Computer Science, Technology and Applications

Attention Augmented Learning Machines

Theory and Applications

Guoqiang Zhong
Jinxuan Sun
Editor

NOVA

Attention Augmented Learning Machines

Guoqiang Zhong and Jinxuan Sun

Attention Augmented Learning Machines

Theory and Applications

nova
science publishers
New York

Library of Congress Cataloging-in-Publication Data

ISBN: 979-8-88697-780-6
PDF ISBN: 979-8-89113-161-3
ePUB ISBN: 979-8-89113-162-0

Published by Nova Science Publishers, Inc. † New York

Contents

Preface

Deep learning has developed for more than 10 years. Many novel models are proposed. Among others, the attention models have greatly impacted the deep learning area. Similar to the attention mechanism of human beings, the attention mechanism improves the performance of many deep learning models based on its discovery of important information hidden in data and motivates the emergence of many new deep learning models, like Transformer and its variants.

This book includes eight chapters and aims to introduce some interesting works on the attention mechanism. Chapter 1 is a review of the attention mechanism used in the deep learning area, while Chapters 2 and 3 present two models that integrate the attention mechanism into gated recurrent units (GRUs) and long short-term memory (LSTM), respectively, making them pay attention to important information in the sequences. Chapter 4 designs a multi-attention fusion mechanism and uses it for industrial surface defect detection. Chapter 5 enhances Transformer for object detection applications. Moreover, Chapter 6 proposes a dual-path architecture called dual-path mutual attention network (DPMAN) for medical image classification, and Chapter 7 proposes a novel graph model called attention-gated graph neural network (AGGNN) for text classification. In addition, Chapter 8 combines the generative adversarial networks (GAN), LSTM, and the attention mechanism to build a generative model for stock price prediction. These chapters introduce new designs of the attention mechanism and demonstrate their effectiveness using extensive experiments and ablation studies on various applications.

This book can be used by college students (undergraduate or graduate) chosen to major in computer science, artificial intelligence, electrical engineering, and mathematics, or others who study or have the potential to use deep learning algorithms. It could be of special interest to professors who research pattern recognition, machine learning, computer vision, neural language processing, and related fields, or engineers who apply deep learning models to their products. On the other hand, the reader is assumed to be already familiar with basic computer programming, machine learning, pattern recognition, and computer vision.

The works presented in this book are mainly supported by the National Key Research and Development Program of China under Grant No. 2018AAA0100400, HY Project under Grant No. LZY2022033004, the Natural Science Foundation of Shandong Province under Grants No. ZR2020MF131 and No. ZR2021ZD19, Project of the Marine Science and Technology Cooperative Innovation Center under Grant No. 22-05-CXZX-04-03-17, the Science and Technology Program of Qingdao under Grant No. 21-1-4-ny-19-nsh, and Project of Associative Training of Ocean University of China under Grant No. 202265007. We want to thank Qingdao AI Computing Center and Eco-Innovation Center for providing inclusive computing power and technical support for MindSpore during the completion of this book. Moreover, we would like to give our special thanks to Zhaoyang Niu and Jiajia Dong, who provided essential aid to edit this book, and all chapter authors, who performed research and delivered so many interesting

ideas and wonderful works. Last, but not least, we thank all members of the Intelligent Data Analysis and Systems Lab, who are just like our family members, always being friendly and supportive. There are many teachers, collaborators, colleagues, and friends. We are sorry as we cannot list all the names here.

In addition, due to the knowledge and time limitations, there may be some mistakes in this book. We hope that the readers can help point them out. Any comments and suggestions are welcome (E-mail address: gqzhong@outlook.com).

