thesis's research.

B.2 Package Used

Some important python modules are given below

numpy

This module is used to process and organize some primary data and matrix calculations by CPU.

scikit-learn

This module is used as a toolbox for traditional statistics and machine learning. We used it to calculate the confusion matrix, normalize the EEG signal and calculate the T-SNE algorithm.

keras

This module is used to build more sample deep learning structures compared to Tensorflow (high-level API of Tensorflow). We used this module to build DCNet.

matplotlib

This module is a plotting library in python. We used it to generate some experiment results.

tensorflow

This library is a software library for machine learning (lower level than keras). It included keras library. We used this library to implement SupCLR.

tensorflow_datasets

This is a dataset collection in tensorflow. This library automatically load tensorflow supported datasets for the experiments. We used this library to load CIFAR10 and CIFAR100 in chapter 3.

B.3 software and tools used

Some important software or tools are listed below.

tensorboard

Tensorboard is a visualization tool that allows us to see Tensorflow experiment results on the webpage. We also used this tool to output our results, such as Fig. 3.8, Fig. 3.10, Fig. 3.11, etc.

Google colab

The google colab is a web-based interactive python platform with built-in machine learning modules configured. We used it to implement EEG preprocess stage step-by-step.

Github

We used Github to fork SimCLR [10] source code and implement SupCLR based on that. We also use Github to update our local code on remote server.

Vim

Vim is a powerful text editor we used on remote server. We used it to edit remote experiment scripts and make small changes on the code.

PyCharm

PyCharm is the IDE we used to develop and debug our code.

Appendix C

Values of Properties

System Properties	Value
System	Windows 10 & Windows 11 Pro
Processor	Intel i7-11700K
Random Access Memory (RAM)	32 GB
System Type	64-bit OS, x64-based processor
GPU	NVIDIA GeForce RTX 3080 Ti
Graphic Memory	12 GB

Table C.1: **System Properties for the EEG SEED, SEED-V datasets and SupCLR projection head experiment**

System Properties	Value
System	Ubuntu 18.04.6 LTS
Processor	Intel Xeon Processors
Random Access Memory (RAM)	32 GB
System Type	64-bit OS, x64-based processor
GPU	4 * NVIDIA GeForce RTX 6000
Graphic Memory	4 * 24 GB

Table C.2: **System Properties for the DREAMER datasets and SupCLR all other experiments**

Bibliography

[1] Geoffrey L Ahern and Gary E Schwartz. Differential lateralization for positive and negative emotion in the human brain: EEG spectral analysis. *Neuropsychologia*, 23(6):745–755, 1985.

[2] Abdullah H. Al-Shabili and Ivan Selesnick. Positive sparse signal denoising: What does a cnn learn? *IEEE Signal Processing Letters*, 29:912–916, 2022.

[3] Michela Balconi and Claudio Lucchiari. Consciousness and arousal effects on emotional face processing as revealed by brain oscillations, a gamma band analysis. *International Journal of Psychophysiology*, 67(1):41–46, 2008.

[4] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? 2017.

[5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. Training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[6] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44:291–302, 9 2020.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020-December, 5 2020.

[8] Xin Chai, Qisong Wang, Yongping Zhao, Yongqiang Li, Dan Liu, Xin Liu, and Ou Bai. A fast, efficient domain adaptation technique for cross-domain electroencephalography(EEG)-based emotion recognition. *Sensors*, 17(5), 2017.

[9] Guillaume Chanel, Joep J. M. Kierkels, Mohammad Soleymani, and Thierry Pun. Short-term emotion assessment in a recall paradigm. *International Journal of Human-Computer Studies*, 67(8):607–627, Aug 2009.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[11] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. 3 2020.

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. 11 2020.

[13] Juan Cheng, Meiyao Chen, Chang Li, Yu Liu, Rencheng Song, Aiping Liu, and Xun Chen. Emotion recognition from multi-channel EEG via deep forest. *IEEE Journal of Biomedical and Health Informatics*, 25(2):453–464, 2021.

[14] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, 10 2018.

[15] R. Duan, J. Zhu, and B. Lu. Differential entropy feature for EEG-based emotion classification. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 81–84, Nov 2013.

[16] Zhongke Gao, Xinmin Wang, Yuxuan Yang, Yanli Li, Kai Ma, and Guanrong Chen. A channel-fused dense convolutional network for eeg-based emotion recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 13:945–954, 12 2021.

