Feature Extraction Enhances Model Performance

by

Xiaofan Wang

Lakehead University

A thesis submitted in partial fulfillment of the
Requirements for the degree of
MASTERS
in the Department of Computer Science

September, 2025

EXAMINATION COMMITTEE

Supervisor		
Dr. Ruizhong Wei	Computer Science	Lakehead University
External Examiner		
Dr. Yimin Yang	Electrical and Computer Engineering	Western University
Departmental Examiner		
Dr. Sabah Mohammed	Computer Science	Lakehead University

Abstract

Deep learning has emerged as a prominent approach in traditional machine learning paradigms due to its superior capability for deep-level feature extraction. This, in turn, demonstrates that the efficiency, depth, and richness of feature extraction have a profound impact on model performance. Features serve as key characteristics for distinguishing objects and represent dimensionality-reduced representations of data. This paper proposes two effective models applied to EEG emotion recognition and NL2SQL tasks, respectively, which enhance model performance through optimized feature extraction.

In previous models for processing EEG signals, researchers have typically focused on only partial features of EEG while rarely integrating these features comprehensively. To address this limitation, we designed a multi-feature extraction method that improves performance by extracting and combining frequency, spatial, temporal, and global features from EEG signals. We conducted extensive experiments on the SEED and DEAP datasets, generating confusion matrices, t-SNE distributions, and brain region activation heatmaps to demonstrate the effectiveness of our model. Additionally, our method incorporates an adaptive GCN that eliminates the requirement for pre-defined adjacency matrices.

For the NL2SQL task, unlike traditional models that train from scratch, we designed a framework based on fine-tuning pre-trained BERT and conducted experiments on the WikiSQL, Academic, and Spider datasets. The results demonstrate that our model achieves superior performance compared to traditional models in clause prediction and exhibits stronger generalization capabilities, indicating that the prior knowledge embedded in pre-trained models also benefits the model's feature extraction capacity.

Contents

List of Figures

List of Tables

Publications

Xiaofan Wang, Ruizhong Wei and Yimin Yang, EEG emotion recognition based on multi-feature extraction and neural network with adaptive encoder, submitted to CASCON (35th IEEE International Conference on Colloaborative Advances in Software and Computing), 2025.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the following individuals and institutions:

First and foremost, **Dr. Ruizhong Wei**, my supervisor, who provided me with the opportunity to explore academic research. He offered invaluable guidance throughout my research endeavors. The completion of this thesis was challenging, yet through his patient instruction and encouragement, I was able to successfully accomplish this work. It has been a privilege to have the opportunity to work under his supervision.

Dr. Yimin Yang, who also provided me with valuable guidance and offered constructive suggestions regarding my academic writing.

The **Vector Institute and the Natural Sciences and Engineering Research Council of Canada (NSERC)**, for graciously providing me with scholarship funding that served as essential financial support, enabling me to cover the tuition fees as an international student.

Chapter 1

Introduction

Contents

1.1 Overview

Improving model performance remains a central concern in machine learning research, with feature extraction being a critical factor. Features serve as the key information for distinguishing between entities and must possess discriminability, expressiveness, and robustness. Discriminability enables accurate classification across different categories, expressiveness enhances the model's ability to capture high-level semantic representations, and robustness ensures stability and effectiveness under noise, perturbations, or domain shifts. Unlike traditional machine learning methods that rely heavily on handcrafted features, deep learning enables models to automatically learn feature representations from data, thus reducing the constraints of human prior knowledge. Despite its capacity to learn complex and hierarchical features, deep learning still faces challenges in effectively extracting and utilizing these representations. To address this, a wide range of architectures and mechanisms have been proposed. Recurrent Neural Networks (RNNs) model sequential dependencies[37]; Long Short-Term Memory (LSTM) networks introduce gating mechanisms to capture long-range dependencies[47]; Convolutional Neural Networks (CNNs) extract spatial features through local receptive fields[62]; and Transformers leverage self-attention to capture global contextual relationships[107]. More recently, large language models (LLMs), such as ChatGPT[20], have demonstrated impressive generalization and understanding capabilities, highlighting the vast potential of deep learning in feature extraction. This thesis investigates the impact of feature extraction on deep learning model performance, focusing on EEG-based emotion recognition and NL2SQL tasks, and explores performance improvements through multi-granular feature extraction and pretraining-finetuning strategies.

1.2 Background

1.2.1 Machine Learning and Deep Learning

Artificial Intelligence (AI) is a long-standing dream of humanity. It refers to enabling machines to possess human-like intelligence, such as the ability to learn and solve problems autonomously without specific programming. Humans have conducted extensive exploration in the field of AI. As early as the 1950s, Alan Turing proposed the concept of the Turing Test to determine whether a machine possesses intelligence. In the Turing Test, a human evaluator is asked to engage in a conversation and determine whether the other party is a machine or a human. If the evaluator cannot tell the difference, the machine is considered to have passed the Turing Test and is deemed to possess human-level intelligence. The Turing Test is widely regarded as a foundational concept of artificial intelligence. Machine learning is an important path toward achieving artificial intelligence. In order to enable machines to learn, a great deal of research has been conducted, leading to remarkable achievements. Deep learning, a subfield of machine learning, significantly increases the depth of neural networks. The relationship among AI, machine learning, and deep learning is shown in Fig. 1.1.

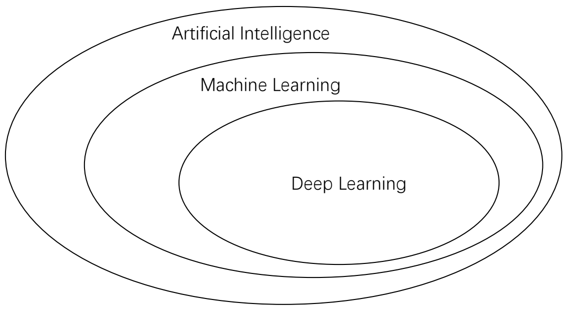

Figure 1.1: Relationship of AI, machine learning and deep learning

Machine learning encompasses various categories, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning involves training models on datasets composed of input data and corresponding labels, enabling tasks such as classification, prediction, and regression. Unsupervised learning, on the other hand, utilizes unlabeled data to perform tasks such as clustering, dimensionality reduction, and association analysis. Semi-supervised learning combines both supervised and unsupervised learning. Reinforcement learning optimizes training strategies by interacting with an environment and learning from the feedback received through rewards or penalties.

As the major branch of supervised learning and the core of deep learning, neural networks trace their origins to the perceptron, which was first introduced by Rosenblatt in the 1950s. He established the foundational rules and proved the convergence of the algorithm [92]. In 1970, Linnainmaa, in his master's thesis, introduced a reverse mode of automatic differentiation, which is now regarded as a precursor to the backpropagation algorithm [74]. Subsequently, through the continued efforts of researchers such as Werbos, Rumelhart, Hinton, and Williams, the multi-layer perceptron (MLP) was proposed, and the backpropagation algorithm was successfully applied to MLPs, enabling effective training [52]. The advancement of MLPs laid the essential groundwork for modern neural networks.

With the significant advancement in computational power—particularly the emergence of GPU computing—deep learning, a subfield of machine learning, has gradually become one of the most prominent research directions. Deep neural networks typically consist of a large number of layers, where lower layers are capable of extracting low-level and simple features from data, while higher layers can capture more complex and abstract representations. This hierarchical feature extraction endows deep learning with a representational power that far exceeds that of traditional machine learning approaches [121].

Over the past three decades, deep learning has experienced remarkable development, leading to the emergence of numerous influential models such as Recurrent Neural Networks (RNNs) [37], Convolutional Neural Networks (CNNs) [62], Long Short-Term Memory Networks (LSTMs) [47], Deep Belief Networks (DBNs) [46], Residual Neural Networks (ResNets) [45], Autoencoders [56], and the Transformer [107]. Deep learning has also been extensively applied to a wide range of tasks, including Natural Language Processing (NLP), Computer Vision (CV), autonomous driving, emotion recognition, social relationship modeling, image generation, and text generation [98]. This thesis is likewise grounded in deep learning as the primary methodological framework.

1.2.2 EEG Emotion Recognition

Understanding human emotions is a fundamental topic in psychological science, and extensive research has been conducted in this area [16, 29, 89]. A wide range of data sources contain information relevant to human emotions, such as body movements, facial expressions, text, speech, and gaze patterns. Although these signals are often intuitive and accessible, they are typically indirect and frequently contaminated by noise unrelated to emotional states. In contrast, biosignals generated by human physiological activities offer a more reliable means of emotion recognition [101]. These biosignals include electrocardiogram (ECG), heart rate (HR), blood pressure, respiration rate, electromyogram (EMG), and electroencephalogram (EEG). When neurons in the brain are active, the transmission of neurotransmitters induces postsynaptic potentials, which in turn generate electromagnetic waves. These weak electrical signals, once they pass through the skull, can be captured and recorded as EEG signals.

Electroencephalography (EEG) signals were first introduced into the field of neuroscience by Geoffrey et al. [3]. Due to their high temporal and spatial resolution, ease of analysis, low acquisition complexity, and the relatively low cost of electrophysiological recording systems, EEG signals have been widely adopted and applied in various domains [78]. With the rise of deep learning, several researchers have explored the integration of deep learning techniques with EEG data, yielding promising results [25].

EEG signals can be categorized into five frequency bands: delta (β: 1–4Hz), theta (θ: 4–8 Hz), alpha (α: 8–13 Hz), beta (β: 14–30 Hz), and gamma (γ: 30–80 Hz). The analysis of features derived from these frequency bands provides a fundamental reference for EEG-based research [68]. In addition to this, researchers have also quantified emotional states by extracting features such as the power spectral density (PSD) of each band [9], functional connectivity between different brain regions (e.g., coherence and phase-locking value) [123], and hemispheric asymmetry (e.g., frontal alpha asymmetry, which has been associated with emotional tendencies) [23].

1.2.3 Natural Language to SQL

With the widespread adoption of computers and the Internet, digitalization has permeated all sectors of society, leading to an increasing demand for data storage. Against this backdrop, database technology emerged and evolved. Among various types of databases, relational databases have become the predominant paradigm, and their management is conducted through a specialized language known as Structured Query Language (SQL).

However, the widespread adoption of database applications has created a disconnect with the reality that the majority of users are not trained in computer science. The learning curve associated with Structured Query Language (SQL) presents a barrier to effective database access. This challenge has motivated research into techniques that translate natural language (NL) into SQL queries—a technology commonly referred to as the Natural

Language Interface to Databases (NLIDB).

The construction of Natural Language Interfaces to Databases (NLIDB), specifically the task of translating natural language to SQL (NL2SQL), has traditionally relied on users providing well-structured and semantically clear natural language queries. Moreover, users were often required to have a considerable understanding of the underlying database schema [4]. However, with the continuous advancements of deep learning in the field of natural language processing (NLP) [84], researchers have increasingly turned to deep neural networks for NL2SQL implementation. A growing body of work based on deep learning has demonstrated that deep neural networks possess superior capabilities in capturing deep semantic representations and exhibit enhanced generalization performance[33, 111, 118, 128]. These models can effectively handle complex, ambiguous natural language queries without the need for extensive manual feature engineering.

The implementation approaches for NL2SQL can be broadly categorized into four main types:

1. Rule-Based Methods: These approaches rely on manually crafted rules and typically require strict syntactic constraints. Traditional implementations of NL2SQL primarily fall into this category [53, 54, 65].

2. Neural Network-Based Methods: These methods employ neural networks to translate natural language questions into SQL queries, commonly using a sequence-to-sequence framework.

3. Pretrained Language Model-Based Methods: Following the introduction of the transformer architecture [107], pretrained models such as BERT [30] and T5 [91] have emerged and achieved state-of-the-art performance in NL2SQL tasks [67, 96]. This thesis adopts an approach based on this category.

4. Large Language Model (LLM)-Based Methods: The advent of models like ChatGPT [1] has brought transformative changes to the field of natural language processing. Consequently, LLM-based methods for NL2SQL have seen a significant rise in popularity.

1.3 Related Work

1.3.1 Activation Function

Activation functions play a crucial role in neural network training, as they introduce essential nonlinearity to neural networks. Without activation functions, multi-layer neural networks would be equivalent to single-layer neural networks, since the composition of multiple linear transformations is equivalent to a single linear transformation [26]. The following provides an introduction to several commonly used activation functions.

ReLU

The Rectified Linear Unit (ReLU) is the most commonly used activation function due to its computational simplicity, straightforward differentiation, and ability to reduce computational costs. ReLU is defined as:

$$\mathrm{ReLU}(x) = \max(0, x)$$

where x represents the input value. ReLU essentially converts all negative values to zero. However, a limitation arises when neurons consistently produce negative outputs, causing the activation function to always output zero, which prevents gradient updates from occurring.

ReLU has a variant called Leaky ReLU, defined as:

$$\mathrm{LeakyReLU}(x) = \max(\alpha x, x)$$

where x is the input value and α is typically a small positive constant. Compared to ReLU, Leaky ReLU preserves negative values, thereby addressing the gradient update problem.

Sigmoid

The sigmoid function is commonly used in binary classification problems and is defined as follows:

$$\text{sigmoid}(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

where x represents the input value. Notably, the sigmoid function is frequently denoted as σ. In contrast to simple linear functions such as ReLU, the sigmoid function maps outputs to the range (0, 1), effectively transforming numerical values into probability distributions, while maintaining smooth numerical transitions. However, it suffers from the vanishing gradient problem when the absolute values of neural network outputs become excessively large. Additionally, the function is centered around $\frac{1}{2}$ rather than 0, which may result in all inputs to the subsequent layer being either positive or negative, potentially leading to bias shift during training.

Softmax

Softmax shares certain similarities with sigmoid but is commonly used for multi-class classification problems. It is defined as follows:

$$\text{softmax}(x) = \frac{e^{x_i}}{\sum\limits_{j=1}^{n} e^{x_j}}, \quad \textit{where } x_i \in x$$

Here,$x = [x_1, x_2, \ldots, x_n]$ represents a vector composed of multiple input values. In contrast to sigmoid, which accepts only a single input value, softmax can map the multi-dimensional numerical outputs of neurons into a probability distribution that sums to 1, thereby representing the probability of belonging to each class. However, similar to sigmoid, softmax also suffers from gradient vanishing and bias shift problems, and involves more complex computations.

Tanh

The hyperbolic tangent function, denoted as tanh, is defined as follows:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

where x represents the input value. Similar to sigmoid and softmax functions, tanh employs exponential functions with the natural constant e as the base to achieve numerical smoothing, mapping values to the range (-1, 1) and thereby accomplishing numerical normalization. In contrast to these functions, tanh possesses the distinctive characteristic of being zero-centered, which mitigates the bias shift problem. Consequently, tanh demonstrates superior convergence performance and enhanced training stability in numerous deep learning applications, establishing itself as a widely adopted activation function in contemporary machine learning frameworks[36].

1.3.2 Pooling

Pooling layers are a ubiquitous and fundamental component in deep neural networks [41]. By downsampling the output from the preceding layer, they substantially compress the information from the previous layer and reduce the spatial dimensions of feature maps, thereby significantly reducing computational costs and markedly alleviating overfitting.

There are three basic types of pooling layers: average pooling, max pooling, and global pooling. Each will be introduced in the following sections.

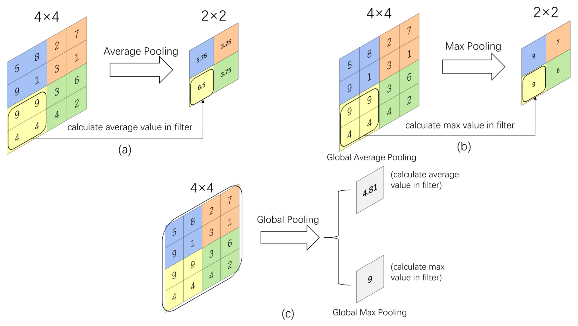
4×4
2×2
Average Pooling
calculate average value in filter
(a)
Max Pooling
calculate max value in filter
(b)
Global Average Pooling
Global Pooling
(calculate average value in filter)
4.81
(calculate max value in filter)
9
Global Max Pooling
(c)

Figure 1.2: Average Pooling, Max Pooling and Global Pooling

Average Pooling

As illustrated in Fig. 1.2(a), the pooling operation employs a rectangular window named a filter that is applied to the input matrix. The values within the window are computed to derive their collective average. The window slides across the input matrix with a specified stride, and an average calculation is performed within the window after each sliding step. Upon completing the traversal of the entire input matrix through this sliding process, the computed averages form an output matrix as the result. In Fig. 1.2(a), the input matrix has dimensions of 4×4, the filter size is 2×2, the stride is (2, 2), and the resulting output matrix dimensions are 2×2.

Max Pooling

As illustrated in Fig. 1.2(b), max pooling and average pooling are highly similar operations, with the key distinction being that average pooling computes the mean value within the filter window, whereas max pooling computes the maximum value.

Global Pooling

As illustrated in Fig. 1.2(c), global pooling can be categorized into two types: global average pooling and global max pooling. Global average pooling computes the mean value of all elements across the entire input matrix, whereas global max pooling determines the maximum value among all elements in the entire input matrix.

In addition to the three most common and fundamental pooling methods mentioned above, there exist several other approaches, including Spatial Pyramid Pooling [61], Stochastic Pooling [120], Region of Interest Pooling [42], Mixed Pooling [117], and L_p Pooling [97], among others.

1.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent one of the most fundamental and important classes of networks in deep learning. CNNs were first introduced by LeCun et al. and applied to image recognition[62], subsequently achieving remarkable breakthroughs in both computer vision and natural language processing domains, garnering widespread attention from the academic community. In general, CNNs are neural networks for processing data that has a known grid-like topology. A typical layer of a convolutional network consists of three stages (also referred as three layers): convolution stage (Affine transform), detector stage (nonlinearity) and pooling stage.

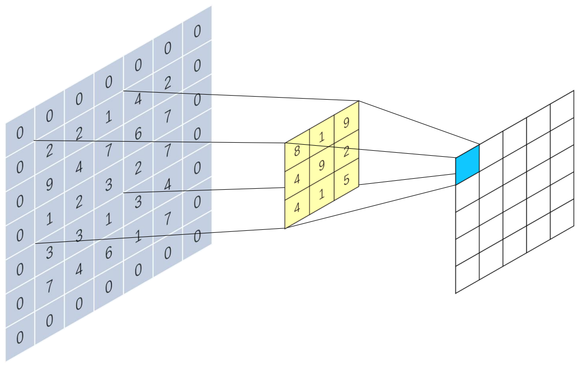
Figure 1.3: Single Convolutional kernel on a single layer to produce a feature map

The most basic convolution operation in a CNN is illustrated in Fig. 1.3, where a grayscale image with a single channel measuring 5×5 pixels is processed using a single 3×3 convolutional kernel. The convolutional kernel serves as a movable data window containing trainable weight parameters, with the region of the image covered by the kernel window referred to as the receptive field. During the convolution process, the image is padded with zeros around its perimeter to facilitate boundary processing, after which the convolutional kernel performs element-wise multiplication between the image data within its window and its corresponding weights, followed by summation to produce a single scalar output. The kernel then traverses the image systematically from the upper-left corner to the lower-right corner with a fixed stride, repeating the multiplication and summation operation at each position. The stride represents the spatial displacement of the convolutional kernel in both dimensions during each movement iteration and is typically fixed and predetermined as a hyperparameter by the practitioner during model design. The collection of scalar outputs generated at each kernel position forms a two-dimensional matrix known as a feature map, which in the example depicted in Figure X maintains dimensions of 5×5.

