applied sciences

MDPI

Article

A Multi-Layer Feature Fusion Model Based on Convolution and Attention Mechanisms for Text Classification

Hua Yang [1,*,†], Shuxiang Zhang [1,†], Hao Shen [2], Gexiang Zhang [3], Xingquan Deng [1], Jianglin Xiong [1], Li Feng [1], Junxiong Wang [1], Haifeng Zhang [1] and Shenyang Sheng [1]

[1] School of Mathematics and Computer Science, Wuhan Polytechnic University, Wuhan 430040, China; kdocked@163.com (S.Z.); xqcoder@163.com (X.D.); xiongjianglin@126.com (J.X.); goodboy20230715@163.com (L.F.); junxioang_wang@163.com (J.W.); 15507993082@163.com (H.Z.); 13781289840@163.com (S.S.)
[2] Baijuncheng Technology Co., Ltd., Wuhan 434000, China; shenhao589@billjc.com
[3] School of Automation, Chengdu University of Information Technology, Chengdu 610059, China; zhgxdylan@126.com
* Correspondence: huayang@whpu.edu.cn
† These authors contributed equally to this work.

Abstract: Text classification is one of the fundamental tasks in natural language processing and is widely applied in various domains. CNN effectively utilizes local features, while the Attention mechanism performs well in capturing content-based global interactions. In this paper, we propose a multi-layer feature fusion text classification model called CAC, based on the Combination of CNN and Attention. The model adopts the idea of first extracting local features and then calculating global attention, while drawing inspiration from the interaction process between membranes in membrane computing to improve the performance of text classification. Specifically, the CAC model utilizes the local feature extraction capability of CNN to transform the original semantics into a multi-dimensional feature space. Then, global attention is computed in each respective feature space to capture global contextual information within the text. Finally, the locally extracted features and globally extracted features are fused for classification. Experimental results on various public datasets demonstrate that the CAC model, which combines CNN and Attention, outperforms models that solely rely on the Attention mechanism. In terms of accuracy and performance, the CAC model also exhibits significant improvements over other models based on CNN, RNN, and Attention.

Keywords: text classification; convolutional neural networks (CNN); attention mechanism; multi-layer feature fusion

Citation: Yang, H.; Zhang, S.; Shen, H.; Zhang, G.; Deng, X.; Xiong, J.; Feng, L.; Wang, J.; Zhang, H.; Sheng, S. A Multi-Layer Feature Fusion Model Based on Convolution and Attention Mechanisms for Text Classification. *Appl. Sci.* **2023**, *13*, 8550. https://doi.org/10.3390/app13148550

Academic Editor: Ugo Vaccaro

Received: 19 June 2023
Revised: 16 July 2023
Accepted: 19 July 2023
Published: 24 July 2023

1. Introduction

With the rapid development of internet technology, an increasing amount of textual information is flowing into the internet. These massive amounts of text data contain rich semantic information, which plays a significant guiding role in the activities and development of society. Therefore, it has become particularly important to classify and extract information from these textual data. Text classification technology, as one of the most fundamental and essential tasks in the field of natural language processing, enables people to process text data more conveniently and extract the most critical information from it.

In recent years, with the continuous maturity of deep learning technology, significant progress has been made in text classification. A series of deep learning models based on convolutional neural networks (CNN), recurrent neural networks (RNN), and attention mechanisms have been widely applied to text classification tasks. The core problem of text classification lies in the extraction and representation of text features, in which deep learning technology [1–3] has more advantages over traditional machine learning [4–7]

methods. Its multi-layered and complex neural network structure can automatically learn the hidden features of text from data, and therefore, it can achieve better classification results than shallow learning models.

Convolutional neural networks (CNN) have always been the core and mainstream algorithm in the field of image processing, which extracts different feature maps by applying convolution kernels of different sizes and strides on the input image and performs certain calculations. In 2014, Kim first introduced CNN into the field of natural language processing and proposed the TextCNN model [8] for text classification. Subsequently, a series of CNN-based text processing models [9–12] were proposed and applied to text classification tasks, proving the powerful semantic feature extraction ability of CNN. However, due to the limitations of the convolution kernel size, CNN has strong advantages in local feature extraction, but is difficult to extract long-distance dependency relationships within text. In order to tackle this problem, researchers have explored various approaches, such as increasing the depth of the network [13,14] or incorporating multi-scale feature connections [15] to indirectly extract long-range features. This has partially alleviated this limitation of CNN and achieved some promising results. However, some researchers believe that increasing the depth is not a good method [16]. Although increasing the depth can improve accuracy, it also makes the model more complex, increases the number of parameters, produces larger performance overhead, and is more prone to overfitting during training. This has also spawned the emergence of models such as SVDCNN [17] and GCN [18].

Recurrent Neural Networks (RNNs), especially LSTM [19–21] and GRU [22–24], have been the primary techniques in the field of text processing. RNNs use sequential time steps to compute and propagate hidden features to represent and classify text [25–28], which endows RNNs with the capability of memorization, making it better at modeling sentence sequences. However, it is precisely this recurrent nature of RNNs that makes it prone to gradient vanishing and exploding problems when processing long sequences, and its sequential nature hinders parallel computing during training. To address these issues, some researchers [29] have improved RNNs' performance by incorporating the Attention mechanism. Based on this, Google introduced the Transformer [30] and Bert [31] models in 2017 and 2018, respectively, which entirely use the Attention mechanism to calculate global dependency relationships between inputs and outputs and also support more parallelization. In particular, BERT, a pre-trained model based on the Transformer, has brought new breakthroughs to various natural language processing tasks, including text classification, fully demonstrating the value of the Attention mechanism [32–36] in global feature processing. However, Transformer-based models tend to be more complex and require more data and higher computing resources.

This paper proposes a CAC classification model based on CNN and the Attention mechanism, which fully leverages the local feature extraction capabilities of CNNs. By introducing the Attention mechanism, the model avoids the limitations of CNN in global dependency modeling and achieves a perfect combination of the advantages of both approaches. While RNN networks can also be used for global feature extraction, there have been many works that combine RNN with CNN [37–39] or RNN with Attention mechanisms [40–42] to model and classify textual data. However, considering the issue of feature forgetting and poor parallelization of RNN and the superior feature extraction capabilities of attention compared to RNN, we have chosen to build the CAC model by combining CNN with Attention. Unlike other CNN and Attention-based models [43–46], our CAC model has a simpler structure and lower complexity.

In the design of the model's structure, we draw inspiration from the computational principles of membrane computing [47,48]. Membrane computing is a hierarchical and parallel computational model where individual computing units are encapsulated within different membranes and communicate and interact through the membranes to exchange information and perform computations, thus achieving highly parallel computing processes. Building upon this, we incorporate the hierarchical structure, isolated computation, and parallel computation features of membrane computing into the design of our model,

combining the advantages of CNN and Attention. This enables the model to handle both local features and extract global relationships. Furthermore, we simplify the stacked computational Attention structure to support multi-layer computations, significantly reducing the computational complexity of the model. Experimental results on multiple text classification datasets demonstrate that compared to CNN and Transformer, our novel model not only converges faster but also achieves significantly improved accuracy.

The remainder of this paper is organized as follows. Section 2 focuses on the implementation process of our model, including its structure, parameter settings, and other details. Section 3 presents the performance of our model on different datasets. In Section 4, we performed ablation experiments to investigate the role of each component of our model. Finally, we summarize the main contributions and limitations of this paper and suggest future research directions.

2. Methodologies

2.1. Overview

Our model consists of five main modules in terms of structure, as shown in Figure 1, from bottom to top: the input embedding layer, the feature extraction layer, the feature fusion layer, the text representation layer, and the classification prediction layer.