[17] Naresh Singh Gaurav Menghani. Efficientdl. `https://github.com/EfficientDL/book`, 2021.

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 3 2018.

[19] Shu Gong, Kaibo Xing, Andrzej Cichocki, and Junhua Li. Deep learning in eeg: Advance of the last ten-year critical period. *IEEE Transactions on Cognitive and Developmental Systems*, 14:348–365, 6 2022.

[20] Khadidja Gouizi, Fethi Bereksi Reguig, and Choubeila Maaoui. Analysis physiological signals for emotion recognition. *7th International Workshop on Systems, Signal Processing and their Applications, WoSSPA 2011*, pages 147–150, 2011.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. 6 2020.

[22] S. K. Hadjidimitriou and L. J. Hadjileontiadis. Toward an EEG-based recognition of music liking using time-frequency analysis. *IEEE Transactions on Biomedical Engineering*, 59(12):3498–3510, Dec 2012.

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. 11 2019.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[25] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[27] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. 2:985–990 vol.2, 2004.

[28] John J.B.Allen, James A.Coan, and Maria Nazarian. Issues and assumptions on the road from raw signals to metrics of frontal EEG asymmetry in emotion. *Biological Psychology*, 67(1):183–218, 2004.

[29] R. Jenke, A. Peer, and M. Buss. Feature extraction and selection for emotion recognition from EEG. *IEEE Transactions on Affective Computing*, 5(3):327–339, July 2014.

[30] Haoning Kan, Jiale Yu, Jiajin Huang, Zihe Liu, and Haiyan Zhou. Self-supervised group meiosis contrastive learning for eeg-based emotion recognition. 7 2022.

[31] Stamos Katsigiannis and Naeem Ramzan. Dreamer: A database for emotion recognition through EEG and ecg signals from wireless low-cost off-the-shelf devices. *IEEE Journal of Biomedical and Health Informatics*, 22(1):98–107, 2018.

[32] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. 2017.

[33] Pascal Koiran and Eduardo D Sontag. Neural networks with quadratic vc dimension. *Journal of Computer and System Sciences*, 54(1):190–198, 1997.

[34] D. D. Langleben, L. Schroeder, J. A. Maldjian, R. C. Gur, S. McDonald, J. D. Ragland, C. P. O'brien, and A. R. Childress. Brain activity during simulated deception: An event-related functional magnetic resonance study. *NeuroImage*, 15:727–732, 3 2002.

[35] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature 2015 521:7553*, 521:436–444, 5 2015.

[36] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1681:319–345, 1999.

[37] Seung Bo Lee, Hyun Ji Kim, Hakseung Kim, Ji Hoon Jeong, Seong Whan Lee, and Dong Joo Kim. Comparative analysis of features extracted from eeg spatial, spectral and temporal domains for binary and multiclass motor imagery classification. *Information Sciences*, 502:190–200, 10 2019.

[38] Dongdong Li, Bing Chai, Zhe Wang, Hai Yang, and Wenli Du. EEG emotion recognition based on 3-d feature representation and dilated fully convolutional networks. *IEEE Transactions on Cognitive and Developmental Systems*, 13(4):885–897, 2021.

[39] Jinpeng Li, Hao Chen, and Ting Cai. Foit: Fast online instance transfer for improved eeg emotion recognition. *Proceedings - 2020 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2020*, pages 2618–2625, 12 2020.

[40] Jinpeng Li, Shuang Qiu, Changde Du, Yixin Wang, and Huiguang He. Domain adaptation for eeg emotion recognition based on latent representation similarity. *IEEE Transactions on Cognitive and Developmental Systems*, 12:344–353, 6 2020.

[41] T. Li, W. Liu, W. Zheng, and B. Lu. Classification of five emotions from EEG and eye movement signals: Discrimination ability and stability over time. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 607–610, March 2019.

[42] Yang Li, Wenming Zheng, Zhen Cui, Tong Zhang, and Yuan Zong. A novel neural network model based on cerebral hemispheric asymmetry for EEG emotion recognition. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1561–1567. ijcai.org, 2018.

[43] Yang Li, Wenming Zheng, Lei Wang, Yuan Zong, and Zhen Cui. From regional to global brain: A novel hierarchical spatial-temporal neural network model for eeg emotion recognition. *IEEE Transactions on Affective Computing*, 13:568–578, 2022.