Guoqiang Zhong and Jinxuan Sun

January 29, 2023

Book Description

The attention mechanism is currently a hot topic in the deep learning, computer vision, and natural language processing areas. Many attention models have been proposed, and they are applied to various tasks, such as text classification, image recognition, and graph analysis. This book includes eight chapters and aims to introduce some interesting works on the attention mechanism. Chapter 1 is a review of the attention mechanism used in the deep learning area, while Chapter 2 and Chapter 3 present two models that integrate the attention mechanism into gated recurrent units (GRUs) and long short-term memory (LSTM), respectively, making them pay attention to important information in the sequences. Chapter 4 designs a multi-attention fusion mechanism and uses it for industrial surface defect detection. Chapter 5 enhances Transformer for object detection applications. Moreover, Chapter 6 proposes a dual-path architecture called dual-path mutual attention network (DPMAN) for medical image classification, and Chapter 7 proposes a novel graph model called attention-gated graph neural network (AGGNN) for text classification. In addition, Chapter 8 combines the generative adversarial networks (GANs), LSTM, and an attention mechanism to build a generative model for stock price prediction.

Chapter 1

The Attention Mechanism Used in the Deep Learning Area

Jiajia Dong[1], Guoqiang Zhong[1,*], Zhaoyang Niu[1] and Hui Yu[2]

[1] College of Computer Science and Technology, Ocean University of China, Qingdao, China

[2] School of Creative Technologies, University of Portsmouth, UK

Abstract

Attention is arguably one of the most powerful concepts in the deep learning field nowadays. It is inspired by the human biological system, which focuses on the distinctive parts when processing large amounts of information. With the development of deep neural networks, attention mechanism has been widely used in diverse application domains. In this chapter, we review the different neural network structures that integrate attention and define a unified model suitable for most attention structures. Specifically, we propose a taxonomy of attention models according to four dimensions: positions, forms, input representation, and output representation. Furthermore, we introduce specific implementation forms of the attention mechanism in diverse application domains. In addition, we discuss the interpretability that attention brings to deep neural networks to a certain extent and prospects in the deep learning area.

Key Words: Attention mechanism, Deep learning, Recurrent neural network (RNN), Convolutional neural network (CNN), Encoder–decoder, Unified attention model, Computer vision applications, Natural language processing

1.1 Introduction

Attention is a complex cognitive function indispensable for human beings [1, 2]. One important property of human perception is that one does not tend to process whole information at once. Instead, humans tend to selectively concentrate on a part of the information when and where it is needed, while ignoring other perceivable information. For instance, when humans perceive things, we usually do not see all the scenes from the beginning to the end but observe and pay attention to specific parts according to our needs. When humans find that a scene often has something they want to observe in a certain part, they will learn to focus on that part when

* E-mail address: gqzhong@ouc.edu.cn

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

similar scenes appear again in the future and focus more attention on the useful part. This is a means for humans to quickly select high-value information from massive information using limited processing resources. The human biological attention mechanism greatly improves the efficiency and accuracy of information processing.

A human's attention mechanism can be divided into two types according to its generation method [3]. The first is bottom-up unconscious attention, called saliency-based attention. Saliency-based attention driven by external stimuli does not require active intervention. Therefore, it has nothing to do with the task similar to the max-pooling and gating [4, 5] mechanisms in deep learning. The second type is top-down conscious attention, called focused attention. Focused attention refers to the attention that has a predetermined purpose and relies on specific tasks. It enables humans to focus attention on a certain object consciously and actively. Most of the attention mechanisms in deep learning are designed according to specific tasks, so most of them are focused attention. The attention mechanism introduced in this chapter usually refers to focused attention except for special statements.

In the case of information overload, the attention mechanism uses limited computing resources to process more important information as a resource allocation scheme. Some researchers also bring attention to the computer vision area. Some researchers also bring attention to the computer vision area. Itti et al. [6] proposed a saliency-based visual attention model that extracts local low-level visual features to get some potential salient regions. In the neural network, Mnih et al. [7] used the attention mechanism on the recurrent neural network (RNN) model to classify images. Bahdanau et al. [8] use the attention mechanism to simultaneously perform translation and alignment on machine translation tasks. Subsequently, attention mechanism has become an increasingly common ingredient of neural architectures and has been applied to various tasks, such as image caption generatoin [9, 10], text classification [11], machine translation [12–15], action recognition [16–18], sentiment analysis [19], speech recognition [20–22], recommendation [23, 24], and graph data analysis [25, 26].