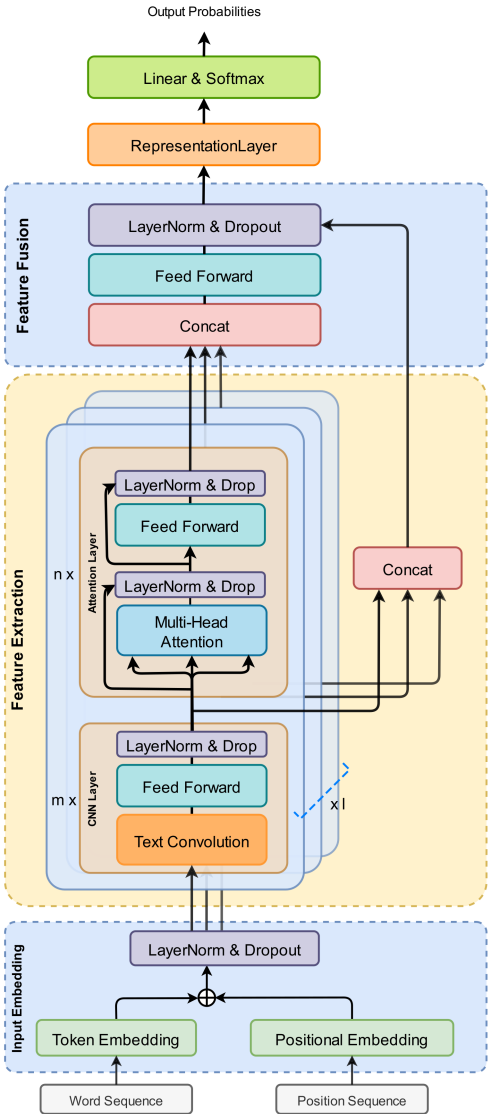

Figure 1. The overall structure of the proposed CAC model.

The input embedding layer is responsible for converting the text into vector representations, capturing the semantic information of words. By mapping words to high-dimensional vector space, the word embedding layer represents words in the text as continuous and semantically similar vectors, helping the model understand the relationships and meanings between words. The feature extraction layer is the core part of the model, utilizing methods such as convolutional neural networks and attention mechanisms to extract local and global features from the text. The convolutional neural network extracts local features by sliding windows over the text, capturing the local relationships between words. The attention mechanism models the global interactions within the entire text sequence by learning the importance weights of different positions, enabling the model to focus on key words or phrases and capture the global semantic information. The feature fusion layer combines the local and global features obtained from the feature extraction layer to comprehensively consider information from different levels. By merging representations of different features, the model can better capture multi-level semantic information in the text and improve the accuracy and robustness of the classification task. The text representation layer encodes and integrates the fused features to generate a vector representing the entire text. By representing the text as a fixed-length vector, the model can transform texts of different lengths into a unified representation, facilitating subsequent classification prediction tasks. Finally, the classification prediction layer uses the SoftMax classifier to map the text representation to predefined categories, completing the text classification task.

In addition, in the structural design of the model, we incorporate computational principles such as hierarchical structure, isolated computation, and parallel computation inspired by membrane computing. In terms of hierarchical structure design, we treat the input embedding layer, feature extraction layer, feature fusion layer, text representation layer, and classification prediction layer as independent hierarchical modules, each with its own computational units and structure. Each layer performs feature extraction in different feature spaces. In isolated computation, similar to the membrane-isolated nature of computational units in membrane computing, the local and global features of the text are computed separately. Different computational rules are applied in each feature extraction unit, and information is propagated, and feature fusion is achieved through interactions between membranes. In parallel computation, multiple parallel feature extraction modules are designed, with each module responsible for processing text information at different scales, thereby improving computational efficiency and model performance.

To achieve richer feature representation, we employ different optimization strategies and transformation rules, enabling each layer to perform feature propagation in different feature spaces. Specifically, at the input layer, we enhance the sequential information of the input sequence using learnable positional vectors. At the feature extraction layer, we employ convolution and attention computations to capture both local and global relational features within the sequence. To reduce computational complexity, we simplify the stacked attention structure to support multi-layer computations. In the feature fusion layer, we merge the features obtained from both global and local outputs. Finally, at the linear classification layer, we utilize a standard fully connected neural network to map the feature vectors to the predicted results.

In the following sections, we will provide a detailed explanation of the implementation details and main functions of each module in the model.

2.2. Input Embedding Module

When modeling text-related data, the first step is to vectorize it. In machine learning, common methods for text representation include one-hot encoding, bag-of-words model, and TF-IDF. However, in deep learning, the most common approach is to map individual words (or characters) to a low-dimensional dense vector space using an embedding layer known as Token Embedding.

Unlike CNN models with n-gram or RNN models with temporal structures, these models' network structures already possess the capability to capture sequential features.

However, this is not the case for models based on attention mechanisms. This is because attention mechanisms, in their computational process, involve linear transformations of a few matrices without considering the positional information of words in the text. In self-attention mechanisms, the representation of each word is computed based on the representations of all other words in the text, without explicit indication of which words are closer or farther from the current word. Therefore, to enable attention-based models to utilize the sequential information in the text, the addition of positional information becomes crucial. The introduction of positional vectors addresses this issue. By adding a positional vector to each word, the model can better differentiate words at different positions in the text and capture the relationships and dependencies between words more sensitively during the attention computation process.

As shown in Figure 1, the CAC model consists of two key components in the Input Embedding module: Token Embedding and Positional Embedding. Token Embedding assigns an index to each word based on a constructed vocabulary (Word Sequence), and then uses an Embedding layer to represent each word as a word vector. On the other hand, Positional Embedding encodes the positional information of each word (Position sequence), and also utilizes an Embedding layer to convert the positional encoding into positional vectors.

The presence of positional vectors enables the model to better understand the temporal information within the text and accurately capture important contextual relationships. In our model, we employ learnable embeddings to represent the positional encoding. As shown in Figure 2, for a given word, its input representation $E = [e_1, e_2, \ldots, e_n]$ is obtained by element-wise summation of the word embeddings $T = [t_1, t_2, \ldots, t_n]$ and position embeddings $P = [p_1, p_2, \ldots, p_n]$ transformed by the Embedding layer. Each element of the positional embeddings P is learnable, effectively preserving the position information. The calculation formula is as follows:

$$E = LayerNorm\Big(Dropout\Big(T * \sqrt{d} + P\Big)\Big), \tag{1}$$

where E, T, and P are all $n \times d$ matrices, where n represents the length of the sentence sequence, and d represents the dimensionality of the vectors. In the calculation process, a series of operations are performed on the input word vectors. First, we scale the word vector matrix T by a factor of $\sqrt{d}$ to adapt the range of values to the training process of the model, ensuring better convergence. Next, we perform an element-wise addition of the scaled word vector matrix T and the position vector matrix P to introduce positional information of the words in the sentence. With the addition of positional vectors, the model can better distinguish words at different positions in the sentence and more sensitively capture the relationships and dependencies between words in the subsequent feature extraction process.

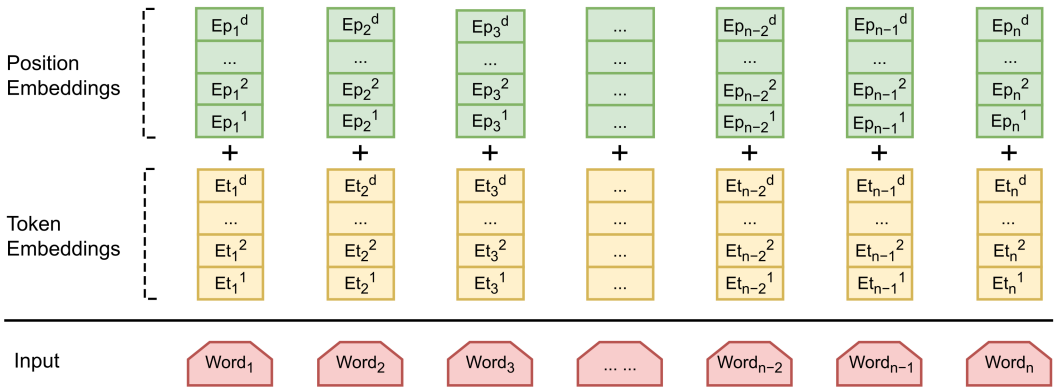

Figure 2. Computational process of input embedding.