However, real-world scenarios are often more complex than the aforementioned example. This is because the majority of images used in machine learning are color images, which typically contain three primary color channels: RGB (Red, Green, Blue). More-

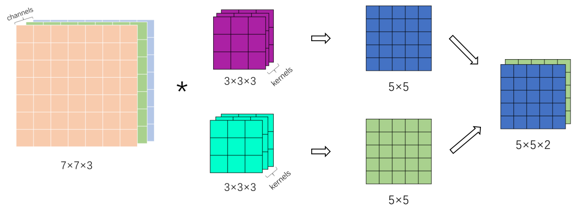

Figure 1.4: Multiple Convolutional kernels on a multi-channel image

over, to extract features as comprehensively as possible, multiple convolutional kernels are often employed rather than just one. As illustrated in Fig. 1.4, the padded image data has dimensions of $7 \times 7 \times 3$, with two convolutional kernels being utilized. Each convolutional kernel contains a number of trainable weight matrices equal to the number of image channels—three in the case of Figure Y. Each weight matrix performs convolution with its corresponding channel in the image, and the results from these weight matrix convolutions are subsequently summed to produce the output feature map. The feature maps generated by all convolutional kernels are then stacked together to form a multi-channel output tensor, which serves as the output of the convolutional layer.

1.3.4 Graph Convolutional Network

Graph data represents an important data structure. However, the spatial distribution of graph data is non-Euclidean, meaning that the local structure around each node may vary, thus lacking translational invariance in the data structure. This renders traditional CNN, which operate on regular grids, no longer applicable. Consequently, graph convolutional networks (GCNs), which are capable of performing convolution operations on graphs, have become a primary research objective for researchers in this field.

Graph Neural Networks (GNNs) serve as the prototype for Graph Convolutional Networks (GCNs), originally proposed by Gori et al. [44] and subsequently developed by Scarselli [94]. Their research introduced the fundamental concept of Neighborhood Aggregation in GNNs. A node in the graph aggregates information from its neighboring nodes and the edges connecting to these neighbors to update its own state. For a node v in a graph, where u represents its neighboring nodes and e denotes the edge connecting v and u, the state update of v and the corresponding output are governed by the following two equations:

$$s_v = f(x_v, x_e, x_u, s_u)$$
$$o_v = g(s_v, x_v)$$

Here, f is the local state transition function, and g is the local output function. $s_.$ represents the features of node v, node u, and edge e, while $s_.$ represents the states of nodes v and u. Through the stacking of multiple such neighborhood aggregation layers, GNNs enable information exchange among all nodes, thereby extracting deep-level features of the entire graph.

GCN successfully inherits the fundamental concepts of GNNs[57]. Given a graph with adjacency matrix A, degree matrix D, n nodes, and feature vector x for a particular node, GCN applies the following convolution formula at that node:

$$\Theta \ * \ x = U\Theta(\Lambda)U^T x$$
$$\Theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})$$
$$\widetilde{L} = \frac{2}{\lambda_{max} L} - I_n$$
$$L = D - A$$

Where $*$ represents the convolution operation in the graph, and Θ denotes the convolution kernel. L is the Laplacian matrix of the graph. $\widetilde{L}$ is the normalized Laplacian matrix, that is, the normalized L. λ represents the eigenvalues of the Laplacian matrix. I_n denotes the n-dimensional identity matrix. T_k represents the k-th order Chebyshev polynomial, θ_k is the coefficient value of T_k, and K is the approximation order of the Chebyshev polynomial selected by the practitioner. U is the matrix composed of eigenvectors of the normalized Laplacian matrix $\widetilde{L}$. Λ is the diagonal matrix formed by the eigenvalues of the graph Laplacian matrix L. Through such convolution operations, GCNs can effectively achieve local information aggregation and propagation over graph structures, thereby capturing complex associative relationships between nodes in the graph and enabling deep-level feature extraction from graph data.

1.3.5 Transformer

The Transformer framework was initially proposed by the Google Brain team to address the long-term dependency problem in sequences [107], and subsequently demonstrated powerful performance in natural language processing [27, 116] and computer vision [34], exerting tremendous impact on the AI field and garnering widespread attention from researchers. The transformer framework proposed in the paper is illustrated in Fig. 1.5. The Transformer primarily consists of an encoder and a decoder, both of which contain multiple layers. The encoder takes the input text after word embedding, while the decoder takes the target text after word embedding. Within each layer of both the encoder and decoder, there exists a multi-head attention module, a feed-forward neural network, as well as normalization layers and residual connections. However, the decoder contains an additional masked multi-head self-attention module compared to the encoder, which serves to mask the unpredicted portions of the target text, thereby preventing data leakage.

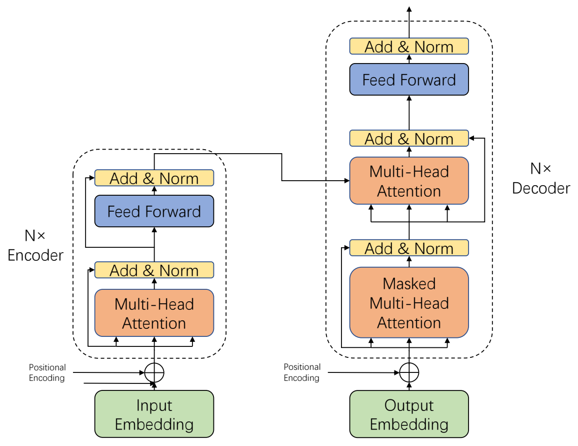

Figure 1.5: Overall Structure of Transformer

The core functionality of the Transformer lies in its multi-head self-attention mechanism. For each attention head, the input tensor is multiplied by trainable weight matrices W_Q, W_K, and W_V to obtain the query (Q), key (K), and value (V) matrices. Subsequently, the Q, K, and V matrices undergo computation according to the following formula:

$$\text{SelfAttention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

where d represents the embedding dimension, which has a value of 512 in the original paper.

In multi-head attention, the matrices W_Q, W_K, and W_V with shape d×d are typically partitioned into multiple submatrices of shape $d$head_size according to the number of heads. As illustrated in Fig. 1.6, W_q is a submatrix of W_Q with shape $d$head_size, where head_size $= \frac{d}{\text{head\_number}}$. Given an input word vector x from the text matrix with length d, the output is computed as $x' = xW_q$. Since W_q is a submatrix of W_Q, it remains trainable. The same operations apply to W_K and W_V. In the paper, the head$_\text{n}$umber is set to 8, resulting in head_size $= d = d = d = d \quad = \frac{512}{} = 64$

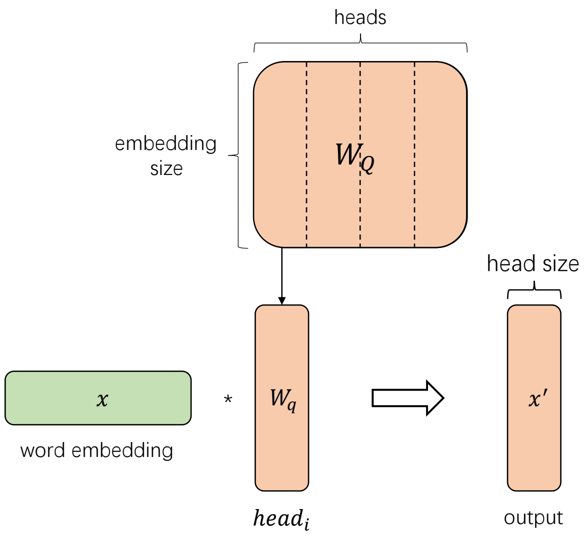

Figure 1.6: Process of Multi-Head Attention

At the conclusion of multi-head attention, the outputs from each head are concatenated and multiplied by a trainable weight matrix W_O to fuse the multi-head information and generate the final result, as illustrated in the following formula:

$$\text{MultiHead}(X) = \text{Concat}(head_1, head_2, \ldots, head_h)W_O$$

where X represents the text matrix after embedding, with dimensions of sequence_length×

d, h denotes the number of heads, and $head_i$ $\{i = 1, 2, \ldots, h\}$ represents the output of each individual head, with the self-attention computation of each head on X having been described in the preceding section.

1.3.6 BERT

Context understanding is crucial in NLP and can even be considered one of the core tasks in natural language processing. In the past, researchers commonly employed Recurrent Neural Networks (RNNs) and LSTM models. However, these models suffer from limitations such as difficulty in capturing long-term dependencies and inability to effectively leverage the parallel processing advantages of modern hardware, thereby constraining both model expressiveness and computational efficiency. The introduction of the Transformer architecture has significantly improved these issues[107]. The self-attention mechanism, combined with the encoder-decoder framework, enables the model to globally attend to complex relationships between words throughout the entire text, achieving deep modeling capabilities for long-range contexts. Due to the typically large scale of Transformer models, they are commonly employed for transfer learning, specifically through pre-training and fine-tuning paradigms. The model first undergoes large-scale training on a general foundational task (pre-training), followed by small-scale, task-specific retraining (fine-tuning). This methodology liberates researchers from the isolated learning paradigm of training from scratch, dramatically improving training efficiency while enhancing model generalization and accuracy[93].

The advent of Pre-trained Transformers has ushered in a new era for Large Language Models (LLMs). This has given rise to a series of prominent models, including Transformer-XL [27], Generative Pre-Training (GPT) [90], Cross-Lingual Language Model (XLM) [60], and XLNet[116], among others, with Bidirectional Encoder Representations from Transformers (BERT) being one of the notable representatives in this family.

BERT was proposed by Devlin et al. [30], with multiple layers of Transformer encoders connected in series as its main structure, as shown in Fig. 1.7. The training of BERT consists of two stages: the first stage is pre-training based on large-scale corpora, and the second stage is fine-tuning based on specific tasks.

The pre-training inspiration for BERT is derived from the Cloze task [103], employing

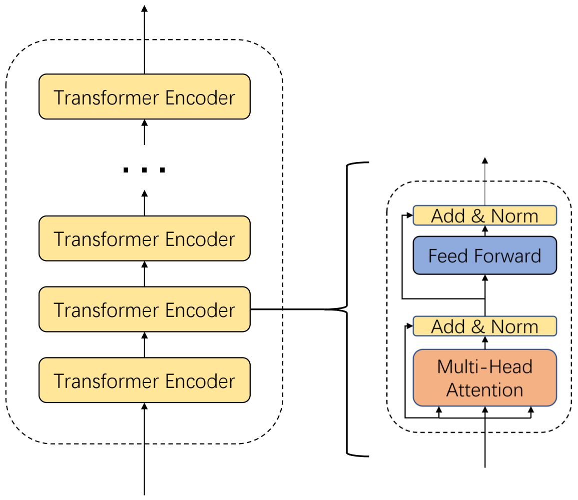

Figure 1.7: Main Structure of BERT

a masked language model (MLM) approach that randomly masks a portion of tokens in the input, enabling BERT to learn to predict the masked tokens through the remaining text. Unlike the left-to-right sequential prediction of Transformers, BERT can perform token prediction from arbitrary directions and positions, thus BERT is referred to as a bidirectional Transformer. Due to this unsupervised training design, the construction of training dataset of BERT can eliminate substantial manual annotation costs, thereby making pre-training on large-scale corpora feasible, such as BookCorpus [130]. During the fine-tuning phase following pre-training completion, practitioners can incorporate additional output layers to conduct supervised training on the BERT model, leveraging the extensive foundational knowledge acquired through pre-training to efficiently accomplish specific NLP tasks.

BERT comprises two model variants: BERT-base and BERT-large. BERT-base consists of 12 Transformer encoder layers with a hidden size (token embedding dimension) of 768, 12 multi-head self-attention heads, and a total of 110 million parameters. BERT-large, in contrast, employs 24 encoder layers, a hidden size of 1024, 16 attention heads, and contains a total of 340 million parameters.

1.4 Research Objective

Deep learning, a branch of machine learning, has experienced remarkably rapid development over the past two decades. This advancement can be attributed not only to the rapid enhancement of computational power in chips (particularly the widespread adoption of GPU-based training), which has made deeper and larger deep learning models feasible, but more importantly to the advantages that deep learning possesses over traditional machine learning approaches that are difficult to match. Specifically, deep learning can extract complex, deep-level features from data, thereby acquiring powerful representational capabilities and achieving remarkable results across various tasks [25, 34, 84]. The emergence of large-scale models represents yet another manifestation of deep learning's potential. Through deeper network architectures, larger parameter scales, and training on massive datasets, these models have not only further enhanced their capacity to model complex features but have also demonstrated unprecedented generalizability and versatility [20, 45]. All of these developments indicate that the performance of deep learning models primarily depends on their feature extraction capabilities.

Building upon the aforementioned perspectives, our research aims to enhance model performance through more diverse and efficient feature extraction strategies. For EEG-based emotion recognition tasks, many existing EEG models focus primarily on extracting temporal and spatial features [6, 8, 25]. In our study, we attempt to extract as comprehensive a range of features as possible to augment the model's representational capacity. For Natural Language to SQL (NL2SQL) tasks, in contrast to training from scratch, our research endeavors to leverage the extensive knowledge embedded in pre-trained models to improve the model's text comprehension capabilities.

1.5 Contribution

In this thesis, we propose an optimization method based on multi-feature extraction to process multi-channel EEG signal data in two-dimensional shape format for emotion classification. We conducted extensive experiments on the SEED and DEAP datasets, obtaining diverse results with comprehensive analysis. Furthermore, based on the concept of pre-training, we applied the BERT model to the NL2SQL task and proposed a framework that combines pre-trained BERT with fixed template generation.

1. We propose an EEG emotion recognition method based on multi-feature extraction and neural network with adaptive mechanisms (MENNA). The model achieves superior performance compared to mainstream approaches on both SEED and DEAP datasets, attaining 95.87% accuracy on SEED dataset and 94.42% (valence dimension) and 93.67% (arousal dimension) on DEAP dataset.

2. For MENNA, we conducted extensive experiments on the SEED and DEAP datasets, generating t-SNE visualizations and brain activation heatmap topographies. The experimental results demonstrate that different EEG features contribute variably to the model performance, with temporal and spatial features exhibiting the most substantial contributions. Moreover, distinct patterns of brain region activation were observed across different emotional states in participants.

3. In the proposed MENNA, the graph convolution in the spatial feature extraction module employs a trainable adjacency matrix, enabling the model to autonomously learn the spatial distribution of channels. This approach eliminates the requirement for practitioners to provide prior knowledge of electrode spatial configurations, thereby reducing training prerequisites and enhancing the model's adaptability.

4. We propose a fine-tuning framework based on pre-trained BERT with fixed templates (BERT with template, BERTwT) for translating natural language questions to SQL queries. We conducted experiments on the WikiSQL, Academic, and Spider datasets, comparing against the well-known SQLNet and Seq2SQL models. Our approach achieved superior performance in both accuracy and generalization capability.

5. Our research demonstrates the significant importance of feature extraction for model performance from two perspectives: multi-feature fusion and pre-training fine-tuning. This provides valuable inspiration for further researches.

1.6 Organization of Thesis

This section was all about introduction and rest of the thesis proceeds as follows, Chapter 2 introduces our proposed EEG Emotion Recognition method Based on Multi-Feature Extraction and Neural Network with Adaptive (MENNA) approach, and presents the experimental results on the SEED and DEAP datasets.

Chapter 3 introduces our proposed fine-tuning framework based on pre-trained BERT with fixed templates (BERT with template, BERTwT), and presents the experimental results on the WikiSQL, Academic, and Spider datasets.

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

EEG Emotion Recognition Based on Multi-Feature Extraction and Neural Network with Adaptive Encoder

Contents

2.1 Introduction

Emotion, as a basic and unique element in human psychological activity, plays a central role in various mental and behavioral processes such as cognition, decision making, and social interaction. Therefore, emotion analysis can help researchers better understand the subjective experiences and response mechanisms of individuals in different contexts, providing more data, materials, and theoretical support for many fields such as psychology [48], medicine [8] and human-computer interaction [19]. There are many ways to detect human emotions, which can be categorized into physiological and non-physiological signals at the signal level. Non-physiological signals include facial expressions, speech features, body movements, eye movements, and many more, while physiological signals include respiration rate, electrocardiogram (ECG), electromyogram (EMG) and electroencephalogram (EEG) [6].

Postsynaptic potentials of cortical neurons, particularly synchronous firing of pyramidal cells, generate weak electrical currents that pass through brain tissue, skull, and scalp, resulting in detectable voltage fluctuations on the scalp surface with amplitudes ranging from approximately $10 - 100\ \mu V$ [82, 83]. Electroencephalography (EEG) records these voltage signals using non-invasive electrode devices, making it a relatively convenient and low-cost method for acquiring physiological signals. As such, it has been widely employed in various emotion analysis tasks [25].

EEG signals contain a lot of feature information that can be extracted for classification or prediction tasks. In the classical EEG signal analysis workflow, researchers manually perform feature extraction to generate training data, followed by training traditional machine learning algorithms on the dataset. Therefore, the accuracy of results in such frameworks mainly depend on the design of the feature extraction method. The majority of research has focused on frequency domain features (FDFs) and time- frequency domain features (TFDFs) [85]. For example, in frequency based feature extraction, Li et al. proposed a frequency band searching method and found that the gamma band is particularly effective for emotion classification [68]. Al-Fahoum et al. computed the Power Spectral Density (PSD) across the four main frequency bands using the Fast Fourier Transformation (FFT) [5]. Duan et al. adopted Differential Entropy (DE) as a feature, demonstrating its better performance compared to traditional features [35]. In time-frequency feature extraction, Albaqami et al. used Wavelet Packet Decomposition (WPD) techniques to divide multi-channel EEG recordings into different frequency sub-bands. From these sub-bands, 6 statistical features were extracted and aggregated, followed by classification using various frameworks of the Gradient Boosting Decision Tree (GBDT) [7].

With the advancement of computational power, the emergence and development of deep learning have demonstrated its superior capability in extracting and representing higher-level features [25, 49, 51]. Consequently, an increasing number of researchers have begun to explore various forms of end-to-end deep learning models for decoding EEG signals. Schirrmeister et al. investigated a range of deep ConvNet architectures, showing that convolutional neural networks (CNNs) can achieve superior performance compared to traditional machine learning methods without the need for handcrafted feature extraction, thereby highlighting the potential of deep learning models [95]. Yang et al. constructed 3D EEG data cubes and integrated differential entropy (DE) features across multiple frequency bands using continuous convolutional networks, achieving an accuracy of 90.24% for arousal labels and 89.45% for valence labels [113]. In addition to CNNs, memory-based deep learning models such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have been employed to capture time-dependent information in EEG signals, leading to further performance improvements [100, 104]. Furthermore, due to the spatial configuration of electrodes on the human scalp during EEG signal acquisition—reflecting the functional specialization of different brain regions [73]—the extraction of spatial features has also gained increasing attention. For instance, graph convolutional networks (GCNs) have been introduced to capture spatial location information, yielding promising results across multiple datasets [39, 102].