[44] Y. Lin, C. Wang, T. Jung, T. Wu, S. Jeng, J. Duann, and J. Chen. EEG-based emotion recognition in music listening. *IEEE Transactions on Biomedical Engineering*, 57(7):1798–1806, July 2010.

[45] Yuan Pin Lin, Chi Hong Wang, Tien Lin Wu, Shyh Kang Jeng, and Jyh Horng Chen. Eeg-based emotion recognition in music listening: A comparison of schemes for multiclass support vector machine. *ICASSP, IEEE International*

Conference on Acoustics, Speech and Signal Processing - Proceedings, pages 489–492, 2009.

[46] Wei Liu, Jie Lin Qiu, Wei Long Zheng, and Bao Liang Lu. Comparing recognition performance and robustness of multimodal deep learning models for multimodal emotion recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 14:715–729, 6 2022.

[47] Wei Liu, Wei-Long Zheng, and Bao-Liang Lu. Multimodal emotion recognition using multimodal deep learning. *CoRR*, abs/1602.08225, 2016.

[48] Yifei Lu, Wei-Long Zheng, Binbin Li, and Bao-Liang Lu. Combining Eye Movements and EEG to Enhance Emotion Recognition. *International Joint Conferences on Artificial Intelligence*, pages 1170–1176, 2015.

[49] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11206 LNCS:185–201, 5 2018.

[50] D. Mantini, M. G. Perrucci, C. Del Gratta, G. L. Romani, and M. Corbetta. Electrophysiological signatures of resting state networks in the human brain. *Proceedings of the National Academy of Sciences*, 104(32):13170–13175, 2007.

[51] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better; efficient deep learning: A survey on making deep learning models smaller, faster, and better.

[52] Seong Eun Moon, Chun Jui Chen, Cho Jui Hsieh, Jane Ling Wang, and Jong Seok Lee. Emotional eeg classification using connectivity features and convolutional neural networks. *Neural Networks*, 132:96–107, 12 2020.

[53] Klaus Robert Müller, Michael Tangermann, Guido Dornhege, Matthias Krauledat, Gabriel Curio, and Benjamin Blankertz. Machine learning for real-time single-trial eeg-analysis: from brain-computer interfacing to mental state monitoring. *Journal of neuroscience methods*, 167:82–90, 1 2008.

[54] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9910 LNCS:69–84, 3 2016.

[55] Pierre H Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. Byol works even without batch statistics. 2020.

[56] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain 1. *Psychological Review*, 65:19–27, 1958.

[57] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature 1986 323:6088*, 323:533–536, 1986.

[58] Daniela Sammler, Maren Grigutsch, Thomas Fritz, and Stefan Koelsch. Music and emotion: Electrophysiological correlates of the processing of pleasant and unpleasant music. *Phychopysiology*, 44(2):293–304, 2007.

[59] Michela Sarlo, Giulia Buodo, Silvia Poli, and Daniela Palomba. Changes in EEG alpha power to different disgust elicitors: the specificity of mutilations. *Neuroscience Letters*, 382(3):291–296, 2015.

[60] Alexandre Schaefer, Frédéric Nils, Xavier Sanchez, and Pierre Philippot. Assessing the effectiveness of a large database of emotion-eliciting films: A new tool for emotion researchers. *Cognition and Emotion*, 24(7):1153–1172, 2010.

[61] L. Shi, Y. Jiao, and B. Lu. Differential entropy feature for EEG-based vigilance estimation. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6627–6630, July 2013.

[62] L. Shi and B. Lu. Off-line and on-line vigilance estimation based on linear dynamical system and manifold learning. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 6587–6590, Aug 2010.

[63] Lin Shu, Jinyan Xie, Mingyue Yang, Ziyi Li, Zhenqi Li, Dan Liao, Xiangmin Xu, and Xinyi Yang. A review of emotion recognition using physiological signals. *Sensors*, 18(7), 2018.

[64] Siddharth Siddharth, Tzyy-Ping Jung, and Terrence J. Sejnowski. Utilizing deep learning towards multi-modal bio-sensing and vision-based affective computing. *IEEE Transactions on Affective Computing*, pages 1–1, 2019.

[65] T. Song, W. Zheng, P. Song, and Z. Cui. EEG emotion recognition using dynamical graph convolutional neural networks. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[66] Lech Szymanski and Brendan McCane. Deep networks are effective encoders of periodicity. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10):1816–1827, 2014.