To address the issues of overfitting and vanishing gradients, we have incorporated Dropout and LayerNorm techniques. Dropout randomly sets a fraction of the neuron outputs to zero, reducing overfitting in neural networks. LayerNorm, on the other hand, is used to normalize the input tensors, ensuring a standardized distribution of data and further enhancing the stability and generalization ability of the model.

After these steps, our model obtains the feature representation E of the input sequence, which serves as the input for the next module of feature extraction. By scaling the word vectors, introducing positional vectors, and applying Dropout and LayerNorm, we are able to better utilize the information in the text sequence and extract rich semantic features, thereby improving the performance of the model.

2.3. Feature Extraction Process

In order to better learn the local features of text, we introduce convolutional neural networks based on the Attention mechanism. Specifically, we use a hierarchical design in the feature extraction module, allowing the model to better calculate features in different dimensional spaces. Each layer consists of a local feature extractor and a global feature extractor, where the local feature extractor uses convolutional neural networks to extract local features, while the global feature extractor calculates global features through the Attention mechanism. With this hierarchical design, our model can capture both local and global features of the text, thus better understanding the semantic information of the text.

2.3.1. Convolutional Neural Network Layer

Convolutional neural networks (CNNs) have been widely used in both computer vision and natural language processing fields and have become a mature feature extractor. Compared with recurrent neural networks, the sliding window operation of CNNs is order-independent, and different convolutional kernels have no influence on each other, thus exhibiting very high parallel freedom. In addition, by controlling the number of convolutional kernels, it is possible to achieve channel size scaling, thereby achieving the goal of scaling feature dimensions. To take advantage of these benefits of CNNs, we introduce a layered structure based on CNNs in the feature extraction module, mapping the original text features to different feature spaces and reducing the dimensionality of word vectors to reduce the number of parameters and computation at each layer.

The introduction of text convolution also allows the model to better utilize the positional information of the sequence. The feature vectors output by the convolutional layer contain positional information (depending on the convolution order of the kernels). Unlike traditional convolutional structures, we do not include a Max Pooling layer after the convolutional layer (which only retains the maximum value in the feature extraction), as Max Pooling would lead to the loss of crucial positional encoding information in the feature representation. This way, our model can better capture the local features of the text, thereby improving its performance.

In our model, the role of the convolutional layer is to perform convolutional operations on the word embedding matrix to extract corresponding local features. The process of CNN extracting local features is illustrated in Figure 3. In the model implementation, we use multiple convolutional kernels of different sizes to extract n-gram features of different lengths. By setting different kernel sizes, we can capture word combinations of different lengths and obtain richer local feature information which helps to increase the model's robustness and generalization ability. At the same time, the output dimension of the convolutional layer is reduced by adjusting the number of convolutional kernels. The specific calculation process of a CNN is as follows. For each parallel convolutional layer, we perform n-gram convolution between the input sentence word embedding matrix $E \in \mathbb{R}^{n \times d}$ (n is the sentence length, d is the word embedding dimension) and different-sized convolutional kernels $C = [c_1, c_2, \ldots, c_{d/l}]$, generating local feature outputs of different

layers, where $c_i \in \mathbb{R}^{k \times d}$ (k is the kernel size, l is the parallel layer number). The computation process of the feature map matrix $M_l \in \mathbb{R}^{n \times (d/l)}$ can be referred to as Equation (2):

$$M_l = E \circledast C, \tag{2}$$

where $\circledast$ represents the convolution operation of c_i on E. Specifically, the calculation of values in the feature map is shown in Equation (3):

$$m_i = f(c_i * E + b), \tag{3}$$

where f is the nonlinear activation function of the convolution layer and b is the bias term.

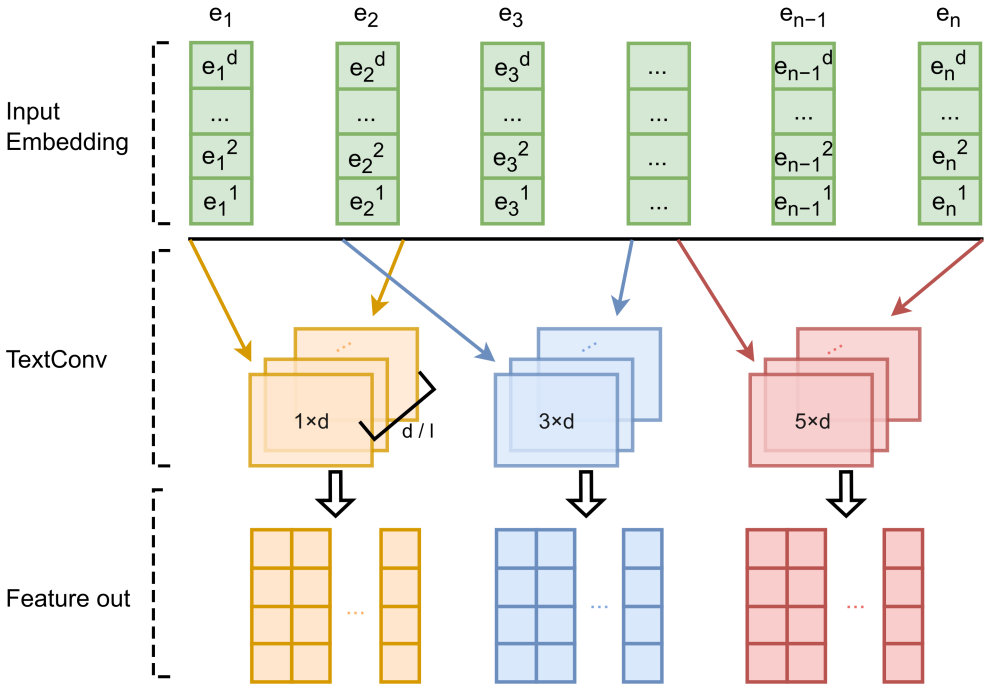

Figure 3. CNN performs local feature extraction and mapping of the input embedding vector.

The convolution operation maps the original text sequence from the word space to a semantic feature space with higher information content. In this way, we can obtain the output matrices of the convolution layers, i.e., $[M_1, M_2, \ldots, M_l]$. These output matrices contain various local features in the input sequence, such as word combinations, phrases, and sentence structures. Through these features, we can better capture the semantic information of the text, thereby improving the performance of text classification tasks.

2.3.2. Attention Mechanism Layer

In text classification tasks, each word contributes differently to the textual features. Therefore, the introduction of attention mechanisms can adjust the weight coefficients (i.e., the values of attention distribution) of key words in self-attention computation, allowing the model to focus more on important information for better results. The incorporation of attention mechanisms enables the model to globally reprocess the locally extracted features obtained through convolution. The attention calculation layer consists of two main components: multi-head attention and a two-layer feed-forward neural network. Among them, multi-head attention is the most critical part, as it captures semantic information from different aspects of the input sequence through multiple attention heads. Adaptive feature weighting and aggregation are performed within each attention head, resulting in the final attention representation. The overall structure of the attention calculation process is illustrated in Figure 4.