Traditional machine learning methods heavily rely on handcrafted features, which inevitably incorporate prior human knowledge and biases, thereby imposing limitations on current scientific research [88]. In contrast, approaches that do not impose prior assumptions—allowing models to autonomously update their parameters and learn informative patterns directly from the data—are more reasonable. Moreover, EEG signals contain a wide range of diverse information, each contributing to the final predictive accuracy to varying degrees. However, many existing deep learning models tend to focus on learning only a subset of this information [25]. Therefore, leveraging as many relevant features as possible is particularly crucial for further enhancing the representational capacity and overall performance of the model.

Our contributions can be summarized as follows:

1. We propose an adaptive deep learning model, MENNA, which extracts multi-level features using multiple sub-models. Specifically, MENNA employs channel-wise attention to extract frequency features, a graph convolutional network (GCN) to extract spatial features, a convolutional neural network (CNN) to extract temporal features, and a self-attention mechanism to capture global features. The extracted features are ultimately fed into a multi-layer perceptron (MLP) for classification and prediction.

2. We conduct extensive experiments on the SEED and DEAP datasets. MENNA achieves a mean accuracy of 95.87% on the SEED dataset. On the DEAP dataset, it obtains a mean accuracy of 94.42% for the valence labels and 93.67% for the arousal labels.

3. To evaluate the contribution of each feature modality, we perform ablation studies by individually removing the frequency, spatial, temporal, and global modules from MENNA, resulting in four variant models. We compare their performance with the full MENNA model to analyze the individual impact of each feature component.

4. We visualize the experimental results to better understand the model behavior. We utilize the t-distributed stochastic neighbor embedding (t-SNE) algorithm to illustrate the distribution of encoded features. Furthermore, we apply gradient-weighted class activation mapping (Grad-CAM) to generate topographic maps, highlighting the model's attention across different brain regions under various emotional states.

The rest of this chapter is organized as follows: Section 2.2 reviews related work. Section 2.3 introduces the proposed method, including the overall architecture of MENNA and the details of each module. Section 2.4 presents the experimental results. Section 2.5 discusses our findings, and Section 2.6 concludes this chapter.

2.2 Related Techniques

2.2.1 Channel-wise Attention

The attention mechanism plays a crucial role in deep learning [18]. After its significance was recognized [13], continued exploration and expansion of attention mechanisms revealed their applicability beyond traditional contexts [79]. In particular, within computer vision tasks, attention can be effectively applied to the channel-wise information of images, a mechanism known as channel-wise attention. Chen et al. [22] highlighted the limitations of prior models that focused solely on spatial properties. By introducing channel-wise attention—assigning weights to each layer in a multi-layer feature map—and using a CNN as the backbone, their proposed SCA-CNN achieved state-of-the-art performance. This idea was further advanced by Hu et al. [50], who introduced the Squeeze-and-Excitation (SE) block. This block utilizes global average pooling to compress all data within each channel into a single representative value, thereby capturing global channel information. Their work achieved excellent results and has had a broad impact on the field.

Given that different frequency bands of the human brain are associated with distinct functional roles [59], and these bands collectively contribute to the formation of the complete EEG signal, it is reasonable to regard the various frequency bands in EEG data as analogous to different channels in an image. Accordingly, channel-wise attention can be applied to effectively extract frequency-specific information.

2.2.2 Renormalization of GCNs

Graphs are a fundamental concept in mathematics and are widely applied in computer science. Representing real-world data in the form of graphs often proves to be highly efficient [12, 69, 129]. As a relatively complex data structure, one common approach to constructing graphs is through graph embedding [15]. However, graph embeddings often struggle to capture the deeper and more intricate patterns underlying the graph structure. Moreover, when dealing with large-scale datasets, graphs tend to contain an enormous number of nodes and edges, which poses significant computational challenges.

CNN has demonstrated exceptional capabilities in processing grid-structured data within the field of deep learning [40, 43], which has inspired researchers to explore the integration of graph processing with CNN architectures. However, graphs are inherently irregular data structures, making the definition and implementation of convolution operations on graphs a significant challenge. To address this issue, Thomas N. Kipf and Max Welling introduced the concept of Graph Convolution in their seminal work [57]. They approximated spectral graph convolution using Chebyshev polynomials, thereby simplifying complex spectral operations into linear transformations involving the adjacency matrix. Each convolutional layer aggregates information only from a node's immediate (i.e., first-order) neighbors. By stacking K such layers, the model becomes capable of capturing features from neighbors up to K hops away. Additionally, they proposed a renormalization trick to alleviate the issues of vanishing and exploding gradients, enabling the construction of deeper Graph Convolutional Networks (GCNs) and enhancing model scalability. Their approach achieved significantly better performance compared to traditional methods, along with substantial improvements in computational efficiency.

Considering that EEG data is typically collected using caps that adhere to international standards, where electrodes are arranged in a spatially structured and consistent manner, and given that different regions of the human brain are functionally specialized [10], GCN can be effectively applied to EEG signals to capture and model spatial dependencies.

2.2.3 Self-Attention Mechanism

The self-attention mechanism was first introduced in the seminal 2017 paper alongside the Transformer architecture [107]. This work has had a profound impact, with the Transformer being widely adopted across tasks in natural language processing (NLP) [27, 30, 116] and computer vision (CV) [34, 77], achieving state-of-the-art performance. Moreover, most contemporary large language models (LLMs) are built upon the Transformer framework [90].

At the heart of the Transformer lies the self-attention mechanism, which simultaneously considers all tokens in a sequence [49]. After applying sinusoidal positional encoding, the encoded token matrix is projected into key and query matrices, enabling key-value querying to compute an attention score matrix. This matrix captures the pairwise relationships between each token and all others through attention weights. Meanwhile, the encoded token matrix is also projected into a value matrix, which is then multiplied by the normalized attention scores. This yields a new token representation, which is a weighted sum of the original token and all other tokens in the sequence. As the self-attention mechanism computes interactions among all tokens, it is inherently capable of modeling long-range dependencies.

To leverage the benefits of self-attention, several studies have adopted Transformer-based architectures as their backbone [63, 99], demonstrating promising results. Inspired by these works, we incorporate the self-attention mechanism into our model to explore more distant and implicit dependencies among EEG samples.

2.3 Method

2.3.1 Overview of the framework

The performance of a model is often significantly influenced by the efficiency with which it utilizes various features present in the data. In this paper, we propose a deep learning model for multi-feature extraction from EEG signals, named MENNA. This model leverages channel, spatial, temporal, and self-attention mechanisms to extract multi-dimensional information from EEG signals for the purpose of emotion classification. The end-to-end workflow of the proposed model is illustrated in Fig. 2.1. First, raw EEG signals are segmented into multiple slices, which are then divided into training and testing sets. Subsequently, the data undergo standard normalization preprocessing. After preprocessing, the data are fed into the model, which sequentially extracts frequency, spatial, temporal, and global features. Finally, a multilayer perceptron is employed to output the classification results, and model parameters are updated via backpropagation. To evaluate the performance of the proposed model, classification accuracy on the test dataset is used as the evaluation metric.

2.3.2 Preprocessing

The raw signal data can generally be represented as $X_{raw} \in \mathbb{R}^{ch \times sp}$, where ch represents the number of electrodes used to receive EEG signals, and sp represents the total number of

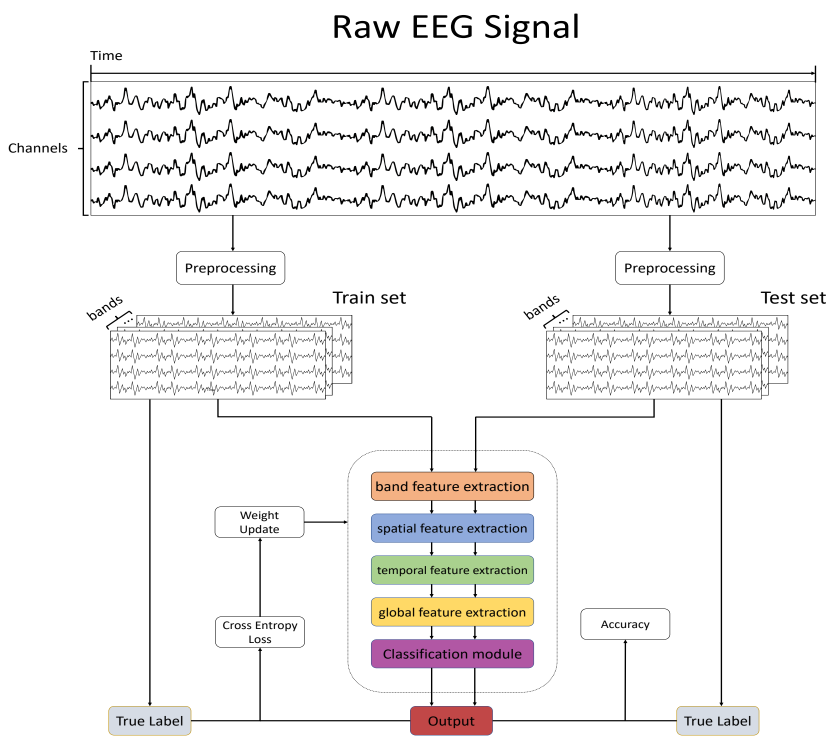

Figure 2.1: Overview of the framework of our proposed method

sampling points in a single trial.

We first apply basic band-pass filtering to the signal, specifically performing standard score normalization, as shown below:

$$\bar{X}_{out} = \frac{X_{raw} - \mu}{\sigma}$$

where $\bar{X}_{out}$ denotes the normalized signal data, μ is the mean of the signal values, and σ is the standard deviation. This normalization helps to remove noise at various frequencies.

Next, we further filter the signal into five frequency bands based on existing research [14]: delta, theta, alpha, beta, and gamma. These correspond to different mental activity states, including sleep, deep relaxation, relaxation, active, and highly active states, respectively. The filtered signal data are represented as:

$$[X_\delta, X_\theta, X_\alpha, X_\beta, X_\gamma]$$

where $X_b \in \mathbb{R}_b^{ch \times sp}, b \in \delta, \theta, \alpha, \beta, \gamma$.

Subsequently, we segment the raw data using a sliding window approach, and the segmented data can be represented as:

$$[X_1^T, X_2^T, \ldots, X_n^T], \quad X_i^T \in \mathbb{R}^{ch \times t}$$

where X_i^T represents a slice of data, and t denotes the number of sampling points within a slice. Based on previous research and for training efficiency, the time window in this study is set to 1 second.

2.3.3 Frequency Feature Extraction

After preprocessing, EEG signals are typically decomposed into multiple frequency bands (five bands) each corresponding to distinct brain wave ranges. Since the distribution of brain wave activity varies with different emotional states, capturing features across these frequency bands is crucial for effective emotion recognition. To enhance model accuracy, it is important for the model to not only extract features from each band but also to learn how to adaptively focus on the most informative frequency bands by allocating appropriate attention.

To address this, we introduce a Channel Attention (CA) mechanism, as illustrated in Fig 2.2. Originally proposed by Hu et al. [50], Channel Attention enhances the representational capacity of convolutional neural networks (CNNs) by enabling the network to adaptively weigh the importance of different channels. In our context, this mechanism allows the model to focus more effectively on the most informative frequency bands during feature extraction.

The original Channel Attention mechanism creatively incorporated RGB color channel information into the training process of convolutional neural networks, enhancing feature representation in image data. This approach gained popularity and was widely adopted across various computer vision (CV) tasks. Similarly, just as a color image consists of three distinct color channels (red, green, and blue), EEG signals can be viewed as being composed of five frequency bands as illustrated below:

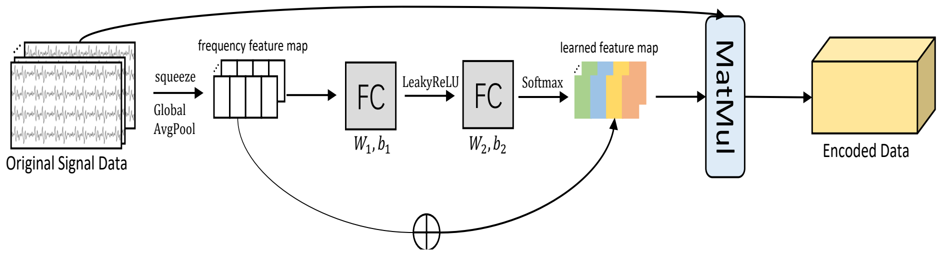

Figure 2.2: Frequency Feature Extraction

$$X^T = [X_\delta^T,\ \ X_\theta^T,\ \ X_\alpha^T,\ \ X_\beta^T,\ \ X_\gamma^T]$$

Here, $X^T \in \mathbb{R}^{channel \times time \times band}$ represents a single slice of EEG signal and $X_i^T \in \mathbb{R}^{channel \times time}$ $(i = \delta,\ \theta,\ \alpha,\ \beta,\ \gamma)$ represents data under each frequency band.

First, we pass each frequency band of X^T through a global average pool and a sigmoid function to obtain the normalized squeezed information N.

$$\begin{aligned} N &= \sigma(GlobalAveragePool(X^T)) \\ &= [\sigma(\frac{1}{channel \times time} \sum_{channel} \sum_{time} X_\delta^T), ..., \\ &\sigma(\frac{1}{channel \times time} \sum_{channel} \sum_{time} X_\gamma^T)] \end{aligned}$$

Then, to capture band-wise dependencies, we employ a Fully-Connected (FC) layer to N and use residual connections to obtain the weight parameter vector s for different bands:

$$\begin{aligned} s &= \text{ChannelAttention}(X^T) \\ &= \text{softmax}(W_2 \cdot \text{LeakyReLU}(W_1 \cdot N + b_1) + b_2) + N \end{aligned}$$

where W_1 and W_2 are the weights of the first and second fully connected layers, respectively, and b_1 and b_2 are their corresponding biases. The activation functions used are LeakyReLU and softmax. Finally, by multiplying the original data with the weight vector

s, we obtain:

$$X_s^T = WeightScale(X^T,\ s) = s \cdot X^T$$

where X_s^T represents the signal data with attention weights.

In the entire process described above, global average pooling ensures that all information within a band is considered and compressed. The sigmoid function normalizes the pooled values to facilitate subsequent processing in the fully-connected layers. The LeakyReLU activation function in the first fully-connected layer enhances the non-linear representation capability of the band feature extraction module, while the softmax activation function in the second fully-connected layer ensures that the sum of attention weights across layers is 1. Additionally, residual connections are introduced to further enhance the representation capability of the model.

2.3.4 Spatial Feature Extraction

EEG signals contain a huge amount of spatial information. This is because when an electrode cap is placed on the scalp to collect EEG signals, multiple electrodes, that is, channels in EEG signals, form a three-dimensional spatial relationship. According to the Brodmann area system, different regions of the cerebral cortex have distinct functions. For instance, BA17-BA19 areas are responsible for processing visual information, while BA41-BA42 areas are involved in auditory processing. Human emotions are a complex integration of multiple brain functions. Therefore, fully extracting and utilizing this spatial information is crucial for understanding human emotions.

To represent the spatial relationships between channels, an EEG signal sample can be viewed as a graph. Nodes in the graph represent individual channels, where the node values correspond to the voltage recorded by each channel at a given moment, while edges represent the spatial relationships between the nodes. Since different channels correspond to electrodes that measure electrical activity in different brain regions, the edges also indirectly reflect the relationships between brain regions. In graph data, edges are typically represented by an adjacency matrix A, which in this study is defined as $A \in \mathbb{R}^{channel \times channel}$.

Graph Convolutional Networks (GCNs) have demonstrated strong performance in handling graph-structured data, and some EEG-based emotion recognition studies have adopted this approach. In some of these studies, researchers often predefine an adjacency matrix

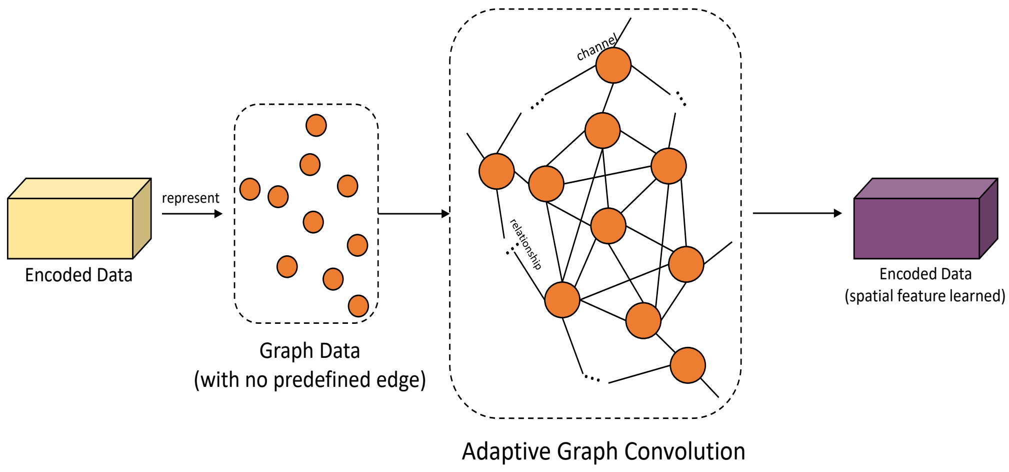
Encoded Data
represent
Graph Data
(with no predefined edge)
channel
relationship
Adaptive Graph Convolution
Encoded Data
(spatial feature learned)

Figure 2.3: Spatial Feature Extraction

before model training based on prior knowledge, such as the three-dimensional coordinates of electrodes, the importance of electrode regions, and symmetrical relationships between electrodes. However, this approach may introduce limitations from prior knowledge, potentially reducing the model's representation capability. Therefore, we introduce an adaptation mechanism to construct an adaptive graph convolution layer, allowing the model to learn the adjacency matrix autonomously and avoid the aforementioned issues.

Compared to the original definition of graph convolution, to reduce computation cost, the convolution process of a polynomial-approximation-based GCN can be expressed as

$$X = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})X$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

$$T_1(x) = x, \quad T_0(x) = 1$$

In the equations above, $T(\cdot)$ is the Chebyshev polynomial, K is the polynomial order, and $\widetilde{L}$ is the normalized Laplacian matrix, which can be given by:

$$\widetilde{L} = \frac{2}{\lambda_{max}} L - I_N$$

$$L = D - A$$

Here, L is the Laplacian matrix, A is the adjacency matrix, and D is the degree matrix. The parameter λ represents the eigenvalues of the Laplacian matrix, and θ are the learnable parameters.

However, the above method requires using a predefined adjacency matrix A, whereas this module employs an adaptive learning approach. Therefore, in this module, we set A as a trainable weight. Since D can be derived from A, and L can be computed from D and A, we treat the entire $\widetilde{L}$ as a learnable weight, denoted as $\widetilde{A}$. The original equation then simplifies to:

$$X = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L})X = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{A})X$$

In general, to improve training efficiency, we set $\theta_k = 1$ and $K = 3$. The equation can then be further simplified as:

$$\begin{aligned} X &= \sum_{k=0}^{2} T_k(\widetilde{A})X \\ &= (T_0(\widetilde{A}) + T_1(\widetilde{A}) + T_2(\widetilde{A}))X \\ &= (1 + \widetilde{A} + 2\widetilde{A}^2 - 1)X \\ &= 2\widetilde{A}^2 X + \widetilde{A}X \end{aligned}$$

As shown in Fig 2.3, after passing the input data $X \in \mathbb{R}^{channel \times time \times band}$ through the adaptive GCN, its spatial features are extracted and represented, producing an output $X \in \mathbb{R}^{channel \times time \times band}$.