In addition to providing performance improvements, the attention mechanism can also be used to explain incomprehensible neural architecture behavior. The lack of interpretability will constitute both a practical and ethical issue for the application of neural networks in medical, financial, military, insurance, and other industries [27]. However, the interpretability of deep learning has been a problem so far. Although whether the attention mechanism can be used as a reliable method to explain deep networks is still a controversial issue [28, 29], it can provide an intuitive explanation to a certain extent [30–33]. For example, the weights computed by attention mechanism could reflect the correlation between input and output.

This survey is structured as follows. In Section 1.2, we summarize network architectures applicable to the attention mechanism. In Section 1.3, we define a general attention model by introducing a well-known model proposed by Bahdanau et al. [8] as an instance. Section 1.4 presents our taxonomy of attention models. Section 1.5 elaborates on the uses of the attention mechanism in various computer vision (CV) and natural language processing (NLP) tasks. In Section 1.6, we discuss the interpretability that brings attention to deep neural networks to a certain extent. In Section 1.7, we conclude this chapter.

1.2 Network Architectures with Attention

In this section, we elaborate on three salient neural network architectures used in conjunction with attention.

1.2.1 Sequence to Sequence

Here, we briefly describe the *Sequence-to-Sequence* (also called RNN Encoder–Decoder) architecture proposed by Cho et al. [4], and Sutskever et al. [14] used it in machine translation tasks. This architecture consists of two recurrent neural networks (RNNs) that act as an encoder and a decoder pair.

The encoder maps a variable-length source sequence $x = (x_1, \cdots, x_{T_x})$ to a fixed-length vector *c*. The most common approach is to use an RNN such that

$$h_t = f\left(x_t, h_{t-1}\right), \tag{1.1}$$

and

$$c = q\left(\left\{h_1, \cdots, h_{T_x}\right\}\right), \tag{1.2}$$

where h_t is a hidden state at time step *t*, and *c* is a context vector generated from the sequence of the hidden states. *f* and *q* are nonlinear activation functions. For example, *f* may be as simple as an element-wise logistic sigmoid function and as complex as a gated recurrent unit (GRU) [4] or long short-term memory (LSTM) [5], while *q* may be as the hidden state h_{T_x} of the last time step.

The decoder is trained to generate a variable-length target sequence $y = (y_1, \cdots, y_{T_y})$ by predicting the next symbol y_t given the hidden state h_t. However, unlike the encoder, both y_t and h_t are also conditioned on y_{t-1} and on the context vector *c*. Hence, the hidden state of the decoder at each time step *t* is computed by

$$h_t = f(h_{t-1}, y_{t-1}, c). \tag{1.3}$$

And each conditional probability is modeled as

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, h_t, c). \tag{1.4}$$

The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. Since this Encoder–Decoder structure does not limit the length of the input and output sequence, it has a wide range of applications, such as image and video captioning, visual question answering, and speech recognition. Moreover, the encoder and decoder can also be composed of other architectures, not necessarily RNNs. Although the attention model can be regarded as a general idea and does not depend on a specific framework itself [7, 34, 35], most attention models are currently accompanied by the Encoder–Decoder framework.

1.2.2 Memory Networks

The memory network [36–39] saves some task-related information in the auxiliary memory by introducing external auxiliary memory units, and then reads it when needed. It not only effectively increases the network capacity but also improves the network computing efficiency. A well-known example is a differentiable end-to-end memory network architecture proposed by Sukhbaatar et al. [38], which can read information from external information multiple times.

The core idea is to convert the input set into two external memory units: one for addressing by comparing the correlation with query and the other for output. End-to-end memory networks can be regarded as a form of attention: the key-value pair attention mechanism. Unlike the usual attention, instead of modeling attention only over a single sequence, they use two external memory units to model it over a large database of sequences.

1.2.3 Networks without RNNs

Encoder–Decoder architectures-based RNNs typically factor computation along the symbol positions of the input and output sequences. This inherently sequential nature results in computational inefficiency, as the processing cannot be parallelized. On the other hand, the computational complexity of establishing long-distance dependence for a sequence of length n through RNN is $O(n)$. To address these problems, researchers try to abandon RNNs in network architectures.