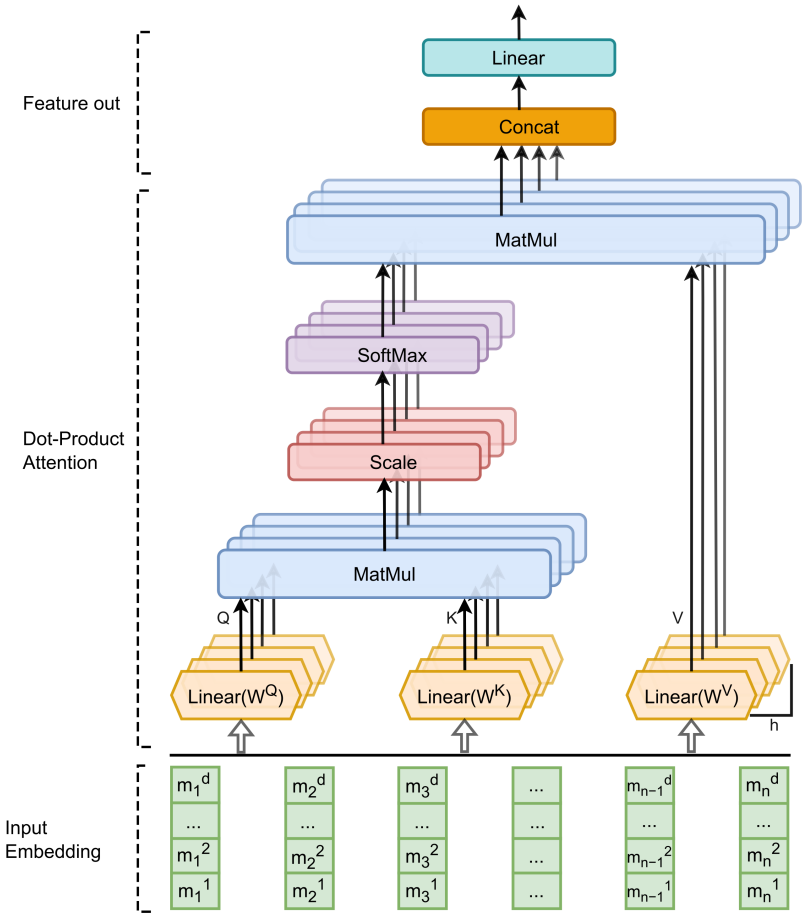

Figure 4. Attention performs global attention calculations.

From the structure depicted in the diagram, it can be observed that Multi-Head Attention performs multiple self-attention computations (multiple heads) on the same input word vectors. This approach aims to capture important features from different subword spaces, thereby enhancing the model's ability to gather diverse information. Self-attention is a mechanism used to calculate the weighted sum of each element based on all elements in the input sequence, enabling the capture of relationships and importance between elements. The computation formula for self-attention is as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

Within each output $head_i$ of self-attention, important features corresponding to different subword spaces can be obtained. By concatenating these outputs and multiplying them with a mapping matrix W^o (aimed at compressing the output matrix), we can obtain the output of the entire Multi-Head Attention:

$$MultiHead\ (Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^o, \tag{5}$$

$$head_i = Attention(Q_i, K_i, V_i), \tag{6}$$

where $Q_i = QW_i^Q$, $K_i = KW_i^K$, $V_i = VW_i^V$, $i \in [1, h]$ and $Q, K, V \in \mathbb{R}^{n \times d/l}$, from the input embedding M_l, mapping matrix $W_i^Q \in \mathbb{R}^{d/l \times d_k}$, $W_i^K \in \mathbb{R}^{d/l \times d_k}$, $W_i^V \in \mathbb{R}^{d/l \times d_v}$, $W^o \in \mathbb{R}^{hd_v \times d/l}$. In this study, we set $h = 4$ parallel Self-Attention layers to perform global attention calculation on the local feature maps (Input Embedding) from the convolutional layer, where $d_k = d_v = d/h = 64$ in each head.

By employing the multi-head attention mechanism, we partition the original high-dimensional feature space into multiple lower-dimensional subspaces and independently compute attention within each subspace. This approach enriches the representation of feature information while reducing the computational burden on the model. Through the attention calculation layer, we obtain the final output matrix $[O_1, O_2, \ldots, O_l]$ of the feature extraction module. This output matrix synthesizes the feature information from multiple low-dimensional subspaces, with each subspace capturing different semantic aspects of the original text. Consequently, it significantly enhances the model's expressive power and improves its robustness and performance when dealing with diverse types of textual data.

2.4. Feature Fusion and Representation

In the task of text classification, both convolutional neural networks (CNNs) and attention mechanisms serve as crucial feature extraction modules, capable of capturing rich local and global features. However, in practical applications, we have observed that fusing local and global features can further enhance the model's performance. Therefore, in this paper, we employ a residual connection to integrate the outputs of the convolutional and attention layers, aiming to strengthen the model's generalization ability. Specifically, we define a residual connection function that adds the outputs of the convolutional and attention layers and feeds them into a fully connected neural network (*FFN*) with two hidden layers, thereby enhancing the model's representational power. The calculation formula for this residual connection function is as follows:

$$FFN(Concat(O_1, O_2, \ldots, O_l)) + Concat(M_1, M_2, \ldots, M_l) \tag{7}$$

In terms of text representation, various methods are commonly employed to transform the input text sequence into a fixed-length vector representation for further processing by the model. Common approaches include taking the maximum or mean value. In the case of the BERT model, a special token "[CLS]" is added to the original input sequence, and after training the model, the vector corresponding to "[CLS]" is extracted as the representation of the entire text. Maximum representation emphasizes key information and may be more effective for tasks that prioritize important keywords or features. The output vector of the "[CLS]" token contains rich semantic information but may not fully consider features from different positions in the text sequence. On the other hand, mean representation focuses more on the fusion of overall features and is suitable for more global tasks. Therefore, in this paper, we choose to perform a mean operation on the word vectors after the residual connection to obtain a vector representation of the entire sentence sequence.

After the aforementioned steps, we obtain higher-level feature representations of the given text sequence. These feature representations are then passed to subsequent fully connected layers, where a series of linear transformations and activation functions are applied to further extract and combine features. Finally, a softmax layer is used for classification prediction, producing a probability distribution over the categories to which the text belongs. The entire process can be viewed as an end-to-end training process, optimizing network parameters through the backpropagation algorithm to maximize prediction accuracy.

2.5. Feed Forward, LayerNorm and Dropout

Our model incorporates several techniques to enhance its performance. These include adding two feed-forward neural networks (FFN) in the text convolutional layer, attention layer, and feature fusion representation layer, as well as utilizing LayerNorm, Dropout, and residual connections. In the feed-forward neural networks, we employ the gelu [49] non-linear activation function, which introduces non-linearity and alters the output space of features, thus improving the model's performance. Specifically, for the input of the FFN, we first map it to a higher-dimensional space using a fully connected layer, apply the gelu activation function, and then map it back to the original dimensionality using another fully

connected layer, as shown in Equation (8). This allows the model to better learn non-linear features and enhance its ability to handle complex data.

$$FFN(x) = gelu(xW_1 + b_1)W_2 + b_2 \tag{8}$$

In addition to *FFN*, we also introduce LayerNorm and Dropout to enhance the robustness and generalization ability of our model. LayerNorm enables the normalization of features for each sample, ensuring similar mean and variance distributions across dimensions, thereby reducing the problem of internal covariate shift. *Dropout* randomly sets a portion of neuron outputs to zero, preventing the model from overfitting to specific input data. Lastly, we employ residual connections, where the original input is added to the features processed by the *FFN*. This can expedite the convergence of the model and mitigate the issue of gradient vanishing. Therefore, the output for each component is given as follows:

$$LayerNorm(x + Dropout(Sublayer(x))) \tag{9}$$

With the series of operations described above, our model is able to obtain richer feature representations, thereby enhancing its performance in handling complex tasks. These techniques not only expedite the convergence speed during training but also effectively prevent overfitting and gradient vanishing issues, thereby strengthening the network's sensitivity to data.

3. Results

To validate the effectiveness of our proposed new model, we conducted experiments on multiple widely used text datasets. We compared the performance of our model against several baseline models to provide a comprehensive evaluation of its capabilities.

3.1. Datasets

Our experiments covered five widely used public datasets, including Yelp Review Polarity, AG's News, Yelp Review Full, and DBpedia. By validating our model on these public datasets, we ensure the reliability and reproducibility of the results. These datasets also provide a relative standard benchmark for evaluating and comparing the performance of different models on specific tasks. Additionally, other researchers can easily access the same datasets for validation purposes. Table 1 presents the basic characteristics of the datasets.