2.3.5 Temporal Feature Extraction

Convolutional Neural Networks (CNNs) are frequently used in recognition and forecasting tasks, demonstrating their ability to represent temporal features. For the task of EEG emotion recognition, inspired by the approaches in related research, we adopt CNNs directly to extract temporal information.

As depicted in Fig 2.4, we apply convolution operations along the temporal dimension of the signal data using kernels with a size of $(K_t, 1)$ and a stride of $(1, 1)$.

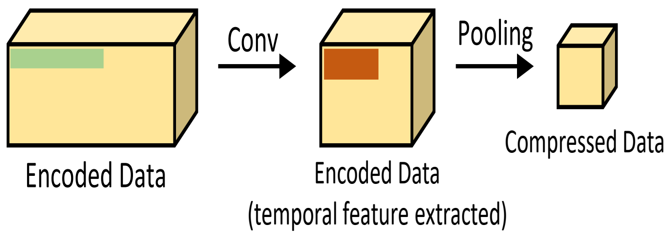

Figure 2.4: Temporal Feature Extraction

To reduce the computational complexity of subsequent modules and compress the information, we apply Average Pooling along both the spatial and temporal dimensions. The total kernel size is set to (K_{P1}, K_{P2}), with a step of $(1, 1)$, transforming the data into patches.

The above process can be represented as:

$$X = C * X$$
$$X = AvgPooling(X)$$

where C denotes the convolution operation applied to the input X, and AvgPooling represents the average pooling operation.

2.3.6 Global Feature Extraction

The self-attention mechanism proposed in Transformer possesses a global attention mechanism, which enables it to effectively extract global features. In the previous three feature extraction modules, we focused only on local information in the data while neglecting long-range dependencies. Therefore, this module utilizes multi-head self-attention (MHSA) to further learn from the temporally and spatially encoded and compressed data.

As illustrated in Fig 2.5, this module consists of K_{SA} layers of Transformer Encoder, where each layer contains three learnable weight matrices: W_Q, W_K, and W_V.

First, each patch in the compressed data X obtained from the previous steps is treated

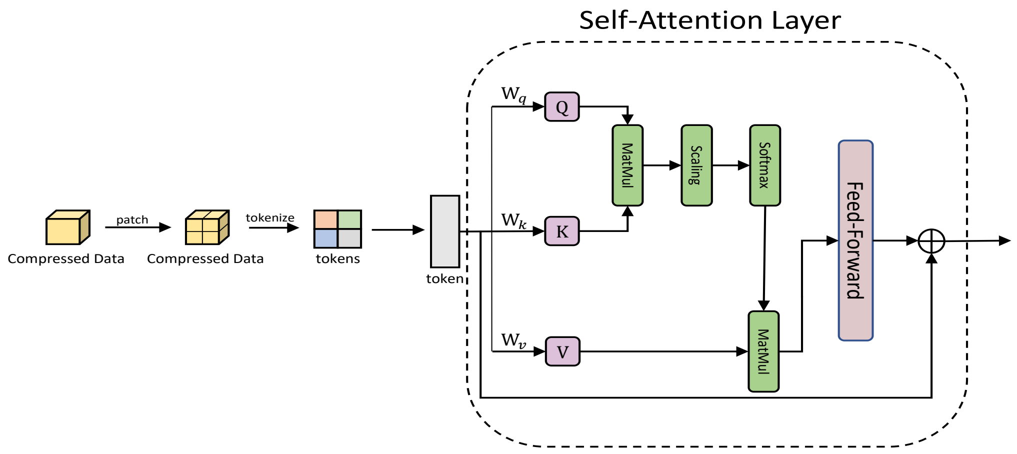

Figure 2.5: Global Feature Extraction

as a token. Then, the matrices W_Q, W_K and W_V are multiplied with the input data X to obtain the query (Q), key (K), and value (V) matrices.

Next, Q is multiplied by K, resulting in the correlation between each token and all other tokens. Finally, the product is scaled, multiplied with V, and passed through a softmax function to obtain the attention score matrix. This process is called self-attention ($SA()$) and is defined as follows:

$$\begin{aligned} SA(X) &= Softmax(\frac{QK^T}{\sqrt{d_k}})V \\ &= Softmax(\frac{(W_QX)(W_KX)^T}{\sqrt{d_k}})(W_VX) \\ &= Softmax(\frac{W_QXX^TW_K^T}{\sqrt{d_k}})(W_VX) \end{aligned}$$

where d_k represents the dimension of the embedding for each token, which in this case corresponds to the length of each patch along the band dimension.

This module also adopts the multi-head attention mechanism (MH), which is defined as follows:

$$MH(Q, K, V) = Concat(SAH_0, ..., SAH_h)$$

where

$$SAH_i = SA(Q_i, K_i, V_i),$$

$i = 0, 1, \ldots, h$

Through multi-head attention, this module can represent data from multiple perspectives, thereby further enhancing the performance of the model.

The model employs a fully connected layer to classify the encoded data. The classified data is then passed through a softmax function, producing an N-dimensional vector as output. This output represents the probability distribution of the model's predictions. The process is formulated as follows:

$$Y_{pred} = Softmax(W_{classification} X + B_{classification})$$

Finally, the predicted result Y_{pred} is compared with the true label Y_{label} using the cross-entropy loss function:

$$L = -\sum_i P_i(Y_{label}) \log(P_i Y_{pred})$$

where $P_i(\cdot)$ represents the probability distribution of either the predicted or true label. The cross-entropy loss quantifies the discrepancy between the model's predictions and the actual labels.

2.4 Experiments

2.4.1 Introduction of Datasets

SEED Dataset

The SEED dataset was created by the Brain-like Computing & Machine Intelligence (BCMI) laboratory at Shanghai Jiao Tong University [35, 124]. In the experiment for dataset construction, 15 carefully selected movie clips, each approximately 4 minutes long and with strong emotional appeal, were presented to 15 subjects for viewing.

For each subject, three experimental sessions were conducted, with each session occurring on different days. Within a session, the subject was required to watch all 15 movie clips. The process of a subject watching a complete movie clip is referred to as a “trial.” Before each trial, there was a 5-second hint period and after the movie clip ended, the subject was given 45 seconds for self-assessment and 15 seconds for rest. After watching the movie, the subject was required to immediately fill out a questionnaire to provide feedback. Additionally, two movie clips with the same emotional category were never shown consecutively. This design ensures that the emotional states elicited by each movie clip remain as independent as possible, minimizing interference between them. The experimental procedure is illustrated in the figure.

At the hardware level, the experiment employed an ESI NeuroScan System consisting of 62 AgCl electrodes, following the international 10-20 system, with an EEG signal sampling rate of 1000 Hz. The electrode impedance was maintained below 5 kΩ, and the scalp electrode distribution is shown in the figure.

The final EEG dataset consists of three folders: “EEG_raw”, ”Preprocessed_EEG” and “ExtractedFeatures”. The “EEG_raw” folder contains EEG signals collected directly from the electrodes in .cnt format, with a sampling rate of 1000 Hz. The “Preprocessed_EEG” folder contains EEG signal data for 15 subjects, with each subject having 3 sessions, resulting in a total of 45 sessions. The signals were downsampled to 200 Hz. The data for each session is stored as a .mat file, which can be opened and read using MATLAB. Each .mat file contains 15 matrices, with each matrix representing the data from a single trial. The shape of each matrix is $62 \times D_s$, where 62 corresponds to the number of electrodes (i.e., 62 channels), and D_s represents the total number of data samples obtained through downsampling for a given trial. The “ExtractedFeatures” folder contains signal data with extracted features, including differential entropy (DE) features, power spectral density (PSD) features, differential asymmetry (DASM) features, and rational asymmetry (RASM) features.

DEAP Dataset

The DEAP dataset, formally known as the Database for Emotion Analysis using Physiological Signals, was developed by a research team at Queen Mary University of London in collaboration with researchers from several other universities [58]. DEAP is a multimodal dataset designed for the analysis of human emotions. The researchers recruited 32 healthy adult participants, each of whom was asked to watch 40 one-minute-long music video clips. These clips served as emotional stimuli intended to elicit affective responses. To mitigate potential order effects, the presentation sequence of the videos was randomized for each participant.

In the experiment, EEG signals were recorded using the Biosemi ActiveTwo system, employing 32 active Ag/AgCl electrodes positioned according to the international 10-20 system standard, with a sampling rate of 512 Hz. At the beginning of the session, a 2-minute baseline recording was collected from each participant, which is typically used as a reference for subsequent data processing or baseline correction. Following this, each participant completed 40 trials. Each trial consisted of a 2-second informing phase, a 5-second baseline recording, a 1-minute video stimulus presentation, and a self-assessment phase. During the self-assessment phase, participants rated their emotional responses along four dimensions: arousal, valence, dominance, and liking. These ratings were discrete and provided on a 9-point scale, where higher values indicated greater intensity of the corresponding emotional dimension. Valence reflects the degree of pleasure, arousal represents the level of physiological activation, dominance refers to the sense of control, and liking indicates the degree of fondness or preference [80, 81].

After the experiment, the original data were downsampled to 128 Hz and band-pass filtered within the range of 4.0 to 45.0 Hz. Each trial contains a total of 63 seconds of signal data: the first 3 seconds correspond to the baseline recording, while the remaining 60 seconds capture the physiological responses during the music video presentation. The DEAP dataset is provided in two formats: a .dat format compatible with Python and a .mat format for use with MATLAB. Additional information, such as the sources of the video stimuli and participants' demographic details, is stored in the accompanying metadata folder.

2.4.2 Experiment Settings

To verify the effectiveness of the proposed method, extensive experiments were conducted on two benchmark datasets, namely SEED and DEAP. To demonstrate the validity of the multi-feature extraction strategy and evaluate the contribution of each feature type to the overall model performance, we further designed four ablation models, each removing one specific feature extraction module from MENNA. As shown in the table, these variants are: MENNAwF (MENNA without the frequency feature), which removes the frequency feature extraction module and retains only the GCN, CNN, and self-attention components; MENNAwS (MENNA without the spatial feature), which excludes the spatial feature extraction module and keeps channel attention, CNN, and self-attention; MENNAwT (MENNA without the temporal feature), which eliminates the temporal feature extraction module while preserving channel attention, GCN, and self-attention; and MENNAwG (MENNA without the global feature), which removes the global feature extraction module and retains channel attention, GCN, and CNN. In addition, we compared the proposed MENNA model against several traditional machine learning methods and recent deep learning approaches to further highlight its superior performance.

Our method is implemented with PyTorch framework in Python 3.12.0 on a NVIDIA GeForce RTX 4090 GPU, with 24 GB video memory (VRAM). We apply the Adam optimizer to train the model with the learning rate being set to 0.0001 and dropout rate being set to 0.5. The adjacency matrix in the spatial feature extraction module is randomly initialized. The convolutional kernel size in the temporal feature extraction module is 1×30 for the SEED dataset and $1{\times}20$ for the DEAP dataset, and the kernel number is set to 16. The pooling is an average 2D pooling, with the size of $1{\times}75$ and stride of 1×15. In the global feature extraction module, the number of heads h is set to 8.

2.4.3 Experiment for SEED Dataset

For the SEED dataset, we adopt a five-fold cross-validation strategy. Specifically, the signal data is divided into five equally sized subsets. In each fold, one subset is used as the test set, while the remaining four subsets will serve as the training set. This process is repeated five times, with each subset used exactly once as the test set. The final test accuracy is reported as the average accuracy across all five folds. In addition, we perform subject-independent cross-session evaluation. In this setting, experiments are conducted separately for each individual. The signal data from multiple sessions of the same subject are shuffled and evenly split into training and testing sets, ensuring that the evaluation remains subject-independent.

For the raw EEG signal data of a single trial, we apply a sliding window approach to extract segments of 1-second duration along the trial, forming the training samples. The sliding window has a step size of 1 second. Given that the sampling rate of the SEED dataset is 200 Hz, each resulting sample $X_i \in \mathbb{R}^{62\times 200}$ $(i = 1, 2, \ldots, N)$, where N represents the number of seconds in the recording of the trial.

We selected 15 different methods, including both traditional machine learning methods and deep learning models: 1) Support Vector Machine (SVM) [28]; 2) Extreme Learning Machine (ELM) [28]; 3) Logistic Regression (LR) [28]; 4) Deep Belief Networks (DBNs) [124]; 5) Adaptive Subspace Feature Matching (ASFM) [21]; 6) Dynamical Convolutional Neural Networks (DGCNN) [102]; 7) Bimodal Deep AutoEncoder (BDAE) [75]; 8) Graph Regularized Extreme Learning Machine (GELM) [125]; 9) Hierarchical Network with Subnetwork Nodes (HNSN) [115]; 10) Group Sparse Canonical Correlation Analysis (GSCCA) [126]; 11) Spatial–Temporal Recurrent Neural Network (STRNN) [122]; 12) Bi-hemispheres Domain Adversarial Neural Network (BiDANN) [71]; 13) Bi-Hemispheric Discrepancy Model (Bi-HDM) [70]; 14) Spatial and Temporal Neural Network models with Regional to Global hierarchy [72]; 15) Regularized Graph Neural Networks (RGNN) [127].

Table 2.1 presents the average accuracy of our proposed method on the three-class emotion classification task, including positive, neutral, and negative categories. The results demonstrate that each feature extraction module contributes to the overall accuracy of the model. Specifically, frequency-domain features contributed 4.44%, spatial features contributed 6.97%, temporal features contributed 11.34%, and global features contributed 3.07%. The complete model achieved an average accuracy of 95.87%. Compared with deep learning models, it outperformed the second-best model, RGNN, by 1.63%. Furthermore, it achieved significantly better performance than traditional machine learning methods.

Fig. 2.6 presents the confusion matrices illustrating the experimental results of MENNA and four additional models on the SEED dataset. Within these matrices, deeper shades of blue in the grid cells indicate higher values, while lighter shades indicate lower values. Analysis of these confusion matrices shows the following observations:

- Overall, MENNA demonstrates balanced performance across the three emotion categories. This suggests that multi-level feature extraction enables the model to achieve stable recognition performance for different emotion categories, indicating the model has robust feature integration capability.

Methods	Mean
SVM [28]	89.99
ELM [28]	82.92
LR [28]	82.70
DBNs [124]	86.08
ASFM [21]	83.51
DGCNN [102]	90.40
BDAE [75]	91.01
GELM [125]	91.07
HNSN [115]	91.51
GSCCA [126]	83.72
STRNN [122]	89.50
BiDANN [71]	83.28
Bi-HDM [70]	93.12
R2G-STNN [72]	93.38
RGNN [127]	94.24
MENNAwF	91.43
MENNAwS	88.90
MENNAwT	84.53
MENNAwG	92.80
OURS	**95.87**

Table 2.1: Mean Accuracy of Subject Dependent Cross Session Test on SEED dataset

- Specifically, MENNA exhibits its best performance for the positive emotion class, while its performance for the negative class is the worst. This result is also observed in the majority of the four additional models. The reason is likely to be that the positive emotions have more distinctive characteristics, while the negative emotions are more confusing.

- Notably, MENNAwS is the least effective in recognizing neutral emotions. This finding indicates that spatial features play an important role in the accurate recognition of neutral emotions.

Next, we visualize the results using the t-SNE algorithm to illustrate the feature distribution of the data after processing by the model. In the visualization, grey represents the positive emotional category, orange denotes neutral, and red denotes negative. As shown in Fig. 2.7, after processing by the model, the features corresponding to different emotional

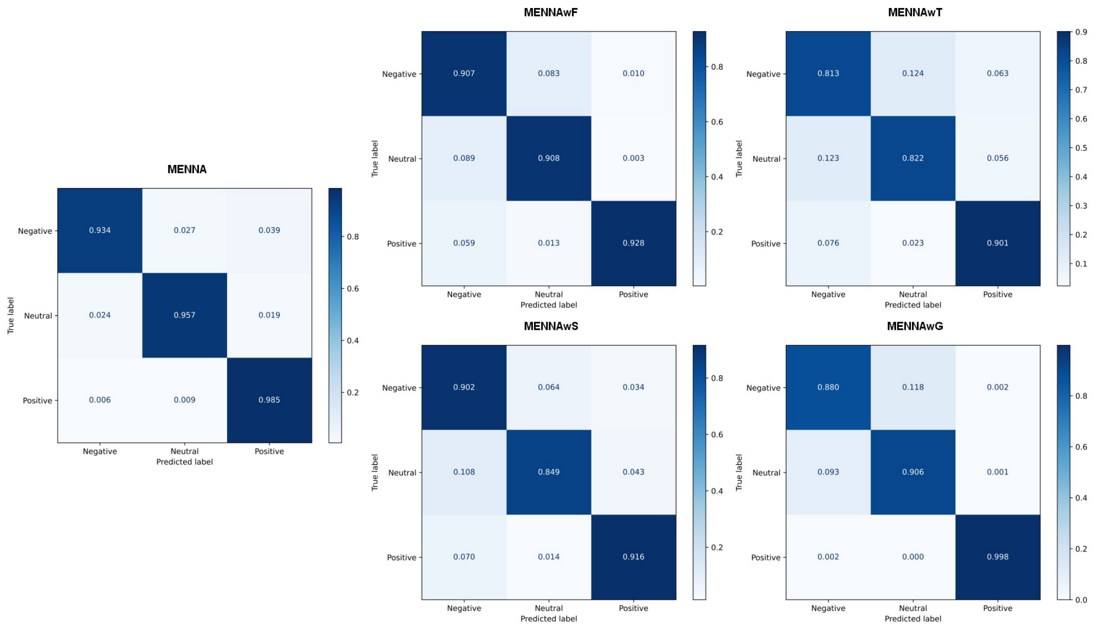

Figure 2.6: The confusion matrix of our proposed method on SEED

categories shows clear separability. Among the models, MENNA, MENNAwG, and MENNAwF demonstrate better performance, with features distributed farther apart, indicating better recognition ability. However, the features from MENNAwS and MENNAwT are relatively close to each other, suggesting weaker separability.

Finally, we employed the Gradient-weighted Class Activation Mapping (Grad-CAM) algorithm to visualize the model's representation of brain signals under different emotional states. By projecting the electrode distribution topographically, we generated heatmaps in which deeper red regions indicate higher gradient-weighted activations, while deeper blue regions indicate lower activations. As shown in Fig. 2.8, the activation patterns across different electrode positions vary depending on the emotional category. This suggests that the model attends to different brain regions when recognizing different emotions, further supporting the notion that distinct brain areas are functionally specialized for processing different emotional states.