In contrast to the fact that recurrent networks maintain a hidden state of the entire past, convolutional networks do not rely on the computations of the previous time step, so it allows parallel computing of each element in a sequence. Gehring et al. [40] proposed an architecture based entirely on *convolutional neural networks*. This architecture enables the network to capture long-distance dependencies by stacking multiple layers of CNN; the computational complexity becomes $O(n / k)$ for a multi-layer CNN with a convolution kernel size of k. Moreover, this convolution method can discover the compositional structure in the sequence more easily because of its hierarchical presentations.

Vaswani et al. [15] proposed another network architecture, the *Transformer*, which relies entirely on a self-attention mechanism to compute representations of its input and output without resorting to RNNs or CNNs. The Transformer is composed of two components: position-wise feed-forward network (FFN) layer and multi-head attention layer. Position-wise FFN is a fully connected feed-forward network, which is applied to each position separately and identically. This method can ensure the position information of each symbol in the input sequence during the operation. Multi-head attention allows the model to focus on information from different representation subspaces from different positions by stacking multiple self-attention layers, just like multiple channels of CNN. In addition to being more parallelizable, the time complexity of establishing long-distance dependence through the self-attention mechanism is $O(1)$.

Additionally, since there are no RNNs in the above architectures, they cannot learn where they are in the sequence through the recurrent hidden state computation. To this end, they introduced *position embeddings*, which are a combination of positional encodings and input embeddings. There are many choices of positional encodings, learned and fixed [40]. And the overall structure of these models is still Encoder–Decoder.

1.3 Attention Mechanism

In this section, we first describea well-known machine translation architecture using attention introduced by Bahdanau et al. [8] and take it as an instance to define a general attention model.

1.3.1 An Example of an Attention Model

Bahdanau et al. [8] applied the attention mechanism to the task of machine translation, where their model is called *RNNsearch*. The models proposed before that for neural machine translation often belong to a family of encoders–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. A potential issue with the architecture is that a neural network needs to be able to compress all necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those longer than the sentences in the training corpus. The authors conjecture that using a fixed-length vector is a bottleneck in improving the performance of this basic Encoder–Decoder architecture.

In order to address this issue, they propose a novel architecture consisting of a bidirectional RNN [41] as an encoder and a decoder that emulates searching through a source sentence during decoding a translation, as shown in Figure 1.1.

The encoder calculates *annotations* $(h_1,\ldots,h_T)$, which are the hidden state of the BiRNN based on the input sequence $(x_1,\ldots,x_T)$,

$$(h_1,\ldots,h_T) = \text{BiRNN}(x_1,\ldots,x_T). \tag{1.5}$$

The decoder emulates searching through a source sentence during decoding the translation, which consists of the attention function and an RNN. At each time step t, the output y_t is determined jointly by a *context vector* c_t provided by the attention function and the hidden state s_t of the subsequent RNN

$$p(y_t \mid y_1,\ldots,y_{t-1},x) = g(y_{t-1},s_t,c_t). \tag{1.6}$$

The context vector c_t is computed as a weighted sum of these annotations h_j:

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j. \tag{1.7}$$

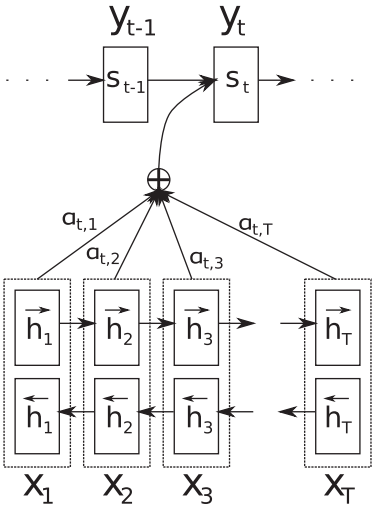


Figure 1.1 Illustration of a single step of decoding in attention-based neural machine translation [8].

The attention weight α_{tj} of each annotation h_j is computed by

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}. \tag{1.8}$$

where

$$e_{tj} = a(s_{t-1}, h_j), \tag{1.9}$$

and a is a learnable function that reflects the importance of the annotation h_j to the next hidden state s_t according to the state s_{t-1}. Using this new approach of introducing an attention mechanism, the information of the source sentence can be distributed in the entire sequence instead of encoding all information into a fixed-length vector through the encoder, and the decoder can selectively retrieve it at each time step. This formulation enables the neural network to focus more on relevant elements of the input than on irrelevant parts.