Table 1. Details of all used experimental datasets.

Data Set	Classes	Training Set	Test Set
Yelp Review Polarity	2	560,000	38,000
AG's News	4	120,000	7600
Yelp Review Full	5	650,000	50,000
DBPedia	14	560,000	70,000

The unit size of the data set is rows.

Yelp Review Polarity: The Yelp Review dataset consists of reviews from Yelp. It was curated by Zhang et al. [9] in 2015 from the Yelp Dataset Challenge. This dataset is a binary sentiment classification dataset, where the reviews are categorized as either positive or negative sentiment.

AG's News: This dataset originates from ComeToMyHead, an academic news search engine, and it consists of news articles from over 2000 news sources. The dataset was created by Zhang et al. [9] in 2015 from these news articles and contains 120,000 training samples and 7600 testing samples. Each sample is a short text categorized into one of four classes: World, Sports, Business, and Sci/Tech.

Yelp Review Full: This dataset shares the same origin as the Yelp Review Polarity dataset and was also constructed by Zhang et al. [9]. The Yelp Review Full dataset is a multi-class dataset consisting of 5 categories. In contrast to the Yelp Review Polarity dataset, Yelp Review Full provides a more fine-grained sentiment rating ranging from 1 to 5.

DBPedia: The DBpedia ontology classification dataset was constructed by Zhang et al. [9] based on DBpedia 2014. It is an English ontology classification dataset. The dataset consists of 14 different categories, including Company, EducationalInstitution, Artist, Athlete, OfficeHolder, MeanOfTransportation, Building, NaturalPlace, Village, Animal, Plant, Album, Film, and WrittenWork.

These datasets cover different domains and sources, aiming to thoroughly evaluate the performance of our model in diverse scenarios. These datasets have diverse characteristics, including different data sizes and numbers of categories, providing us with a wide testing environment. Firstly, these datasets have varying data sizes, ranging from small-scale datasets to large-scale datasets. Small-scale datasets help us quickly validate the effectiveness and stability of the model, while large-scale datasets validate the model's generalization ability and test its scalability in handling large-scale problems. Secondly, these datasets cover different numbers of categories. Some datasets involve binary classification tasks, while others involve multi-class classification problems. Such diversity allows us to test the performance of the model in handling different categories and label quantities, helping us comprehensively evaluate the model's ability for diverse classification tasks. By conducting experiments and comparisons on these datasets, we can gain in-depth understanding and evaluation of the applicability and advantages of our proposed model in various scenarios.

Prior to conducting experiments, we performed data preprocessing on the raw datasets to provide clean, standardized, and manageable data, laying the foundation for subsequent experiments and training. After undergoing steps such as data cleaning and tokenization, the raw dataset can be used to construct processed training and testing sets. One crucial step is building the vocabulary. Since our model does not utilize pre-trained parameters, we need to train the model based on the constructed vocabulary. To convert sentences in the training set into numerical representations, we assign an index to each word in the sentence sequence. By building the vocabulary, we iterate through each sentence in the dataset and map each word to a unique integer index. As a result, the original text sequence is transformed into an integer sequence, where each integer represents the index of a word in the vocabulary, serving as input to the model. Through this data preprocessing process, we transform the original text data into numerical representations that are compatible with the model, preparing them for subsequent experiments and training. This also ensures the use of a clear and consistent data representation during model training and evaluation.

3.2. Experiment Settings

3.2.1. Experiment Platform

The experiments in this paper were conducted under the following conditions: an Ubuntu server equipped with an Intel 12th generation Core i7-12700 CPU and 16 GB of memory. The GPU used was an NVIDIA GeForce RTX2080Ti with 11 GB of VRAM. The programming language used was Python 3.7, with PyTorch version 1.13.0 and CUDA version 11.

3.2.2. Model Parameter Configuration

The model used in this study was implemented based on the PyTorch framework. In the Input Embedding layer, word embedding vectors were obtained by adding learnable word vectors and positional vectors, both with a dimension of 768. The maximum supported sequence length for the model was set to 512. The Feature Extraction layer consisted of 1 CNN layer and 3 stacked Attention layers. The CNN had parallel structures with kernel sizes of 1, 3, and 5, while the Attention layer had 4 attention heads. To reduce computational complexity, the word embedding vector dimension of 768 was reduced

to 256 in both the CNN and Attention structures, and after the Feature Fusion layer, the extracted features from each layer were combined back to 768 dimensions. LayerNorm was applied for normalization after the embedding layer, CNN layer, multi-head attention layer, and linear transformation layer. Dropout with a rate of 0.5 was used for regularization. During the training phase, the Adam optimizer was used with an initial learning rate of 1×10^{-3} and weight decay of 1×10^{-4}. The batch size was adjusted based on the dataset size and computer performance. The model was trained for 100 epochs. The activation function used in the model was the gelu linear activation function.

3.2.3. Evaluation Metrics

The model parameters are continuously updated through gradient descent during iterations on the training set. The training set data is used to optimize and update the model parameters, while the test set is used to evaluate the performance of the model without participating in the parameter optimization process. We calculate the accuracy of the model on various test sets and compare it with other baseline models. The formula for calculating accuracy is as follows:

$$\text{Accuracy} = \frac{\text{Correctly Classified Samples}}{\text{Total Number of Samples}} \tag{10}$$

3.3. Comparison with Baseline Models

3.3.1. Baseline Models

To evaluate the performance of our model, we conducted comparative analysis with multiple types of text classification models on the datasets. Through these comparative experiments, we were able to comprehensively assess the performance of our model in different tasks and determine its practicality in the field.

The baseline models we selected are mainly divided into three categories: CNN-based classification models, RNN-based classification models, and attention-based classification models.

- The CNN-based baseline comparison models are mainly Word-level CNN, Char-level CNN [9] and VDCNN [13].
- The RNN-based baseline comparison models are mainly standard LSTM [9] and D-LSTM [26].
- The attention-based baseline comparison models are mainly Bi-BloSAN [50] and LEAM [51].

3.3.2. Comparison Analysis

We compared the classification accuracies of the CAC model and several baseline models on each dataset. The results showed that our proposed model achieved higher accuracy than the other models on most datasets. Table 2 summarizes the accuracy results of different baseline models compared to our proposed model on each dataset. Based on these experimental results, we can conclude that the CAC model has a high performance in text classification tasks and can serve as a strong benchmark model for future research and applications.

Table 2. Classification accuracy (%) of different models on different datasets.

Model	Yelp Review Polarity	AG's News	Yelp Review Full	DBPedia
Word-level CNN	95.40	91.45	59.84	98.58
Char-level CNN	94.75	90.49	61.60	98.34
VDCNN	95.72	91.33	64.72	98.71
LSTM	94.74	86.06	58.17	98.55
D-LSTM	92.60	92.10	59.60	98.70
Bi-BloSAN	94.56	93.32	62.13	98.77
LEAM	95.31	92.45	64.09	99.02
Our Model	96.62	93.64	65.38	98.92

To provide a more intuitive visualization of the performance differences between different models on the datasets, we plotted the comparison results in a bar chart, as shown in Figure 5. This visualization method allows for a clearer view of the classification accuracy performance of different models on different datasets.