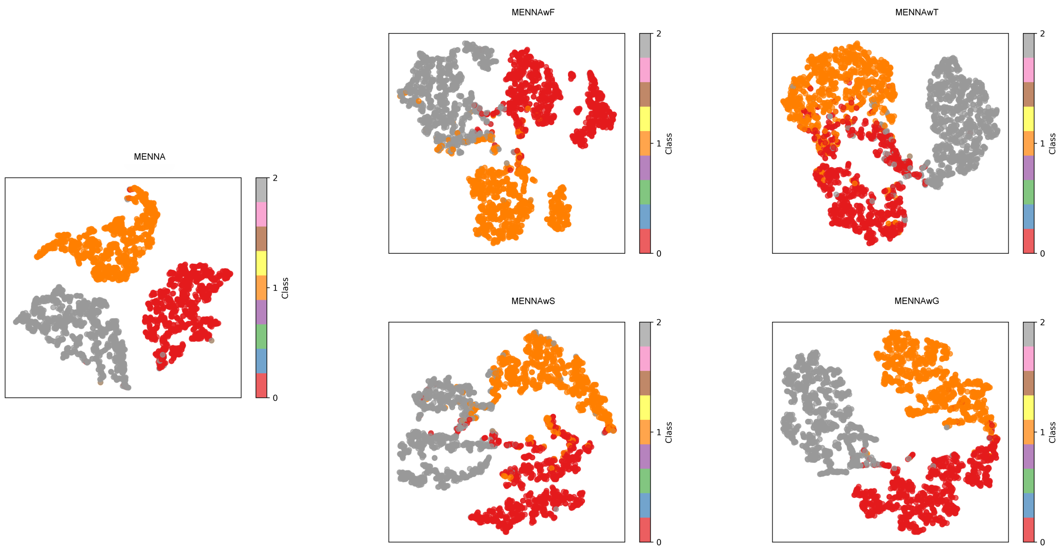

Figure 2.7: The t-SNE feature distribution of our proposed method on SEED

2.4.4 Experiment for DEAP Dataset

To evaluate the model's performance under different data shapes and label types, we conducted experiments on the DEAP dataset. The labels in the dataset are categorized into three dimensions: Valence, Arousal, and Dominance. Participants were asked to watch 40 one-minute music video clips intended to elicit emotional responses, after which they rated their experiences on a discrete 9-point scale. In our experiments, we focused on the Valence and Arousal dimensions. A subject-independent training strategy was adopted, meaning that training was conducted separately for each individual. Furthermore, for each subject, we employed a cross-validation approach to generate five train/test splits.

We chose 7 different methods, to compare with our proposed method: 1) Multilayer Perceptron (MLP) [113]; 2) Support Vector Machine (SVM) [76]; 3) Multi-Column Convolutional Neural Network (Multi-column CNN) [112]; 4) Convolutional Neural Network and Recurrent Neural Network (CNN-RNN) [114]; 5) Continuous Convolutional Neural Network (Conti-CNN) [113]; 6) Stack AutoEncoder with Long Short-Term Memory (SAE-LSTM) [110]; 7) requency Band Correlation Convolutional Neural Network (FBC-CNN) [86].

Table 2.2 presents the results of our experiments. The findings indicate that, for the Va-

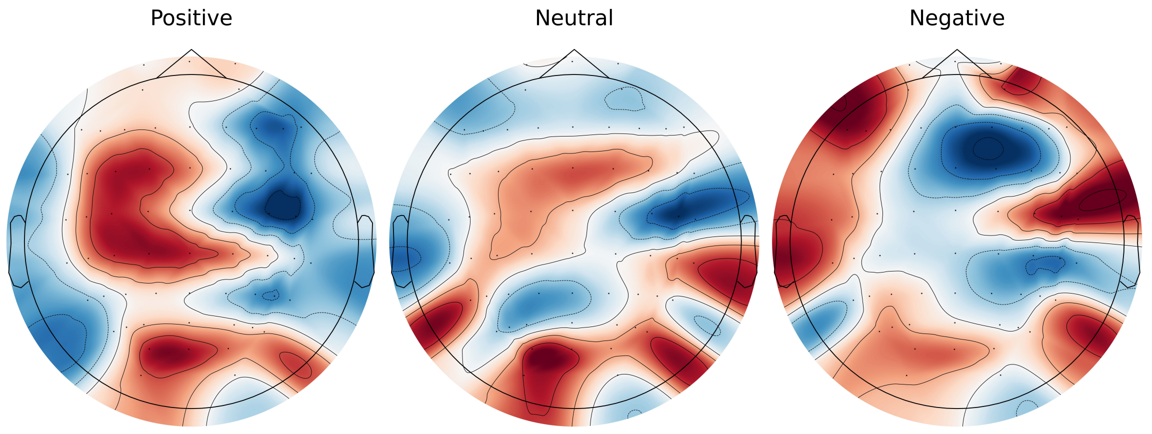

Figure 2.8: Topography of EEG signal from our proposed method on SEED

lence label, the frequency features contributed 2.6%, spatial features contributed 12.42%, temporal features contributed 8.37%, and global features contributed 3.4%. For the Arousal label, the contributions were 1.72% for frequency features, 13.52% for spatial features, 11.02% for temporal features, and 3.62% for global features. Compared to the second-best performing model, the Multi-column CNN, our proposed method achieved an improvement of 4.41% on the Valence label and 3.02% on the Arousal label.

We plotted the confusion matrices based on the experimental results. In Fig. 2.9 and Fig. 2.10, the depth of the blue color in each grid cell represents the magnitude of the corresponding value—darker shades indicate higher values. From these matrices, we can observe that:

- In both valence and arousal labels, the model demonstrates comparable performance across low and high emotional levels. However, the recognition accuracy for low-level emotions is slightly higher than that for high-level emotions. This may be attributed to the criterion used to distinguish between low and high levels: ratings greater than 5 were classified as high, while ratings less than or equal to 5 were classified as low. Given that the DEAP dataset employs a 9-point scale, the high-level category encompasses only four points (6–9), whereas the low-level category includes five points (1–5). The broader range of values in the low-level category may increase the likelihood of the model recognizing low-level emotions.

- Our model's performance on the valence label exhibited only a marginal improve-

Methods	Mean (Valence)	Mean (Arousal)
MLP [113]	87.82	88.68
SVM [76]	88.65	89.07
Multi-column CNN [112]	90.01	90.65
CNN-RNN [114]	89.92	90.81
Conti-CNN [113]	89.45	90.24
SAE-LSTM [110]	81.10	74.38
FBCCNN [86]	90.26	88.90
MENNAwF	91.82	91.95
MENNAwS	82.00	80.15
MENNAwT	86.05	82.65
MENNAwG	91.02	90.05
OURS	**94.42**	**93.67**

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

ment of 0.75% in mean accuracy compared to that on the arousal label. This indicates that the model demonstrates generalizability and stability in cross-label emotion recognition tasks.

- The models MENNAwS and MENNAwF exhibit the lowest mean accuracy, indicating that spatial and temporal features constitute the most critical information in EEG signals. Moreover, the substantial discrepancy in accuracy between the low- and high-level features within MENNAwS suggests that spatial features play a significant role in reducing the variance of recognition outcomes and enhancing the overall stability of emotion recognition performance.

In addition, we employed the t-SNE algorithm to visualize the data features processed by MENNA. In Fig. 2.11 and Fig. 2.12, the red dots represent high-level emotional states, while the gray dots indicate low-level emotional states. It can be observed that there remains a certain degree of overlap between the feature distributions of MENNAwS and MENNAwT, with red regions noticeably intruding into gray regions, and samples of the same class appearing relatively sparse. In contrast, the feature representations of MENNA, MENNAwF, and MENNAwG demonstrate better separation between classes. Data samples within the same class are more densely clustered, exhibiting enhanced connectivity and compactness.

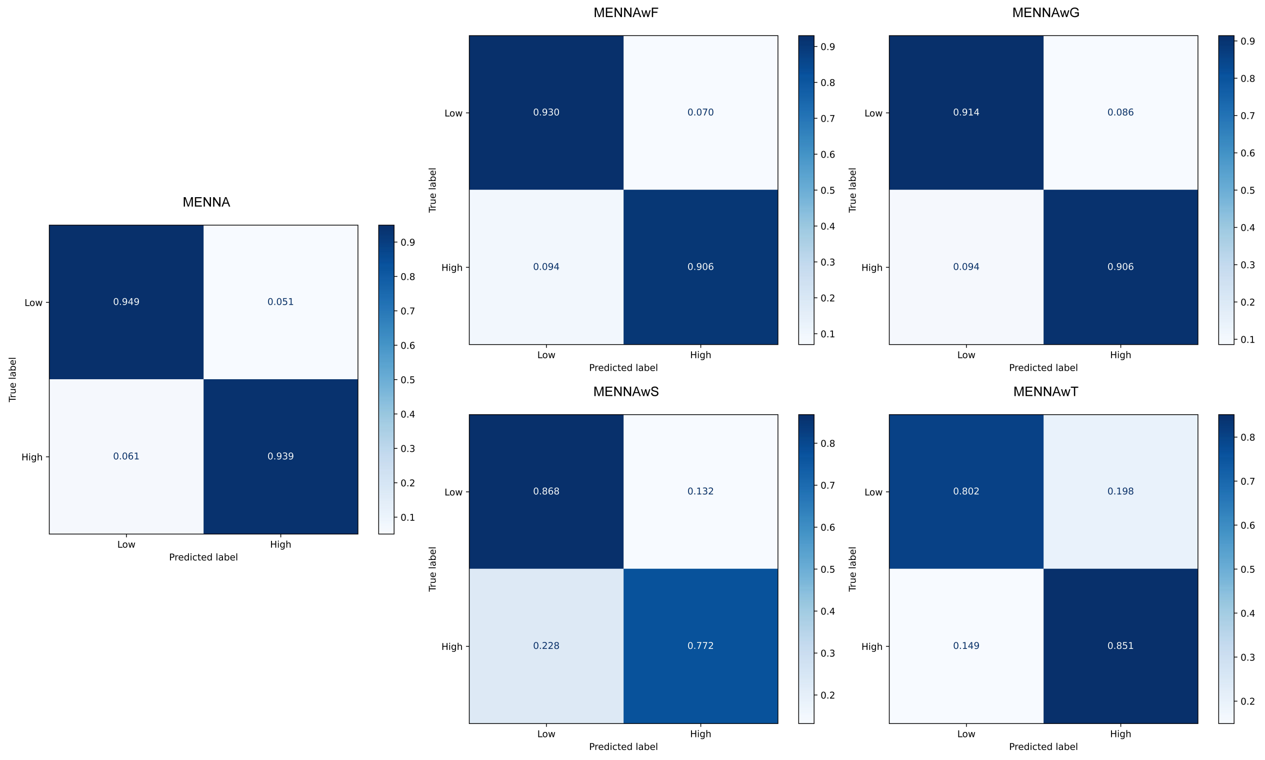

Figure 2.9: The confusion matrix of our proposed method on valence label of DEAP

Based on the topographic maps generated using Grad-CAM, as shown in Fig. 2.13, which illustrate the EEG signal activations at corresponding electrode locations, we observe that for both valence and arousal emotional dimensions, the heatmaps tend to appear predominantly blue under low emotional states, indicating lower levels of cortical activation. In contrast, during high emotional states, the maps exhibit more intense and widespread red regions, suggesting increased neural activity and broader engagement of brain areas. Notably, heightened emotional activation is often associated with increased activity in the prefrontal cortex—an area well recognized for its critical role in emotional processing [31].

2.5 Discussion

Understanding human emotions is one of the fundamental tasks in psychological science. To investigate brain activity, collecting and analyzing electroencephalogram (EEG) sig-

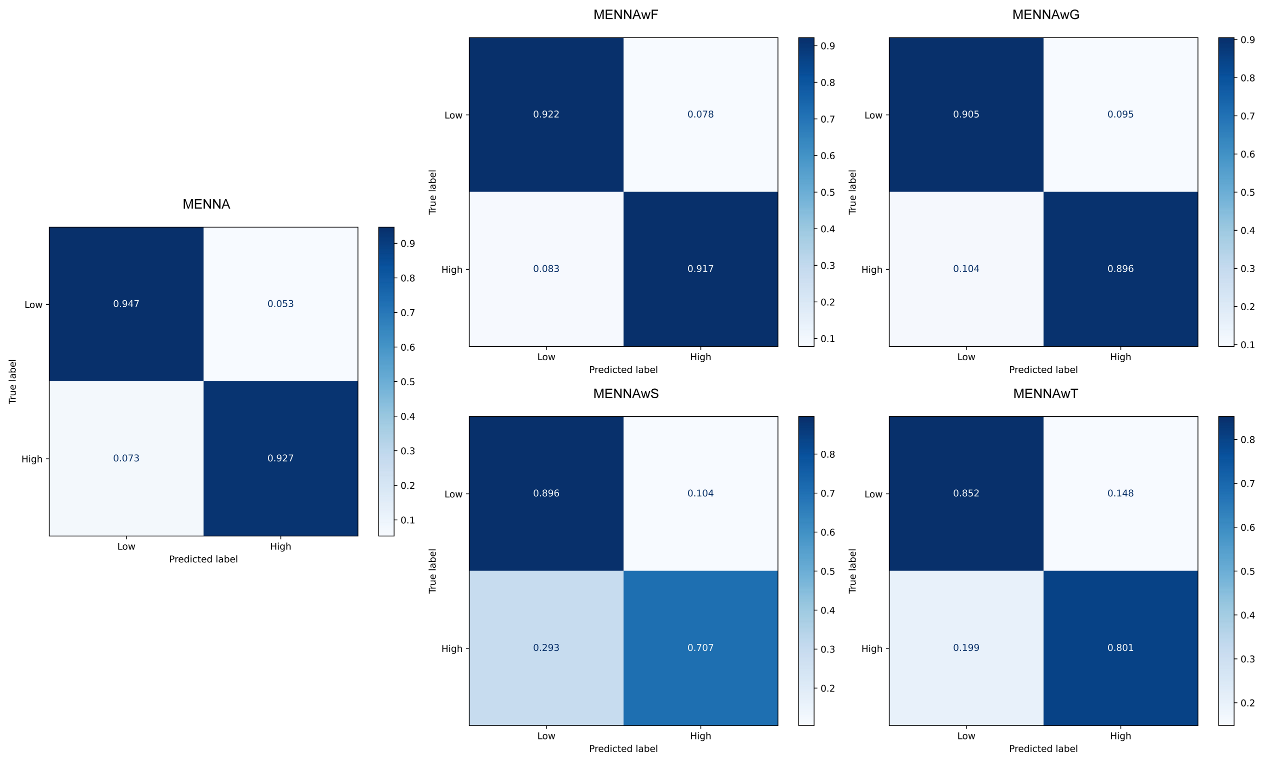

Figure 2.10: The confusion matrix of our proposed method on arousal label of DEAP

nals has become a widely adopted approach [66]. Traditional machine learning methods often struggle to extract high-level features and typically rely on hand-crafted feature engineering, which may introduce biases and limitations stemming from the researchers' prior knowledge. In recent years, the advent of deep learning has transformed this landscape. Deep learning models are capable of hierarchically integrating low-level features into high-level representations and can automatically update parameters during training. However, EEG signals are inherently rich in multi-dimensional information, while many existing studies have focused on only a subset of this information[73, 95, 100, 104, 113]. In response to this, we propose MENNA, a method designed to fully exploit the multi-level and multi-dimensional characteristics of EEG signals.

In our proposed method, we designed four distinct modules for feature extraction: the frequency feature extraction module, spatial feature extraction module, temporal feature extraction module, and global feature extraction module. The EEG signal data is sequentially passed through these modules, which encode and extract relevant features at each stage. The resulting representations are subsequently fed into a multilayer perceptron for

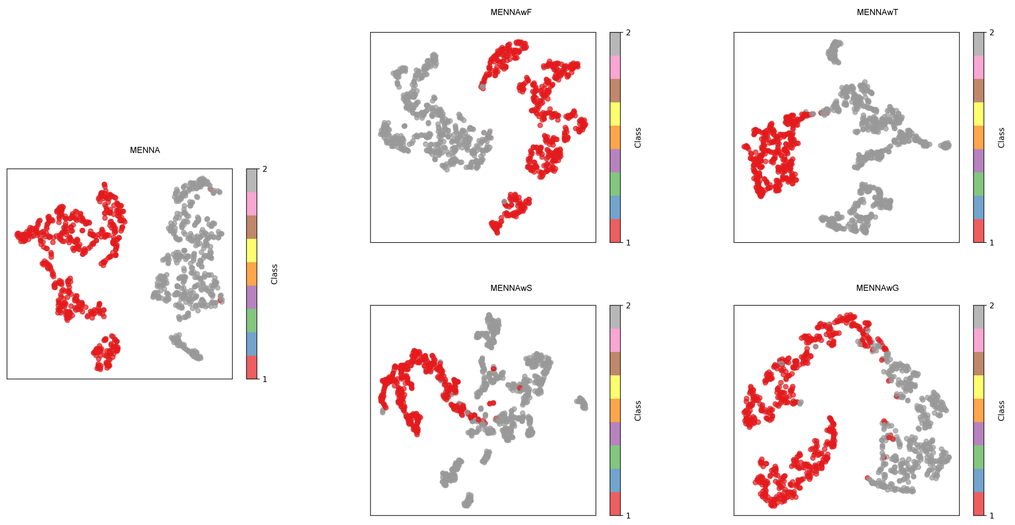

Figure 2.11: The t-SNE feature distribution of our proposed method valence label of DEAP

classification and prediction. We conducted experiments on both the SEED and DEAP datasets using 5-fold cross-validation. The experimental results demonstrate that the model achieves stable performance, with the spatial and temporal modules contributing most significantly to its effectiveness. Specifically, as shown in Figure X for the SEED dataset, the model attends to different brain regions depending on the emotional category. In summary, the integration of multiple feature types greatly enhances the model's performance by capturing complementary aspects of EEG signals.

2.6 Conclusion

This chapter proposes a deep learning model for EEG-based emotion recognition that leverages multi-feature extraction. The model integrates channel-wise attention, graph convolutional networks (GCN), convolutional neural networks (CNN), and self-attention mechanisms. It achieves a mean accuracy of 95.87% on the SEED dataset, and mean accuracies of 94.42% and 93.67% on the valence and arousal labels, respectively, of the DEAP dataset. Overall, the proposed method demonstrates advantages in EEG-based emotion recognition tasks.