[67] Ilya Tolstikhin, Olivier Bousquet, Bernhard Schölkopf, Konstantin Thierbach, Pierre Louis Bazin, Walter de Back, Filippos Gavriilidis, Evgeniya Kirilina, Carsten Jäger, Markus Morawski, Stefan Geyer, Nikolaus Weiskopf, Nico Scherf, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Elon Musk, Neuralink, Martin A Hjortsø, Peter Wolenski, Sebastian Ruder, Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, David Duvenaud, and Carl Doersch. Generative adversarial networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11046 LNCS:1–9, 6 2014.

[68] Gaetano Valenza, Luca Citi, Antonio Lanatá, Enzo Pasquale Scilingo, and Riccardo Barbieri. Revealing real-time emotional responses: a personalized assessment based on heartbeat dynamics. *Scientific Reports 2014 4:1*, 4:1–13, 5 2014.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December:5999–6009, 6 2017.

[70] Fei Wang, Shichao Wu, Weiwei Zhang, Zongfeng Xu, Yahui Zhang, Chengdong Wu, and Sonya Coleman. Emotion recognition with convolutional neural network and EEG-based efdms. *Neuropsychologia*, 146:107506, 2020.

[71] Wanhui Wen, Guangyuan Liu, Nanpu Cheng, Jie Wei, Pengchao Shangguan, and Wenjin Huang. Emotion recognition based on multi-variant correlation of

physiological signals. *IEEE Transactions on Affective Computing*, 5:126–140, 2014.

[72] Xun Wu, Wei-Long Zheng, and Bao-Liang Lu. Investigating EEG-based functional connectivity patterns for multimodal emotion recognition. *ArXiv*, abs/2004.01973, 2020.

[73] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination.

[74] Y. Yang and Q. M. J. Wu. Extreme learning machine with subnetwork hidden nodes for regression and classification. *IEEE Transactions on Cybernetics*, 46(12):2885–2898, Dec 2016.

[75] Y. Yang, Q. M. J. Wu, and Y. Wang. Autoencoder with invertible functions for dimension reduction and image reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(7):1065–1079, July 2018.

[76] Y. Yang, Q. M. J. Wu, W. Zheng, and B. Lu. EEG-based emotion recognition using hierarchical network with subnetwork nodes. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):408–419, June 2018.

[77] Yimin Yang, Wandong Zhang, Jonathan Wu, Will Zhao, and Ao Chen. Deconvolution-and-convolution networks, 2021.

[78] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9907 LNCS:649–666, 3 2016.

[79] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer, and Meta Ai. Opt: Open pre-trained transformer language models. 5 2022.

[80] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Victoria Lin, Todor Mihaylov,

Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer, and Meta Ai. Opt: Open pre-trained transformer language models. 5 2022.

[81] Wandong Zhang, Q. M. Jonathan Wu, Yimin Yang, and Thangarajah Akilan. Multimodel feature reinforcement framework using moore–penrose inverse for big data analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):5008–5021, 2021.

[82] Yue Zhang, Weihai Chen, Chun Liang Lin, Zhongcai Pei, Jianer Chen, and Zuobing Chen. Boosting-lda algriothm with multi-domain feature fusion for motor imagery eeg decoding. *Biomedical Signal Processing and Control*, 70:102983, 9 2021.

[83] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[84] W. Zheng and B. Lu. Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks. *IEEE Transactions on Autonomous Mental Development*, 7(3):162–175, Sep. 2015.

[85] W. Zheng, J. Zhu, and B. Lu. Identifying stable patterns over time for emotion recognition from EEG. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[86] W. Zheng, J. Zhu, Y. Peng, and B. Lu. EEG-based emotion classification using deep belief networks. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2014.

[87] Wei-Long Zheng and Bao-Liang Lu. Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks. *IEEE Transactions on Autonomous Mental Development*, 7(3):162–175, 2015.

[88] Rushuang Zhou, Zhiguo Zhang, Hong Fu, Li Zhang, Linling Li, Gan Huang, Yining Dong, Fali Li, Xin Yang, and Zhen Liang. Pr-pl: A novel transfer learning framework with prototypical representation based pairwise learning for eeg-based emotion recognition. 2 2022.

[89] Wilhelm Ågren wagren. The nt-xent loss upper bound an upper bound for average similarity. 2022.