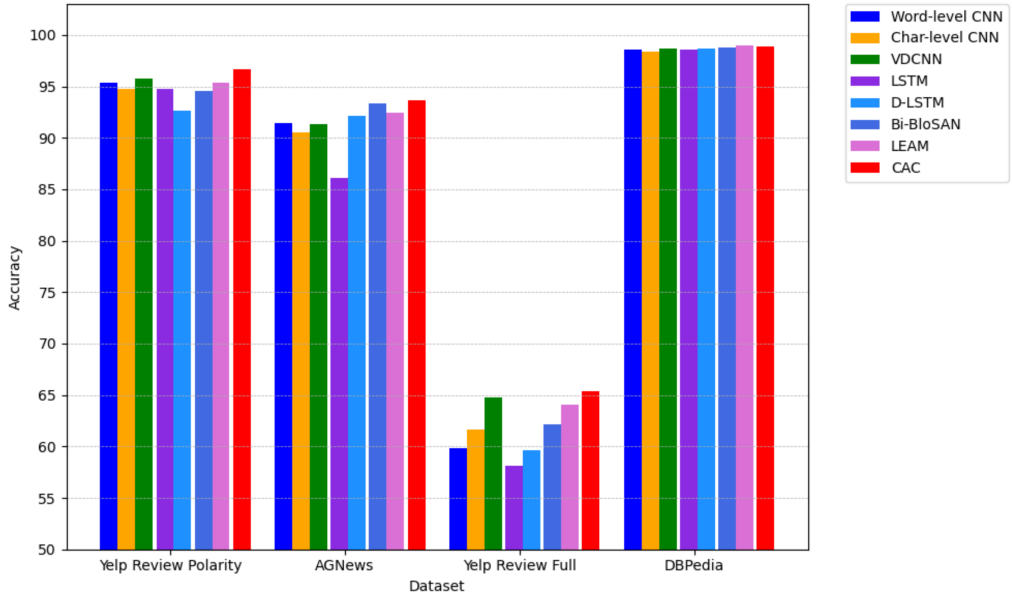

Figure 5. Comparative analysis of the proposed model and model variation on different datasets.

Based on the experimental results above, we can see that our model has a significant advantage in classification accuracy compared to CNN-based text classification models. Our model outperforms both shallow CNN models (word-level and char-level) and deep VDCNN models. This indicates that our model can better learn local features of the text through multi-scale CNN convolution, and better calculate global relationships through attention mechanisms. Additionally, our model is able to capture the local position order of words through CNN and enhance it by adding pre-embedded position vectors, thus better preserving the order information between words.

Compared to RNN-based classification models, our model consistently outperforms standard LSTM and its improved variant (D-LSTM) on all tasks. This suggests that our model is more suitable for extracting sequential features compared to traditional RNN models, which is due to the incorporation of position vectors and CNN that provide advantages in extracting sequence features. In addition, our model introduces a multi-head attention mechanism that better avoids the problem of gradient vanishing during long sequence computation compared to RNN, enabling our model to extract features from long sequences more effectively.

Compared to the attention-based classification models, our model also has a strong competitive advantage. Although LEAM is slightly ahead of our model on the DBpedia dataset, this may be due to the fact that LEAM uses joint training of word and label. However, our model performs stably on most datasets, because our model adopts a hierarchical structure to extract features from different dimensions and uses a local-to-global strategy to achieve better results in semantic feature extraction. In addition, our model can better utilize CNN convolution to learn local features of text and calculate global relationships through attention mechanism, thus achieving better performance.

After conducting a series of comparative experiments, we can conclude that our proposed CAC model has a significant advantage in semantic feature extraction. This is because our model uses a parallel structure to extract features from different dimensions and adopts a strategy of first extracting local features and then computing global relationships, thus better capturing both local and global information of the text. Compared to traditional classification models based on CNN, RNN, or Attention, our model performs

better and achieves higher classification accuracy on most datasets, while also exhibiting good robustness. Therefore, our proposed CAC model has high competitiveness and practical value in text classification tasks.

3.4. Performance Comparison with Transformer

In order to comprehensively evaluate the performance of the CAC model, we conducted an in-depth comparison analysis with the standard Transformer classification model.

3.4.1. Baseline Models

In the experiment, we employed the Encoder structure of the Transformer as the standard feature extractor and used the Concat and Mean methods for feature fusion and representation, which are the same as the CAC model. Regarding parameter settings, we utilized a Transformer structure consisting of 6 stacked Encoders, with each Encoder having 8 attention heads (h = 8). To ensure consistency in the experiment, we adjusted the computation of position embeddings in the Transformer to use learnable position vectors, similar to the CAC model. The word embedding dimension and input sequence length were set to 768 and 512, respectively. We also employed the same data preprocessing methods and dictionary as the CAC model. The number of training epochs was set to 100, and the learning rate was set to 1e-3. Other parameter settings remained consistent with the CAC model.

3.4.2. Results and Analysis

We conducted experiments on the Yelp Review Polarity, AG's News, Yelp Review Full, and DBPedia datasets to calculate and analyze the results of the Transformer model and the CAC model. We plotted the accuracy and average loss curves of the models across each epoch, as shown in Figure 6.

By analyzing the curves of accuracy and loss mentioned above, we can further analyze and explain the advantages of the CAC model over the Transformer model in terms of performance and efficiency.

Firstly, looking at the accuracy curve, our CAC model can converge faster and achieve higher accuracy during the iterations. This indicates that the CAC model is more efficient in learning and adapting to the training data, capturing data features and patterns more quickly. In contrast, the Transformer model has a slower convergence rate and may require more training iterations to reach the same level of accuracy.

Secondly, the curve of loss values demonstrates the effectiveness of our proposed CAC model during the optimization process. As the model iterates, the loss values of the CAC model gradually decrease and stabilize, indicating that the model can better fit the training data and make more accurate predictions in the classification task. On the other hand, the loss values of the Transformer model may decrease at a slower rate, requiring more training time to reach the same level of loss.

Taking all the observations into consideration, we can conclude that the CAC model exhibits better performance and efficiency in handling text classification tasks while maintaining accuracy and reducing computational resources and time consumption. The CAC model leverages CNN structures and position embeddings to capture local features and sequence information more effectively, improving the model's performance. Additionally, the use of multi-head attention mechanisms and residual connections enhances the model's learning capacity and generalization ability. Therefore, the CAC model is a superior choice, providing better performance in text classification tasks.