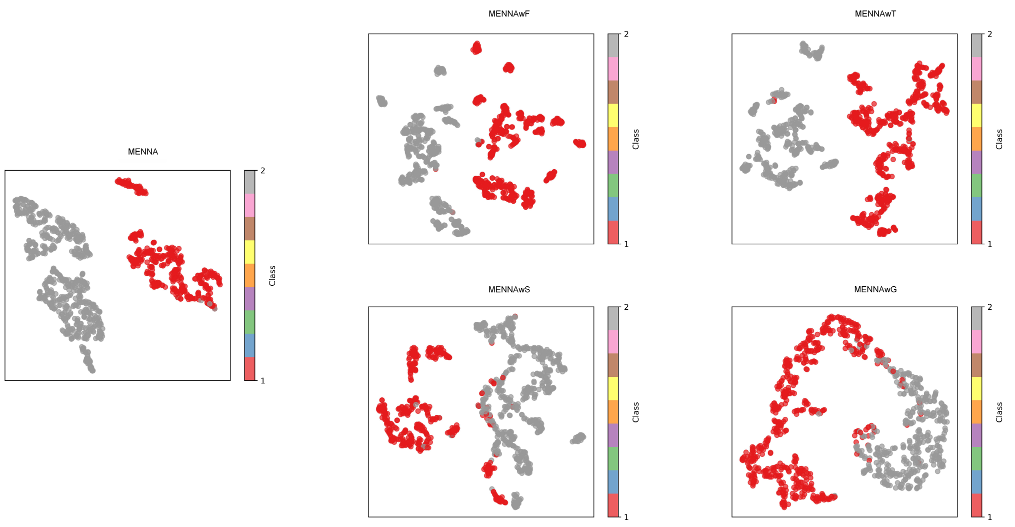

Figure 2.12: The t-SNE feature distribution of our proposed method on arousal label of DEAP

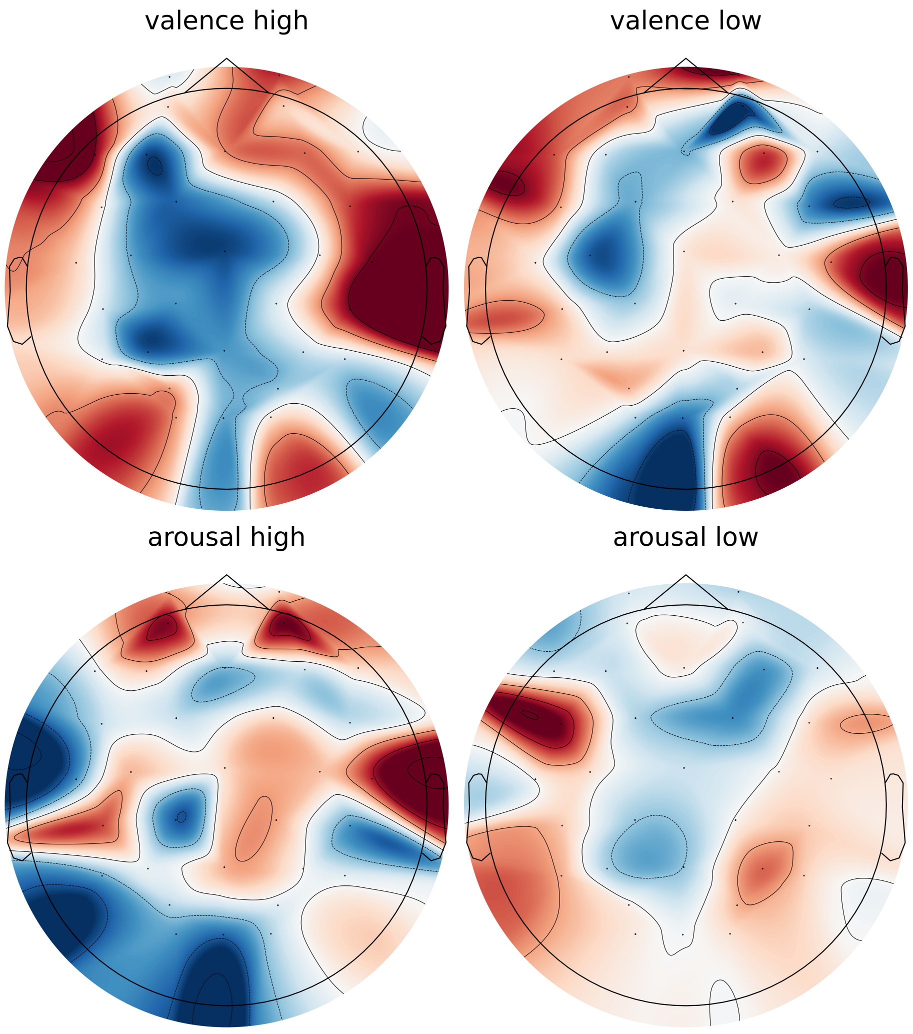

Figure 2.13: Topography of EEG signal from our proposed method on DEAP

Chapter 3

A Pre-trained BERT-based Fine-tuning method for Natural Language to SQL Query Translation

Contents

3.1 Introduction

In today's internet era, the exponential growth of data has made it a pressing issue to quickly and accurately obtain the required information from these datasets. Since Edgar Frank Codd proposed relational databases in 1970 [24], they have become the mainstream data storage model covering various aspects of society, including government, healthcare, education, business, and academia.

In the storage, query, and modification operations of relational databases, structured query language (SQL) is used as the most mainstream method. However, SQL requires users to possess certain computer and database knowledge, making it difficult for ordinary users to learn. In contrast, natural language is the most familiar and widely used language among people. Therefore, researching natural language-to-SQL technology and implementing a natural language interface to databases (NLIDB) can enable non-professionals to query databases using natural language, thereby resolving the SQL language barrier and having significant practical significance and application value.

In fact, as early as the 1970s [11], Affolter, Stockinger et al. [2] have summarized a series of rule-based methods for generating database query languages. However, considering the fuzziness of natural language concepts and grammar as well as the structural complexity of database query languages, these rule-based methods are often only applicable to limited and fixed domains with rigid table structures [17], making it difficult to adapt to today's increasingly broad and complex application scenarios.

In recent years, with the rapid development of deep learning technology, more and more people have noticed its potential in NLIDB. For example, M Uma, V Sneha [106] proposed the idea of implementing NLIDB using natural language processing (NLP) techniques in the field of natural language query. Therefore, with the emergence of more deep learning models and the continuous development and improvement of NLP technology, the implementation of NLIDB based on deep learning has gradually become a hot topic and research area.

The main content of our research is the study of database natural language interface technology based on deep learning, and a method for database natural language interface technology is proposed using BERT models and fixed template generation query sentences. More specifically, this paper draws inspiration from SQLNet's generation method based on fixed templates, and builds upon the basis of WHERE clause generation provided by SQLNet to use the pre-trained BERT language model to encode the input sequence, then through a fully connected layer and softmax, obtain the normalized probability distribution of each column aggregation function, and predict the SELECT statement. By combining the SELECT statement with the WHERE sentence provided by SQLNet, the SQL query is obtained.

Therefore, the main work of our research focuses on generating the SELECT statement, which is achieved through fine-tuning BERT and training a fully connected layer in the model. We call our proposed method as BERT with Template. In general, the working flow process proposed in this paper can be viewed as a sequence tagging problem for each column in the database table (Sequence Tagging), where the label tag is the label of the aggregation function chosen for that column, thereby achieving joint prediction of the selected columns and aggregation functions in the SELECT statement.

Our contributions can be summarized as follows:

1. We propose a fine-tuning training method based on pre-trained BERT that integrates a fixed template-based generation approach, and conducted experiments on the WikiSQL, Academic, and Spider datasets. The prediction accuracy for SELECT statements achieved 85.82%, 84.34%, and 54.48% respectively.
2. Our proposed method also demonstrates good generalizability, indicating that the pre-training fine-tuning approach enables the model to leverage extensive knowledge acquired during pre-training, thereby enhancing the model's adaptability.

The rest of this chapter is organized as follows: Section 3.2 reviews some related techniques. Section 3.3 introduces the proposed method with details. Section 3.4 presents the experimental results. Section 3.5 discusses our research, and Section 3.6 makes a conclusion.

3.2 Related Techniques

Based on deep learning, NLIDB technology has undergone rapid development and has now emerged many advanced models. Below are 2 deep learning models highly relevant to this research that implement database natural language interface technology and a brief introduction to each.

3.2.1 SQLNet

The SQLNet model proposed in 2017 [111] presented a unique solution that fundamentally addressed the problem of sequence sensitivity in traditional sequence-to-sequence architectures. The model utilized a technique based on fixed templates, which included a dependency relation network. This allowed for individual predictions to be made by only considering the dependencies on previous predictions. By integrating these unique strategies, the authors demonstrated that SQLNet's performance surpassed that of the state-of-the-art SQL generation models at the time, with improvements ranging from 9% to 13%.

Below is a brief introduction to SQLNet.

Template-Based Query Generation

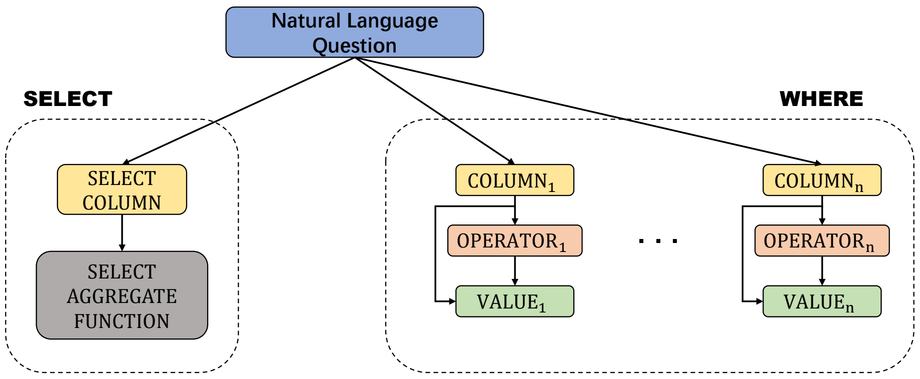

Figure 3.1: Dependencies between parts of a template

As shown in Figure Fig. 3.1, each empty space in the template corresponding to a value to be predicted is represented by a box, and directed edges indicate the dependency relationship between each empty space and its dependent item. Therefore, it can be viewed as a graph model that treats query generation as a problem of graph-based reasoning. In this way, techniques in SQLNet can completely solve the sequence problem in sequence-to-sequence models.

Column Attention-Based Sequence-to-Set Prediction

Intuitively, in a WHERE clause, the column names appearing represent a subset of all column names in the database. Therefore, the focus can be placed on predicting which column names will appear in this subset. The paper achieves this through an LSTM network, computing the column probability distribution $P_{\text{wherecol}}(\text{col} \mid \text{Q})$ using column attention, where col is the column name and Q is the query.

Training Details of SQLNet

Predicting WHERE clause - Column Prediction:
In the generation of the WHERE clause part, SQLNet needs to determine which columns should be included in the WHERE clause. The paper establishes an upper limit N for selecting the number of columns and formulates the problem of predicting column numbers as a (N-1) classification problem (from 0 to N).

Predicting WHERE clause - Operator Prediction:
Predicting the operators between columns and values in the WHERE clause, the paper proposes that this prediction can be taken as a three-class form, selecting one from $\{=, >, <\}$ as the predicted result. Then, based on the encoding vectors $E_{(\text{Q} \mid \text{col})}$ and E_{col} computed in previous steps, it is possible to implement a three-class operator prediction using column attention.

Predicting WHERE clause - Value Prediction:
The paper points out that predicting values is essentially predicting substrings from natural language query strings. Therefore, a bidirectional LSTM is used, with the decoder set to compute the probability distribution of the next token using a pointer network [108].

Predicting SELECT clause:
The SELECT clause includes predicting aggregate function operators and selecting columns. In this part, predicting column names in the SELECT clause is similar to that in the WHERE clause. Therefore, the paper computes the probability distribution of each column based on query Q using $P_{\text{SELECT col}}(i \mid \text{Q})$ and selects the column with the highest probability as the selected column.

Statement Parsing:
The paper uses the Stanford CoreNLP tokenizer to achieve this. To implement this, they use GloVe word embeddings [87] to represent each token as a one-hot vector.

3.2.2 Seq2SQL

Seq2SQL [128] is a deep learning model that can convert natural language queries into various SQL queries. The model utilizes the structural properties of SQL to construct query statements, thereby achieving significant performance improvements, increasing the state-of-the-art execution accuracy from 53.3% to 59.4%.

The model is based on the Sequence-to-SQL [32] idea and consists of three independent sentence generation modules: Aggregate Function Operators (AFO), SELECT Clause, and WHERE Clause.

AGGREGATION
First, calculate the scalar attention score, and then normalize it. Next, use multiple calculations of attention weights to obtain the probability distribution of aggregate function operators. Finally, perform a softmax operation on the probability distribution to obtain the normalized distribution.

SELECT
First, use LSTM to encode each column in the database table. Then, construct another representation for the hidden columns in the query using united weights parameters. Next, pass the column representations through a multi-layer perceptron (MLP) to compute the attention score for each column. Finally, apply a softmax function to obtain the normalized probability distribution of predicted columns.

WHERE
To avoid mismatched queries, the model uses reinforcement learning to learn an optimal policy that maximizes rewards and minimizes errors. The model returns different scores based on the validity and correctness of the parsed SQL statement, and declares a loss function to optimize the reward values.

3.3 Method

3.3.1 Task Description and Analysis

Given that SQL is fundamentally distinct from natural language, possessing stronger structural characteristics and higher grammatical requirements compared to the inherent ambiguity of natural language, direct application of simple sequence-to-sequence machine translation models is not feasible for model selection. Instead, the structural properties of SQL must be taken into consideration. Based on the aforementioned analysis, a template-based generation approach is adopted, which involves decomposing an SQL statement into the linguistic templates, as shown in Fig. 3.2.

$$SQL\ \ Query \begin{cases} SELECT \begin{cases} SELECT \\ AGGREGATE\ FUNCTION \\ COLUMN \end{cases} \\ WHERE \begin{cases} WHERE-CLAUSE \begin{cases} COLUMN \\ OPERATOR \\ VALUE \end{cases} \\ AND/OR\ OPERATOR \\ WHERE-CLAUSE \\ \ldots \end{cases} \end{cases}$$

Figure 3.2: SQL syntax structure

In Fig. 3.2, a SQL query statement can be decomposed into two primary components: the SELECT statement and the WHERE statement. The SELECT statement can be further decomposed into three constituent parts: the fixed SELECT keyword, aggregation functions, and selected columns. The WHERE statement, on the other hand, can be decomposed into multiple WHERE clauses, which are interconnected through logical operators such as AND or OR. Each individual WHERE clause can be further decomposed into its component elements: the selected column, operators (such as ==, >, <, etc.), and values. The example Fig. 3.3 illustrates this decomposition.

From Fig. 3.3, it can be observed that for a relatively complex natural language query,

name	book	price
John	book_1	19
John	book_2	30
Lisa	book_3	12
John	book_4	15
Mike	book_5	50
John	book_6	6

Natural Language Question:

How many books with price between 10 and 20 does John has?

SQL Query:

```
SELECT COUNT(book)
WHERE price < 20  AND  price > 10  AND  name == ”John”
```

RESULT:

2

Figure 3.3: A simple database and its query example

the corresponding SQL statement is often equally complex, which poses a significant challenge for deep learning models. However, if a template-based generation approach is adopted that fully considers and leverages the highly structured characteristics inherent in SQL statements, complex structures can be decomposed into relatively simple components, thereby enabling effective handling of complex statements through the sequential generation of these simpler structures.

3.3.2 Framework Description

Based on the overall approach proposed in the above problem analysis, the model will primarily be divided into two components: SELECT statement generation and WHERE statement generation. For WHERE statement generation, this model will adopt the SQL-Net model to generate WHERE statements. The primary contribution of this model lies in improving the generation of SELECT statements. The overall architecture of the model is illustrated in Fig. 3.4.

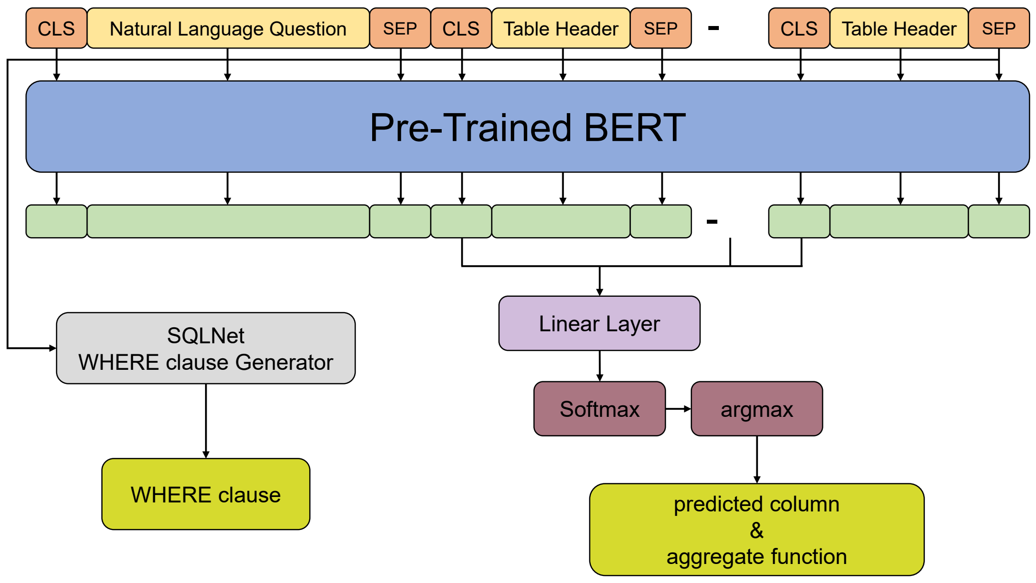

Figure 3.4: Main Structure of Our Proposed Method

Based on the task requirements, namely the design for generating SELECT statements, it is evident that the model's primary task can be divided into the following three stages:

1. Transform the input sequence by encoding it to obtain the encoded results.
2. Process the encoded results and extract information from the processed outcomes, where the information should be capable of characterizing the two essential elements of SELECT statements: first, the selected columns, and second, the aggregate functions employed.
3. Compare the predicted results with the ground truth, calculate the loss function, and compute the accuracy rate.

It can be observed that one of the most critical steps is how to encode the input sequence. Given that the input sequence must necessarily include natural language question sequences, and considering that natural language sequences contain substantial information related to column selection and aggregation functions, as demonstrated in the example shown in Fig. 3.3, it is therefore necessary, based on the core principles of Transformer models [107], to have a model that can comprehensively extract global attention information from the input sequence.

Therefore, the BERT model will be employed to process the input sequences. Since the BERT model [30] is constructed based on the Transformer architecture, it can build self-attention information for the entire input sequence, thereby fully capturing the contextual relationships within the input sequence. This capability is particularly crucial for understanding natural language questions in the input sequences.

Another important reason is that BERT, as a pre-trained model, provides readily available models for use [105]. For development purposes, it is only necessary to fine-tune the BERT model, which can significantly reduce the time required for model training.

3.3.3 Procedure of Our Proposed Method

Input Sequences

Based on the example in Fig. 3.3, the model's input sequence should not consist solely of natural language queries, but should also incorporate the corresponding database table header information. This is because the association between natural language queries and header information is highly correlated, and this correlation can directly determine the predictive information of the output SELECT statements.

Therefore, the following formula is adopted to construct the input:

$$x_{in} = [\text{CLS}], x^Q, [\text{SEP}], [\text{CLS}], x_1^H, [\text{SEP}], ..., [\text{CLS}], x_{N^H}^H, [\text{SEP}]$$

In the above formula Section 3.3.3, x_{in} represents the input sequence to the model, which is formed by concatenating a sentence sequence with several header sequences. Each sequence is enclosed by BERT's special tokens [30] CLS and SEP.