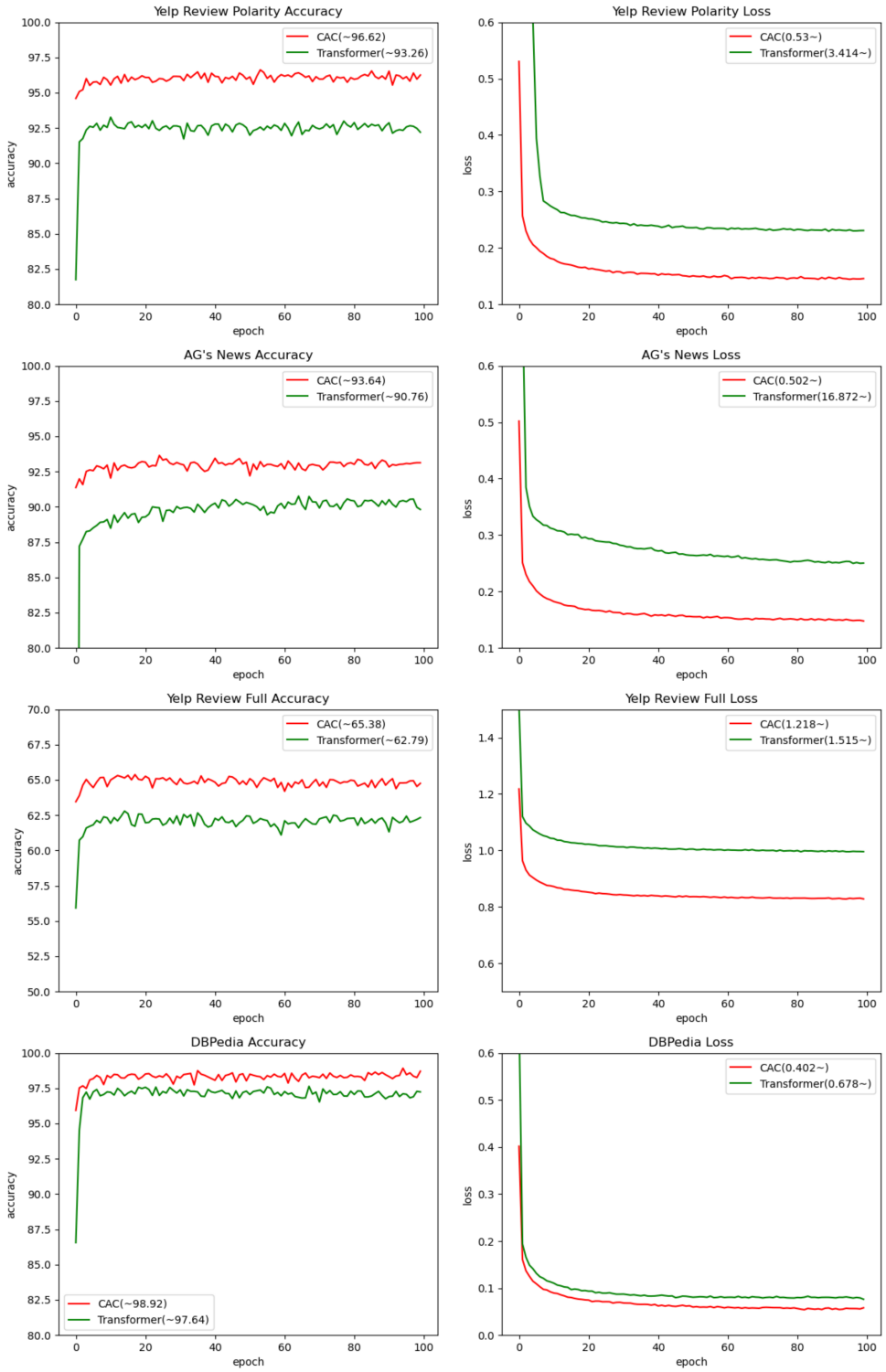

Figure 6. Experimental results of CAC model and Transformer model. The left side is the change of the accuracy rate, and the right side is the change of the loss value.

4. Ablation Study

This section aims to explore the effectiveness of various modules in our proposed model. Thus, we conducted ablation studies based on the AG's News dataset. We individually removed the residual connections between the position vectors, the CNN local feature extraction structure, the output features of CNN, and the output features of the attention

mechanism to observe their impact on model accuracy. The experimental results are shown in Table 3.

Table 3. Ablation studies of the CAC model.

Model	Accuracy
w/o Position Embedding	93.1
w/o CNN	89.81
w/o Residual Connection	90.76
Proposed model	93.64

Removing Position Embedding: The operation of the attention mechanism is actually achieved through linear transformations of matrices, which makes it unable to capture the sequential information of the text when used alone. To address this issue, the concept of position embedding was introduced to enable models based on attention mechanisms to leverage the sequential information of the text. Our proposed model combines the convolutional structure, which allows the model to extract local contextual information from the sentences. Through the convolutional layer, the model can capture the relative positions and sequential relationships of words in the text, thereby compensating for the lack of sequence temporal information that self-attention mechanisms cannot capture without position embeddings. The experimental results in Table 3 show that the accuracy of the model does not decrease significantly when we remove the position embeddings. This suggests that the model can capture sufficient sequence information through the convolutional layer and does not rely on position embeddings. However, when we remove the convolutional layer, there is a noticeable decrease in accuracy, further demonstrating the effectiveness of introducing the convolutional structure. Additionally, position embeddings are usually implemented using fixed-length learnable vectors, which limits the flexibility of the model in handling long sequence inputs. Moreover, position embeddings often introduce additional parameters, and if a large embedding vector length is required for processing long texts, the model's parameter size and computational overhead will significantly increase. Through our experiments, we have shown that the CAC model can completely remove position embeddings when faced with such issues to adapt to more complex tasks without the need for redesigning embeddings. This reduces the complexity of the model and improves its efficiency and scalability.

Removing CNN: In this study, we replaced the original CNN structure with a simple linear mapping matrix that can map the raw input to different dimensional spaces to replace the functionality of CNN. From the results in the table, it can be observed that when the CNN structure is removed, the model's accuracy significantly decreases, indicating the important role of CNN in our model. As mentioned earlier, the CNN structure enhances the model's ability to capture the sequential order, and removing the CNN structure results in a decrease in this capability. The key function of the CNN structure in the model is to extract local features at different scales from the sequence and map the extracted features to different low-dimensional semantic feature spaces, enhancing the attention mechanism's ability to extract local features and enabling the entire model to capture richer semantic information. Without the CNN structure, the model relies solely on the attention mechanism for feature extraction, which greatly impacts the overall feature extraction capability of the model.

Removing Residual Connection: When the residual connection between the output features of the CNN and the output features of the attention mechanism is removed, a noticeable decrease in the model's accuracy is observed. In deep learning, it is generally the case that the deeper the network, the stronger the model's representational power and the better its performance. However, increasing network depth can also introduce optimization-related issues such as vanishing gradients and exploding gradients. To address these issues, we employed methods such as LayerNorm regularization, dropout, and the gelu activation function in the model. Building upon these techniques, we creatively connected the

local features extracted by the CNN and the global features extracted by the attention mechanism, greatly enhancing the model's convergence and generalization capabilities through residual connections. The experiments have demonstrated the effectiveness of this connection method.

5. Conclusions

This paper proposes a new text classification model. As a fundamental task in natural language processing, text classification finds wide applications in various domains, including topic identification in social media, sentiment analysis and opinion mining, content and comment analysis; categorizing articles, blogs, and editorials in the news domain; financial sentiment analysis, market prediction, credit assessment, fraud detection in the financial domain; medical literature classification, disease diagnosis, adverse drug event detection in the medical and biological domain; as well as automated customer service and user support systems in the customer service domain. These are just a few examples of the practical applications of text classification models. In reality, text classification techniques have important applications and impacts in many fields. The purpose of building this new model is to apply it across industries and enhance work efficiency.

We propose a multi-layer feature fusion text classification model called CAC, which combines the advantages of CNN and Attention. This integration enhances the performance of text classification by leveraging the strengths of both approaches. Our main contributions include:

(1) We propose a novel text classification model that combines CNN and Attention for parallel feature extraction. This architecture fully utilizes the advantages of CNN in local feature extraction and Attention in global attention calculation, thereby enhancing the model's feature extraction capability.
(2) We adopt a "local-first, global-later" approach, preserving the sequential information of the text during the feature extraction process and capturing semantic relationships at different levels in the text. Through this approach, our model gains a better understanding of the text's semantics, thereby improving text classification performance.
(3) Extensive experiments have demonstrated that our model achieves outstanding performance on multiple datasets. Compared to methods based on CNN, RNN, and attention models, our model has shown its effectiveness and superiority in terms of accuracy and performance in text classification tasks.

This study successfully applies the proposed architecture to the field of natural language text processing. However, there are still many research directions worth exploring in the future. Since the model incorporates CNN modules, we can further investigate the advantages and potential applications of this model structure in image processing. For example, it may be applicable in image classification, object recognition, object detection, and other areas. By combining the fields of text processing and image processing, we can explore more cross-domain applications. For instance, we can attempt multimodal tasks using our proposed model structure, such as joint analysis and understanding of images and text.

Author Contributions: Conceptualization, H.Y. and S.Z.; methodology, G.Z.; software, S.Z. and L.F.; validation, H.Y., J.W. and S.Z.; formal analysis, S.Z.; investigation, H.S.; resources, H.Y. and H.S.; data curation, H.S.; writing—original draft, H.Y. and S.Z.; writing—review and editing, X.D., J.X., S.S. and H.Z.; visualization, S.Z.; supervision, H.Y.; project administration, H.Y.; funding acquisition, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the School Enterprise Cooperation Project (No.whpu-2021-kj-762 and 1145, No.whpu-2022-kj-1586 and 2153), the Hubei Provincial Teaching and Research Project (No.2018368), and the Ministry of Education Industry-University Cooperation Collaborative Education Project (No.220900786024216).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are available online at: https://huggingface.co/datasets (accessed on 18 June 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text classification algorithms: A survey. *Information* **2019**, *10*, 150. [CrossRef]
2. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.
3. Liu, J.; Wang, X.; Tan, Y.; Huang, L.; Wang, Y. An attention-based multi-representational fusion method for social-media-based text classification. *Information* **2022**, *13*, 171. [CrossRef]
4. Tayal, M.A.; Bajaj, V.; Gore, A.; Yadav, P.; Chouhan, V. Automatic domain classification of text using machine learning. In Proceedings of the 2023 International Conference on Communication, Circuits, and Systems (IC3S), Odisha, India, 26–28 May 2023; pp. 1–5.
5. Nwade, I.; Ozoh, P.; Olayiwola, M.; Musibau, I.; Kolawole, M.; Olubusayo, O.; Adigun, A. Combining text classification with machine learning. *LAUTECH J. Eng. Technol.* **2023**, *17*, 9–17.
6. Jiang, F.; Zhu, Q.; Yang, J.; Chen, G.; Tian, T. Clustering-based interval prediction of electric load using multi-objective pathfinder algorithm and elman neural network. *Appl. Soft Comput.* **2022**, *129*, 109602. [CrossRef]
7. Jiang, F.; Zhu, Q.; Tian, T. An ensemble interval prediction model with change point detection and interval perturbation-based adjustment strategy: A case study of air quality. *Expert Syst. Appl.* **2023**, *222*, 119823. [CrossRef]
8. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
9. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. *Adv. Neural Inf. Process. Syst.* **2015**, *28*.
10. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
11. Umer, M.; Imtiaz, Z.; Ahmad, M.; Nappi, M.; Medaglia, C.; Choi, G.S.; Mehmood, A. Impact of convolutional neural network and fasttext embedding on text classification. *Multimed. Tools Appl.* **2023**, *82*, 5569–5585. [CrossRef]
12. Wang, B.; Sun, Y.; Chu, Y.; Min, C.; Yang, Z.; Lin, H. Local discriminative graph convolutional networks for text classification. *Multimed. Syst.* **2023**, *29*, 2363–2373. [CrossRef]
13. Conneau, A.; Schwenk, H.; Barrault, L.; Lecun, Y. Very deep convolutional networks for text classification. *arXiv* **2016**, arXiv:1606.01781.
14. Johnson, R.; Zhang, T. Deep pyramid convolutional neural networks for text categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 562–570.
15. Wang, S.; Huang, M.; Deng, Z. Densely connected cnn with multi-scale feature attention for text classification. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 4468–4474.
16. Le, H.T.; Cerisara, C.; Denis, A. Do convolutional networks need to be deep for text classification? In Proceedings of the Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
17. Duque, A.B.; Santos, L.L.J.; Macêdo, D.; Zanchettin, C. Squeezed very deep convolutional neural networks for text classification. In Proceedings of the International Conference on Artificial Neural Networks, Munich, Germany, 17–19 September 2019; Springer: Berlin/Heidelberg, Germany; pp. 193–207.
18. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 7370–7377.
19. Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H.; Xu, B. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv* **2016**, arXiv:1611.06639.
20. Johnson, R.; Zhang, T. Supervised and semi-supervised text categorization using lstm for region embeddings. In Proceedings of the International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; pp. 526–534.
21. Dou, G.; Zhao, K.; Guo, M.; Mou, J. Memristor-based lstm network for text classification. *Fractals* **2023**, 2340040. [CrossRef]
22. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
23. Zulqarnain, M.; Ghazali, R.; Ghouse, M.G.; Mushtaq, M.F. Efficient processing of gru based on word embedding for text classification. *Int. J. Inform. Vis.* **2019**, *3*, 377–383. [CrossRef]
24. Huang, Y.; Dai, X.; Yu, J.; Huang, Z. Sa-sgru: Combining improved self-attention and skip-gru for text classification. *Appl. Sci.* **2023**, *13*, 1296. [CrossRef]
25. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv* **2016**, arXiv:1605.05101.
26. Yogatama, D.; Dyer, C.; Ling, W.; Blunsom, P. Generative and discriminative text classification with recurrent neural networks. *arXiv* **2017**, arXiv:1703.01898.

27. Zhang, H.; Xiao, L.; Wang, Y.; Jin, Y. A generalized recurrent neural architecture for text classification with multi-task learning. *arXiv* **2017**, arXiv:1707.02892.
28. Wang, B. Disconnected recurrent neural networks for text categorization. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, 15–20 July 2018; pp. 2311–2320.
29. Luong, M.-T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv* **2015**, arXiv:1508.04025.
30. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
31. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Kamyab, M.; Liu, G.; Adjeisah, M. Attention-based cnn and bi-lstm model based on tf-idf and glove word embedding for sentiment analysis. *Appl. Sci.* **2021**, *11*, 11255. [CrossRef]
33. Prottasha, N.J.; Sami, A.A.; Kowsher, M.; Murad, S.A.; Bairagi, A.K.; Masud, M.; Baz, M. Transfer learning for sentiment analysis using bert based supervised fine-tuning. *Sensors* **2022**, *22*, 4157. [CrossRef] [PubMed]
34. Chen, X.; Cong, P.; Lv, S. A long-text classification method of chinese news based on bert and cnn. *IEEE Access* **2022**, *10*, 34046–34057. [CrossRef]
35. Xu, G.; Chen, S.; Meng, Y.; Zhang, T.; YU, S. Sentiment analysis of Weibo based on global features and local features. *J. South-Cent. Minzu Univ.* **2023**, *42*, 526–534.
36. Qin, J.; Liu, L.; Liu, J.; Ye, Z.; Zhang, Z. Long document retrieval model based on the joint enhancement of BERT and topic model. *J. South-Cent. Minzu Univ.* **2023**, *42*, 469–476.
37. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A c-lstm neural network for text classification. *arXiv* **2015**, arXiv:1511.08630.
38. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language modeling with gated convolutional networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 933–941.
39. Zhu, Q.; Jiang, F.; Li, C. Time-varying interval prediction and decision-making for short-term wind power using convolutional gated recurrent unit and multi-objective elephant clan optimization. *Energy* **2023**, *271*, 127006. [CrossRef]
40. Liu, G.; Guo, J. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing* **2019**, *337*, 325–338. [CrossRef]
41. Dowlagar, S.; Mamidi, R. Multilingual pre-trained transformers and convolutional nn classification models for technical domain identification. *arXiv* **2021**, arXiv:2101.09012.
42. Wenfei, L.; Wei, X.; Dunzhi, W.; Pengcheng, P. Text classification of chinese news based on lstm-attention. *J. South-Cent. Univ. Natl.* **2018**, *37*, 129–133.
43. Zhao, Z.; Wu, Y. *Attention-Based Convolutional Neural Networks for Sentence Classification*; Interspeech: San Francisco, CA, USA, 2016; Volume 8, pp. 705–709.
44. Dowlagar, S.; Mamidi, R. A pre-trained transformer and cnn model with joint language id and part-of-speech tagging for code-mixed social-media text. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), Online, 1–3 September 2021; pp. 367–374.
45. Safaya, A.; Abdullatif, M.; Yuret, D. Bert-cnn for offensive speech identification in social media. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain, 12–13 December 2020.
46. Gulati, A.; Qin, J.; Chiu, C.-C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y. Conformer: Convolution-augmented transformer for speech recognition. *arXiv* **2020**, arXiv:2005.08100.
47. Paun, G. Membrane computing. *Scholarpedia* **2010**, *5*, 9259. [CrossRef]
48. Zhang, G.; Pan, L. A survey of membrane computing as a new branch of natural computing. *Chin. J. Comput.* **2010**, *33*, 208–214. [CrossRef]
49. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv* **2016**, arXiv:1606.08415.
50. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Zhang, C. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv* **2018**, arXiv:1804.00857.
51. Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; Carin, L. Joint embedding of words and labels for text classification. *arXiv* **2018**, arXiv:1805.04174.