Furthermore, where x^Q represents the natural language question sequence, expressed in the following form:

$$x^Q = x_1^Q, x_2^Q, \cdots, x_{N_q}^Q$$

where x_i^Q denotes the i-th token in the question after tokenization using a BERT-based tokenizer, and N_q represents the total number of tokens after tokenization.

For the sequence composed of table headers, x_i^H represents the i-th header sequence of the database table corresponding to the natural language question, while N^H denotes the total number of headers. For any given header sequence x^H:

$$x^H = x_1^H, x_2^H, ..., x_{N_h}^H$$

Each x_i^H represents the i-th token after tokenization of a header using the BERT-based tokenizer, and N_h represents the total number of tokens within a header.

Intermediate Sequences

Based on the above, the intermediate sequence refers to the BERT output sequence. After inputting the input sequence x into the BERT model, we obtain the intermediate sequence x^{enc} that has been encoded by the BERT model:

$$x^{enc} = [\text{CLS}]^{enc}, x^{Q^{enc}}, [\text{SEP}]^{enc}, [\text{CLS}]^{enc}, x_1^{H^{enc}}, [\text{SEP}]^{enc}, ..., [\text{CLS}]^{enc}, {x_{N^H}^{H}}^{enc}, [\text{SEP}]^{enc}$$

The structure of the intermediate sequence is essentially consistent with that of the input sequence, where for any token in the input sequence x, the corresponding token after BERT encoding is denoted as $token^{enc}$.

Output Sequences

After obtaining the aforementioned intermediate sequence, we aim to process the intermediate sequence into the following output sequence y:

$$y = y_1^H, ..., y_{N^H}^H$$

Here, y_i^H represents the token corresponding to the i-th table header, and N^H denotes the total number of table headers. However, it is important to note that for the purpose of facilitating the extraction of key information from the output sequence, y^H differs from the header sequence represented by x^H, and is merely a single $token^{enc}$. Nevertheless, y^H must be capable of representing the complete information contained within a table header sequence x^H.

For each y^H, we have:

$$y^H = [P(agg_1), P(agg_2), \cdots, P(agg_{N_{agg}})]$$

Where N_{agg} = total number of aggregation functions + 2. Generally, this value is 7, consisting of 5 aggregation functions (COUNT, SUM, MAX, MIN, AVG), 1 option for no aggregation function applied to the column, and 1 option indicating that the column is not selected.

Summary

In the above discussion, we mentioned three sequences in the model execution pipeline: the input sequence x, the intermediate sequence x^{enc}, and the output sequence y. Among these, the transformation from x to x^{enc} is accomplished by BERT, while our primary contribution lies in the conversion from x^{enc} to y. Here, we will provide a summary and further elaborate on the technical details.

First, it is necessary to ensure that each y^H serves as a $token^{enc}$ while simultaneously representing the complete information of a table header. Therefore, leveraging the significance of the [CLS] token as a special classification marker in BERT models, we select the [CLS] token that precedes each encoded header sequence $x^{H^{enc}}$. Specifically, for a sequence $[\text{CLS}]^{enc}, x^{H^{enc}}, [\text{SEP}]^{enc}$, we have $y^H = [\text{CLS}]^{enc}$.

Subsequently, it is necessary to extract the following two pieces of information from y:

1. The selected column *col*
2. The aggregation function applied to column *col*

Therefore, according to the form of y^H in equation Section 3.3.3, the argmax function is employed to obtain the index of the maximum aggregation function probability $P(\text{agg})$ within each y^H. There are seven indices in total, representing five distinct aggregation function options, one option for column selection without aggregation function usage, and one option indicating that the column is not selected. Through this approach, the extraction of both types of information mentioned above can be accomplished simultaneously.

However, considering that the $[\text{CLS}]^{enc}$ tokens corresponding to each header sequence in BERT's output sequence x^{enc} are not directly equivalent to y^H, since for a given token, BERT's encoded output is a word vector with dimensions equal to BERT's last hidden layer size (typically 768), while y^H is a 7-dimensional vector where each value represents the probability of the corresponding indexed aggregation function. Therefore, an additional fully connected layer is required to map the vector from last_hidden_layer dimensions to 7 dimensions, which can be formulated as follows:

$$\begin{gathered} y^H = \text{FullConnection}([\text{CLS}]^{enc}) \\ \text{FullConnection}_{in} = \text{last\_hidden\_layer}, \quad \text{FullConnection}_{out} = 7 \end{gathered} \tag{3.1}$$

After performing the fully connected layer, in order to ensure the range of output values, a softmax function is appended following the fully connected layer; finally, an argmax function is employed to extract the predicted aggregation function index.

3.3.4 Loss Function

For the model's loss function, considering that the model's prediction results are generated using a sequence labeling approach—specifically, generating probability distributions over 7 aggregation function labels for each table header, while in reality each header has a unique and deterministic aggregation function label—the loss function will be computed using Sparse Categorical Cross Entropy. The formula for the loss function calculation is as follows:

$$loss = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$
$$loss_{total} = loss_{select} + loss_{where}$$

Where C denotes the number of classes, y represents the ground truth class label vector, and $\hat{y}$ denotes the predicted class probability vector output by the model. In sparse categorical cross-entropy, y is a vector containing only one non-zero element, indicating the true class of the sample, while $\hat{y}$ is a vector of length C, where each element represents the probability that the sample belongs to the corresponding class. Finally, $loss_{total}$

represents the total loss, $loss_{select}$ denotes the prediction loss for SELECT statements calculated according to the aforementioned formula, and $loss_{where}$ represents the prediction loss for WHERE statements. During model training, by minimizing the sparse categorical cross-entropy, the model learns to correctly predict the class of samples.

3.3.5 Summary

In summary, the operational workflow of the model is summarized as follows:

1. Natural language queries and table headers are tokenized to obtain their respective sequences. Each sequence is then enclosed with BERT's special tokens [CLS] and [SEP], and subsequently concatenated to form the input sequence.
2. From the BERT-encoded intermediate sequence, the encoded [CLS] tokens corresponding to each table header are extracted.
3. Each table header's [CLS] token is fed into a fully connected layer (linear layer) to obtain the prediction values for aggregation functions corresponding to each header.
4. The output from the linear layer is processed through a Softmax function for normalization.
5. The normalized values are input into an argmax function to obtain the index of the aggregation function with the highest prediction probability.
6. Through the aforementioned steps, the selected columns and their corresponding aggregation functions are extracted and identified.

3.4 Experiments

3.4.1 Introduction of Datasets

Considering the characteristics of our proposed method and the respective features of different datasets, three datasets are employed in this study: WiKiSQL Dataset [128], Academic Dataset [64], Spider Dataset [119].

Among these datasets, WikiSQL is a dataset with relatively simple structure that spans multiple domains and contains multiple database tables. The SELECT statements only select a single column, and the conditional statements contain only one or more simple WHERE clauses. This dataset will serve as the primary dataset for model training and fundamental testing.

The Academic dataset will serve as the dataset for extended testing of the model. Although the Academic dataset has a simpler structure and utilizes only a single database table, it provides query statements that are relatively favorable for template-based SQL generation approaches. Given the limited availability of cross-domain datasets [55], it can serve as a relatively straightforward examination of the model's predictive capabilities across different datasets. The preprocessing approach for the Academic dataset is relatively straightforward. Considering that the dataset employs JSON data format, we utilize Python's json module [38] for importing and regular expressions for processing, decomposing SQL query statements into the format described above for the WikiSQL dataset, consisting of SELECT column indices, WHERE condition statement lists, and aggregation function indices.

Regarding the Spider dataset, the examples in the Spider dataset exhibit high complexity, with natural language questions and SQL query statements being more complex and varied in form, spanning multiple databases and domains. However, considering the complexity of the Spider dataset and the fact that SQL queries contain structures not supported by the model (such as ORDER BY, JOIN, etc.), it is necessary to remove unsupported structures during preprocessing. Furthermore, when evaluating model performance, we will exclude the prediction of WHERE clauses from consideration.

In summary, the WikiSQL dataset will be utilized for model training and testing, while both Academic and Spider will serve as test sets, providing evaluations of varying difficulty levels.

3.4.2 Experiment Settings

In this experiment, we choose Seq2SQL and SQLNet as the baseline model. We partition the WikiSQL dataset into training, validation, and test sets following a 7:1:2 ratio. For the Academic and Spider datasets, we preprocess them to conform to the format of the

WikiSQL dataset.

For the evaluation metrics of experimental results, this experiment will adopt two evaluation indicators: query-match accuracy and execution accuracy. This is because the model employs a generation method based on fixed templates, and therefore, if an evaluation metric based on exact string matching between generated results and original SQL query statements were adopted, it would inevitably result in a loss of accuracy. For example, in the example presented in Fig. 3.3, the WHERE clauses WHERE price < 20 *AND* price > 10 AND name == ”John” and WHERE price > 10 *AND* price < 20 AND name == ”John” are both correct.

For query matching accuracy (hereinafter referred to as qm_acc), it is further categorized into two types: SELECT statement query matching accuracy (hereinafter referred to as qm_acc_{sel}) and WHERE statement query matching accuracy (hereinafter referred to as qm_acc_{where}). Let correct_x denote the number of correctly matched predicted x clauses, and num_{sql} represent the total number of SQL statements in the dataset. The term ”correct match” is defined as the model's predicted column corresponding to the table header index, aggregation function index, operators in WHERE statements, logical connectors, and values that are consistent with those in the ground truth query statements (where table header indices, aggregation function indices, etc., are generated during dataset preprocessing). Therefore, we have:

$$\text{qm\_acc}_{sel} = \frac{\text{correct}_{select}}{\text{num}_{sql}}$$
$$\text{qm\_acc}_{where} = \frac{\text{correct}_{where}}{\text{num}_{sql}}$$

For execution matching accuracy (hereinafter referred to as exec_acc), let correct_{exec} represent the total number of statements generated by the model that yield correct results when executed as SQL queries, then we have:

$$\text{correct}_{exec} = \frac{\text{correct}_{exec}}{\text{num}_{sql}}$$

Specifically, in the Spider test set, we no longer employ exec_acc as an evaluation metric, retaining only qm_acc as the evaluation criterion. Furthermore, in the second part of the Spider test set, which is exclusively designed for testing the prediction accuracy of

SELECT statements, we have qm_acc = qm_acc$_{sel}$.

Regarding the hyperparameters, we set the number of epochs to 16, batch size to 64, and learning rate to 10^{-4}. For model implementation, we primarily utilized PyTorch 0.4.1 under Python 3.6, with CUDA version 9.0 and cuDNN version 7.6.4. Additionally, our model incorporated the BertModel and BertTokenizer modules from the transformers library (version 4.6.0) developed by Hugging Face, which served as the BERT model and BERT-based tokenizer, respectively [109]. For the hardware infrastructure used in model training, we employed an NVIDIA GeForce RTX 4090 GPU with 24 GB of video memory (VRAM).

3.4.3 Experiment for WikiSQL Dataset

In the experiments, we compare our proposed method with Seq2SQL and SQLNet, where Seq2SQL is configured without reinforcement learning (RL).

The accuracy of SELECT clause prediction for the proposed model during training, showing the variation across training epochs on both the training set and validation set, is illustrated in Fig. 3.5.

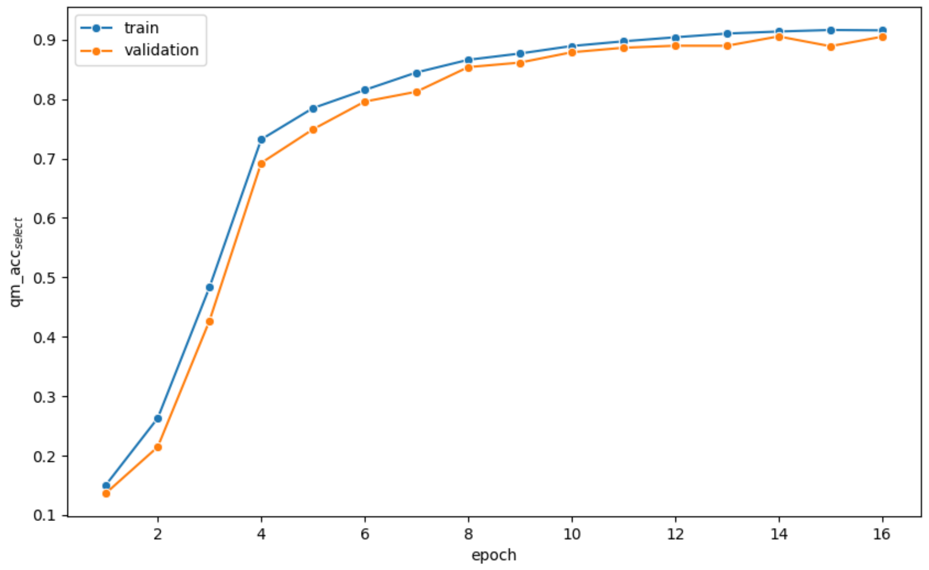

Figure 3.5: Accuracy of our proposed method on train and validation dataset of WiKiSQL

As illustrated in Fig. 3.5, it can be observed that the model's performance achieves convergence within 16 epochs. The model demonstrates exceptional performance, attaining approximately 85% accuracy on SELECT statement generation. Moreover, the model exhibits strong generalization capabilities, as evidenced in the figure, where the accuracy on the training set and validation set shows virtually no discrepancy, indicating that the model exhibits minimal overfitting phenomena.

Since our proposed method uses the SQLNet model for WHERE clause prediction, we aim to examine the impact of other parts in our method (such as the SELECT clause prediction part) on WHERE clause prediction. Therefore, we compared our results with the SQLNet model and generated Figure Fig. 3.6.

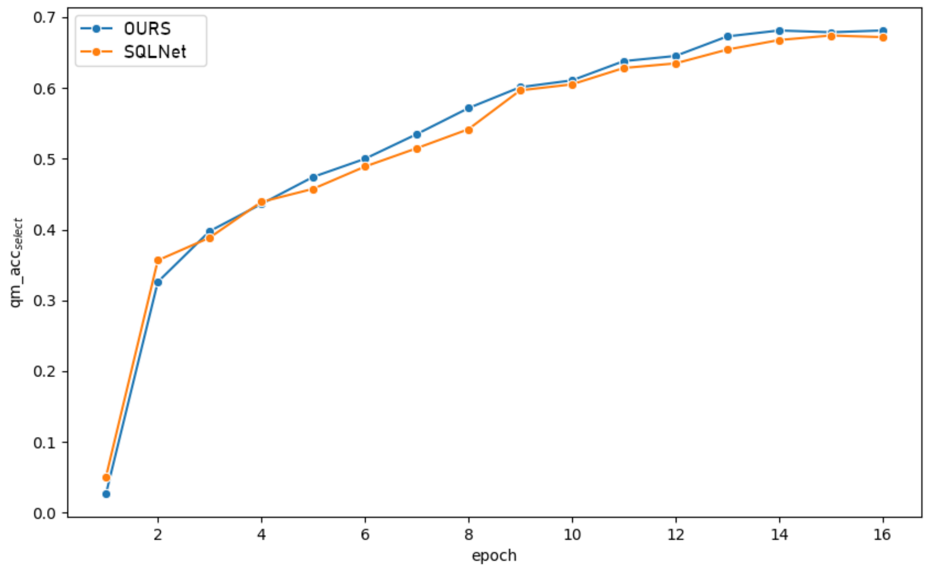

Figure 3.6: Accuracy of our proposed method and SQLNet on WHERE clause prediction

As shown in Fig. 3.6, it can be observed that the performance curves for WHERE clause prediction between the proposed model and the SQLNet model exhibit minimal differences. This indicates that the modifications made to the SELECT clause prediction module in the paper do not affect the performance of the WHERE clause prediction module. The underlying reason for this phenomenon is likely attributed to the fact that both the proposed model and the SQLNet model employ a loss computation method that linearly combines $loss_{select}$ and $loss_{where}$. Consequently, changes in the SELECT clause prediction module have minimal impact on the gradient descent and parameter optimization processes of the WHERE clause prediction module.

The comparsion of the result of our proposed method with the baseline models is shown in Table 3.1.

	qm_acc$_{select}$	qm_acc$_{where}$	qm_acc$_{exec}$
Seq2SQL (no RL)	88.93%	60.36%	57.12%
SQLNet	90.24%	67.18%	68.03%
OURS	**91.57%**	**68.12%**	**68.24%**

Table 3.1: Result compare on WiKiSQL dataset

According to Table 3.1, our method demonstrates improvements over SQLNet, achieving a 1.33% enhancement in SELECT clause prediction, a 0.94% improvement in WHERE clause prediction, and a 0.21% increase in execution accuracy. Compared to the Seq2SQL model, our method exhibits substantially greater improvements, with a 2.64% enhancement in SELECT clause prediction, a 7.76% improvement in WHERE clause prediction, and a 11.12% increase in execution accuracy.

3.4.4 Experiment on Academic & Spider Dataset

We conducted experiment on the Academic dataset, and the result is shown in Table 3.2.

	qm_acc$_{select}$	qm_acc$_{where}$	qm_acc$_{exec}$
Seq2SQL (no RL)	76.25%	45.76%	43.68%
SQLNet	84.31%	59.36%	57.46%
OURS	**89.47%**	57.86%	56.68%

Table 3.2: Result compare on Academic dataset

The performance of our method and the baseline models decreased on the Academic. But our method still made 5.16% improvement compared to SQLNet and 13.22% improvement compared to Seq2SQL on the prediction of SELECT clause on the Academic dataset.

Since the Spider dataset is complex and highly challenging, we conducted experiments only on SELECT clause prediction, with results shown in Table 3.3.

The performace of our method and the baseline models declines a lot on the Spider dataset. But our method made 7.37% improvement compared to SQLNet and 20.7% improvement compared to Seq2SQL on the prediction of SELECT clause on the Spider dataset.

	qm_acc$_{select}$
Seq2SQL (no RL)	33.86%
SQLNet	47.19%
OURS	**54.56%**

Table 3.3: Result compare on Spider dataset

3.5 Discussion

The method proposed in this chapter demonstrates good generalization capabilities and performs well in SELECT clause generation, indicating that fine-tuning approaches based on pre-trained models are effective. However, several areas for improvement remain: WHERE clause generation still relies on SQLNet; template-based generation methods have inherent limitations; our approach has not yet addressed one of the most challenging problems in NL2SQL—the generation of conditional statements. Additionally, due to the large size of the BERT model, the costs associated with training and deployment remain relatively high.

3.6 Conclusion

The experimental results in this chapter demonstrate that the model exhibits excellent generalization capabilities, achieving superior performance in SELECT statement prediction. When confronted with more complex datasets, the model shows relatively smaller performance degradation compared to baseline models. Therefore, this paper provides a novel approach for SQL statement prediction: leveraging pre-training and fine-tuning techniques, such as BERT models, can enhance the model's generalization ability, enabling it to better adapt to the intricate and complex natural language queries encountered in real-world applications.

Chapter 4

Conclusion & Future Work

Contents

4.1 Overview

In Chapter 2 and Chapter 3, we presented our proposed approach. We performed comprehensive experiments to validate our model. Our research may also provide insights for future investigations of related tasks. Additionally, there remains scope for further refinement of our methodology.

4.2 Main Contributions

We validated the importance of feature extraction for improving model performance from two perspectives. In Chapter 2, we extracted frequency, spatial, temporal, and global features from EEG signals; in Chapter 3, we leveraged pre-trained models to understand textual content. Both approaches achieved favorable results.

There are several key contributions of this thesis:

1. In the EEG-based emotion recognition task, unlike many studies that focus solely on extracting spatial and temporal features, we propose a multi-feature extraction approach called MENNA (Multi-Feature Neural Network Architecture), which extracts frequency, spatial, temporal, and global features. Our method achieves 95.87% accuracy on the SEED dataset, 94.42% accuracy on the valence dimension of the DEAP dataset, and 93.67% accuracy on the arousal dimension.

2. We conducted extensive experiments with our proposed MENNA on the SEED and DEAP datasets. The experimental results demonstrate that different EEG features contribute varying degrees of information, with spatial and temporal features being the most significant contributors. We generated confusion matrices, t-SNE feature distributions, and Grad-CAM brain region activation heatmaps for comprehensive analysis. The results indicate that while the model exhibits stable performance across different emotion categories overall, there remain notable variations in accuracy among different emotional states. Furthermore, distinct patterns of brain region activation were observed across different emotional conditions.

3. Our proposed MENNA employs an adaptive GCN approach. In contrast to traditional GCNs where the adjacency matrix requires manual pre-configuration, our

proposed adaptive GCN can autonomously learn graph-related parameters, thereby eliminating manual intervention and enhancing the model's generalizability.

4. We propose a method that combines pre-trained BERT with fixed template-based generation, termed BERT with Template, to address the NL2SQL task. We achieve results of 85.82%, 84.34%, and 54.48% on SELECT statement generation across the WikiSQL, Academic, and Spider datasets, respectively. Furthermore, the model demonstrates considerable potential in terms of generalizability.

4.3 Future Work

In the future, we still have a lot of work to do to improve our methods.

1. We can explore whether EEG signals contain additional extractable features through more extensive literature review. Furthermore, we can apply our methodology to multi-modal data to further validate the effectiveness of our approach.
2. Consider applying distillation or other model optimization techniques to pre-trained models to reduce training costs during model fine-tuning, such as GPU memory consumption and other related issues.
3. Consider more advanced SQL generation methodologies, such as syntax tree-based generation approaches and other sophisticated techniques.
4. Consider the generation of more complex SQL statements, such as the generation of conditional IF clauses.

4.4 Conclusion

In this thesis, our research can be summarized as follows:

1. For multi-channel EEG signals, beyond spatial and temporal features, frequency and global features can also be extracted. Through multi-feature extraction, both the

accuracy and robustness of the model are enhanced. This approach can be applied to deep learning tasks involving other types of signal data.

2. When designing models, the advantages of deep learning can be leveraged to develop adaptive models that enable autonomous learning of key parameters, thereby reducing manual intervention and enhancing model usability.

3. Pre-trained models can be integrated with traditional methods, thereby enhancing the model's capability to extract deep-level features and improving the model's accuracy and generalization performance.

Appendix A

List of Abbreviations

AI	Artificial Intelligence
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
LLM	Large Language Model
GPT	Generative Pretrained Transformer
SQL	Structured Query Language
NL	Natural Language
NL2SQL	Natural Language to Structured Query Language
NLIDB	Natural Language Interface to DataBases
MLP	Multi-Layer Perceptron
GPU	Graphics Processing Unit
DBN	Deep Belief Network
ResNet	Residual Network
NLP	Natural Language Processing
CV	Computer Vision
ECG	Electrocardiography
HR	Heart Rate
EMG	Electromyography
EEG	Electroencephalogram
PSD	Power Spectral Density

GNN	Graph Neural Network
GCN	Graph Convolutional Network
BERT	Bidirectional Encoder Representations from Transformers
XML	Cross-Lingual Language Model
MLM	Masked Language Model
FDFs	Frequency Domain Features
TFDFs	Time Frequency Domain Features
FFT	Fast Fourier Transformation
DE	Differential Entropy
WPD	Wavelet Packet Decomposition
GBDT	Gradient Boosting Decision Tree
t-SNE	t-distributed Stochastic Neighbor Embedding
Grad-CAM	Gradient-weighted Class Activation Mapping
SE	Squeeze-and-Excitation
CA	Channel Attention
FC	Fully-Connected
MA	Multi-head Attention
SVM	Support Vector Machine
ELM	Extreme Learning Machine
LR	Logistic Regression
ASFM	Adaptive Subspace Feature Matching
DGCNN	Dynamical Convolutional Neural Networks
BDAE	Bimodal Deep AutoEncoder
GELM	Graph Regularized Extreme Learning Machine
HNSN	Hierarchical Network with Subnetwork Nodes
GSCCA	Group Sparse Canonical Correlation Analysis
STRNN	Spatial–Temporal Recurrent Neural Network
BiDANN	Bi-hemispheres Domain Adversarial Neural Network
Bi-HDM	Bi-Hemispheric Discrepancy Model
RGNN	Regularized Graph Neural Networks
AFO	Aggregate Function Operators
RL	Reinforcement Learning

Appendix B

Software Used

The followings are the python modules we mainly used.

numpy
A famous and basic python library for scientific computing, which supports large-scale array and matrix computing.

pytorch
A famous open-source machine learning library, we use it to build MENNA.

scikit-learn
This is a toolbox for traditional machine learning and statistical learning, we use it to process results.

matplotlib
This is a plotting library in Python, we use it to draw figures.

keras
A high level deep learning model builder, we use it to build BERT with Template.

transformers
A package offered by huggingface to help users quickly train and infer Transformer models. We use it to fine-tune BERT.

The followings are the software or tools we mainly used.

VSCode
An IDE for coding, with various plugins to improve user expeience. We use it to code.

Windows Remote Destop
A Windows built-in software to help users to connect and manipulate another computer. We use it to connect to server.

hugginggface
A large AI community aiming to provide service of computing, model and dataset storage, etc. We use it to store some datasets.

github
A famous code hosting platform. We search open-source models on it.

matlab
A famous software environment for proprietary multi-paradigm programming language and numeric computing, we use it to preprocess some data.

Appendix C

Values of Properties

System Properties	**Value**
System	Windows 11 Pro
Processor	Intel i9-14900K
Random Access Memory (RAM)	64 GB
System Type	64-bit OS, x64-based processor
GPU	NVIDIA GeForce RTX 4090
Graphic Memory	24 GB

Table C.1: The System Properties for all Experiments

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819, 2019.

[3] 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.

[4] Karam Ahkouk, Machkour Mustapha, Majhadi Khadija, and Mama Rachid. A review of the text to sql frameworks. In *Proceedings of the 4th International Conference on Networking, Information Systems & Security*, pages 1–6, 2021.

[5] Amjed S Al-Fahoum and Ausilah A Al-Fraihat. Methods of eeg signal features extraction using linear analysis in frequency and time-frequency domains. *International Scholarly Research Notices*, 2014(1):730218, 2014.

[6] Soraia M Alarcao and Manuel J Fonseca. Emotions recognition using eeg signals: A survey. *IEEE transactions on affective computing*, 10(3):374–393, 2017.

[7] Hezam Albaqami, Ghulam Mubashar Hassan, Abdulhamit Subasi, and Amitava Datta. Automatic detection of abnormal eeg signals using wavelet feature extraction and gradient boosting decision tree. *Biomedical Signal Processing and Control*, 70:102957, 2021.

[8] Mouhannad Ali, Ahmad Haj Mosa, Fadi Al Machot, and Kyandoghere Kyamakya. Eeg-based emotion recognition approach for e-healthcare applications. In *2016 eighth international conference on ubiquitous and future networks (ICUFN)*, pages 946–950. IEEE, 2016.

[9] Anis Ameera, A Saidatul, and Zunaidi Ibrahim. Analysis of eeg spectrum bands using power spectral density for pleasure and displeasure state. In *IOP conference series: Materials science and engineering*, volume 557, page 012030. IOP Publishing, 2019.

[10] Michael L Anderson, Josh Kinnison, and Luiz Pessoa. Describing functional diversity of brain regions and brain networks. *Neuroimage*, 73:50–58, 2013.

[11] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81, 1995.

[12] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644, 2011.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[14] Erol Başar, Canan Başar-Eroglu, Sirel Karakaş, and Martin Schürmann. Gamma, alpha, delta, and theta oscillations govern cognitive processes. *International journal of psychophysiology*, 39(2-3):241–248, 2001.

[15] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

[16] Kirsten Boehner, Rogério DePaula, Paul Dourish, and Phoebe Sengers. How emotion is made and measured. *International Journal of Human-Computer Studies*, 65(4):275–291, 2007.

[17] Florin Brad, Radu Iacob, Ionel Hosu, and Traian Rebedea. Dataset for a neural natural language interface for databases (nnlidb). *arXiv preprint arXiv:1707.03172*, 2017.

[18] Gianni Brauwers and Flavius Frasincar. A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3279–3298, 2021.

[19] Scott Brave and Cliff Nass. Emotion in human-computer interaction. In *The human-computer interaction handbook*, pages 103–118. CRC Press, 2007.

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[21] 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):1014, 2017.

[22] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5659–5667, 2017.

[23] James A Coan and JJ Allen. State and trait of frontal eeg asymmetry in emotion. *The asymmetrical brain*, pages 566–615, 2003.

[24] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[25] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. Deep learning for electroencephalogram (eeg) classification tasks: a review. *Journal of neural engineering*, 16(3):031001, 2019.

[26] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[27] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[28] Haojin Deng, Shiqi Wang, Yimin Yang, WG Will Zhao, Hui Zhang, Ruizhong Wei, QM Jonathan Wu, and Bao-Liang Lu. Minimizing eeg human interference: A study of an adaptive eeg spatial feature extraction with deep convolutional neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 2024.

[29] Norman K Denzin. *On understanding emotion*. Transaction Publishers, 1984.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[31] Matthew L Dixon, Ravi Thiruchselvam, Rebecca Todd, and Kalina Christoff. Emotion and the prefrontal cortex: An integrative review. *Psychological bulletin*, 143(10):1033, 2017.

[32] Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

[33] Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*, 2018.

[34] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[35] Ruo-Nan Duan, Jia-Yi Zhu, and Bao-Liang Lu. Differential entropy feature for eeg-based emotion classification. In *2013 6th international IEEE/EMBS conference on neural engineering (NER)*, pages 81–84. IEEE, 2013.

[36] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[37] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[38] Python Software Foundation. JSON encoder and decoder.

[39] Yue Gao, Xiangling Fu, Tianxiong Ouyang, and Yi Wang. Eeg-gcn: spatio-temporal and self-adaptive graph convolutional networks for single and multi-view eeg-based emotion recognition. *IEEE Signal Processing Letters*, 29:1574–1578, 2022.

[40] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*, 2016.

[41] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[42] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[44] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[46] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[48] Marlies Houben, Wim Van Den Noortgate, and Peter Kuppens. The relation between short-term emotion dynamics and psychological well-being: A meta-analysis. *Psychological bulletin*, 141(4):901, 2015.

[49] Dichao Hu. An introductory survey on attention mechanisms in nlp problems. In *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2*, pages 432–448. Springer, 2020.

[50] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[51] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[52] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic markets*, 31(3):685–695, 2021.

[53] George Katsogiannis-Meimarakis and Georgia Koutrika. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851, 2021.

[54] George Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936, 2023.

[55] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. Natural language to sql: Where are we today? *Proceedings of the VLDB Endowment*, 13(10):1737–1750, 2020.

[56] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[57] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[58] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. Deap: A database for emotion analysis; using physiological signals. *IEEE transactions on affective computing*, 3(1):18–31, 2011.

[59] Z Koudelková, M Strmiska, and R Jašek. Analysis of brain waves according to their frequency. *Int. J. Biol. Biomed. Eng*, 12:202–207, 2018.

[60] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[61] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[62] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[63] Young-Eun Lee and Seo-Hyun Lee. Eeg-transformer: Self-attention from transformer architecture for decoding eeg of imagined speech. In *2022 10th International winter conference on brain-computer interface (BCI)*, pages 1–4. IEEE, 2022.

[64] Fei Li and Hosagrahar V Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.

[65] Fei Li and Hosagrahar V Jagadish. Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712, 2014.

[66] Gen Li, Chang Ha Lee, Jason J Jung, Young Chul Youn, and David Camacho. Deep learning for eeg data analytics: A survey. *Concurrency and Computation: Practice and Experience*, 32(18):e5199, 2020.

[67] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075, 2023.

[68] Mu Li and Bao-Liang Lu. Emotion classification based on gamma-band eeg. In *2009 Annual International Conference of the IEEE Engineering in medicine and biology society*, pages 1223–1226. ieee, 2009.

[69] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[70] Yang Li, Lei Wang, Wenming Zheng, Yuan Zong, Lei Qi, Zhen Cui, Tong Zhang, and Tengfei Song. A novel bi-hemispheric discrepancy model for eeg emotion recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 13(2):354–367, 2020.

[71] 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 *IJCAI*, pages 1561–1567, 2018.

[72] 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(2):568–578, 2019.

[73] Kristen A Lindquist and Lisa Feldman Barrett. A functional architecture of the human brain: emerging insights from the science of emotion. *Trends in cognitive sciences*, 16(11):533–540, 2012.

[74] Seppo Linnainmaa. Algoritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden. 2020.

[75] Wei Liu, Wei-Long Zheng, and Bao-Liang Lu. Emotion recognition using multimodal deep learning. In *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part II 23*, pages 521–529. Springer, 2016.

[76] Yu Liu, Yufeng Ding, Chang Li, Juan Cheng, Rencheng Song, Feng Wan, and Xun Chen. Multi-channel eeg-based emotion recognition via a multi-level features guided capsule network. *Computers in Biology and Medicine*, 123:103927, 2020.

[77] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[78] Yifei Lu, Wei-Long Zheng, Binbin Li, and Bao-Liang Lu. Combining eye movements and eeg to enhance emotion recognition. In *IJCAI*, volume 15, pages 1170–1176. Buenos Aires, 2015.

[79] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[80] Albert Mehrabian. Framework for a comprehensive description and measurement of emotional states. *Genetic, social, and general psychology monographs*, 121(3):339–361, 1995.

[81] Albert Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14:261–292, 1996.

[82] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[83] Paul L Nunez and Ramesh Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford university press, 2006.

[84] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[85] SK Pahuja, Karan Veer, et al. Recent approaches on classification and feature extraction of eeg signal: A review. *Robotica*, 40(1):77–101, 2022.

[86] Bo Pan and Wei Zheng. Emotion recognition based on eeg using generative adversarial nets and convolutional neural network. *computational and Mathematical Methods in Medicine*, 2021(1):2520394, 2021.

[87] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[88] Luiz Pessoa. Understanding emotion with brain networks. *Current opinion in behavioral sciences*, 19:19–25, 2018.

[89] Robert Plutchik. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist*, 89(4):344–350, 2001.

[90] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[91] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[92] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[93] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Tutorials*, pages 15–18, 2019.

[94] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[95] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural

networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017.

[96] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*, 2021.

[97] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3288–3291. IEEE, 2012.

[98] Koosha Sharifani and Mahyar Amini. Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal*, 10(07):3897–3904, 2023.

[99] Enze Shi, Sigang Yu, Yanqing Kang, Jinru Wu, Lin Zhao, Dajiang Zhu, Jinglei Lv, Tianming Liu, Xintao Hu, and Shu Zhang. Meet: A multi-band eeg transformer for brain states decoding. *IEEE Transactions on Biomedical Engineering*, 71(5):1442–1453, 2023.

[100] Afshin Shoeibi, Delaram Sadeghi, Parisa Moridian, Navid Ghassemi, Jonathan Heras, Roohallah Alizadehsani, Ali Khadem, Yinan Kong, Saeid Nahavandi, Yu-Dong Zhang, et al. Automatic diagnosis of schizophrenia in eeg signals using cnn-lstm models. *Frontiers in neuroinformatics*, 15:777977, 2021.

[101] 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):2074, 2018.

[102] Tengfei Song, Wenming Zheng, Peng Song, and Zhen Cui. Eeg emotion recognition using dynamical graph convolutional neural networks. *IEEE Transactions on Affective Computing*, 11(3):532–541, 2018.

[103] Wilson L Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[104] Stefano Tortora, Stefano Ghidoni, Carmelo Chisari, Silvestro Micera, and Fiorenzo Artoni. Deep learning-based bci for gait decoding from eeg with lstm recurrent neural network. *Journal of neural engineering*, 17(4):046011, 2020.

[105] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.

[106] M Uma, V Sneha, G Sneha, J Bhuvana, and B Bharathi. Formation of sql from natural language query using nlp. In *2019 international conference on computational intelligence in data science (ICCIDS)*, pages 1–5. IEEE, 2019.

[107] 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*, 30, 2017.

[108] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[109] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[110] Xiaofen Xing, Zhenqi Li, Tianyuan Xu, Lin Shu, Bin Hu, and Xiangmin Xu. Sae+ lstm: A new framework for emotion recognition from multi-channel eeg. *Frontiers in neurorobotics*, 13:37, 2019.

[111] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

[112] Heekyung Yang, Jongdae Han, and Kyungha Min. A multi-column cnn model for emotion recognition from eeg signals. *Sensors*, 19(21):4736, 2019.

[113] Yilong Yang, Qingfeng Wu, Yazhen Fu, and Xiaowei Chen. Continuous convolutional neural network with 3d input for eeg-based emotion recognition. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part VII 25*, pages 433–443. Springer, 2018.

[114] Yilong Yang, Qingfeng Wu, Ming Qiu, Yingdong Wang, and Xiaowei Chen. Emotion recognition from multi-channel eeg through parallel convolutional recurrent neural network. In *2018 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2018.

[115] Yimin Yang, QM Jonathan Wu, Wei-Long Zheng, and Bao-Liang Lu. Eeg-based emotion recognition using hierarchical network with subnetwork nodes. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):408–419, 2017.

[116] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[117] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology*, pages 364–375. Springer, 2014.

[118] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769*, 2018.

[119] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

[120] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

[121] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1253, 2018.

[122] Tong Zhang, Wenming Zheng, Zhen Cui, Yuan Zong, and Yang Li. Spatial–temporal recurrent neural network for emotion recognition. *IEEE transactions on cybernetics*, 49(3):839–847, 2018.

[123] Yuchan Zhang, Guanghui Yan, Wenwen Chang, Wenqie Huang, and Yueting Yuan. Eeg-based multi-frequency band functional connectivity analysis and the application of spatio-temporal features in emotion recognition. *Biomedical Signal Processing and Control*, 79:104157, 2023.

[124] 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.

[125] Wei-Long Zheng, Jia-Yi Zhu, and Bao-Liang Lu. Identifying stable patterns over time for emotion recognition from eeg. *IEEE transactions on affective computing*, 10(3):417–429, 2017.

[126] Wenming Zheng. Multichannel eeg-based emotion recognition via group sparse canonical correlation analysis. *IEEE Transactions on Cognitive and Developmental Systems*, 9(3):281–290, 2016.

[127] Peixiang Zhong, Di Wang, and Chunyan Miao. Eeg-based emotion recognition using regularized graph neural networks. *IEEE Transactions on Affective Computing*, 13(3):1290–1301, 2020.

[128] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[129] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 655–664, 2017.

[130] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.