1.3.2 A Unified Attention Model

Figure 1.2 shows the unified attention model we defined, which comprises a core part shared by almost all models found in the surveyed literature. The implementation process of the attention mechanism can be divided into two steps: one is to compute the attention distribution on all input information, and the other is to compute the weighted average of the input information according to the attention distribution.

When computing the attention distribution, the neural networks encode the source data feature as K, called *key*. K can be expressed in various representations according to different tasks and neural architectures. For instance, K may be features of a certain area of an image, word embeddings of a document, or the hidden states of recurrent neural networks, as it appears with the annotations in RNNsearch. Moreover, it is usually necessary to introduce a task-related representation vector q, the *query*, similar to that in the previous hidden state of the output s_{t-1} in RNNsearch. q can also be in the form of a matrix [15] or two vectors [43] according to specific tasks.

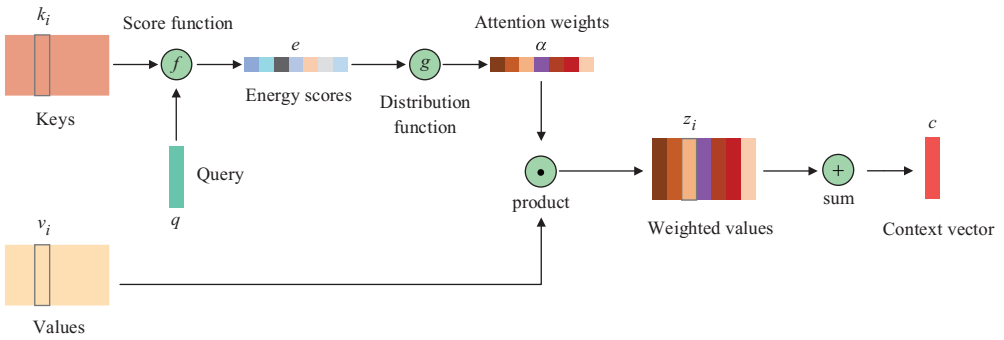


Figure 1.2 The architecture of the unified attention model.

Then, we can compute the correlation between queries and keys through the *score function* f to obtain the *energy score* e, which reflects the importance of queries with respect to keys in deciding the next output,

$$e = f(q,k). \tag{1.10}$$

The score function f is a crucial part of the attention model because it defines how keys and queries are matched or combined. Other authors also call it an energy function [44] or compatibility function [45].

In Table 1.1, we list some common score functions. The two most commonly used attention mechanisms are *additive* attention (like the *alignment model* in RNNsearch) and computationally less expensive *multiplicative* (*dot-product*) attention [13]. Britz et al. [12] empirically compared these two score functions. Moreover, Vaswani et al. [15] proposed a variant of multiplicative attention by adding the scaling factor of $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of keys. While for small values of d_k the two mechanisms perform similarly, additive attention outperforms multiplicative attention without scaling for larger values of d_k. Besides multiplicative attention, Luong et al. [13] also presented general attention, concat attention, and location-based attention. *General* attention extends the concept of multiplicative attention by introducing a learnable matrix parameter W, which can be applied to keys and queries with different representations. *Concat* attention is a different approach that aims to use the joint representation of the keys and queries combined instead of comparing them. It is similar to additive attention except for computing q and K separately. In *location-based* attention, the alignment scores are computed solely from the target hidden state. In other words, the energy scores are only related to q and not to K. Conversely, *self*-attention [46] may be computed only based on K, without any q. Sordoni et al. [43] added a learnable bias that allows bias in the attention mechanism toward certain symbols that tend to be important across the questions independently of the search key, called *bias* attention. Graves et al. [36] presented a model that compares the *similarity* between K and q using the cosine similarity.

Table 1.1 Summary of score function *f.* v,b,W,W_1,W_2 are learnable parameters, and D is the dimension of the input vector. *act* is a nonlinear activation function, such as tanh and ReLU.

Name	Equation	Ref.
additive	$f(q,k) = v^{\mathsf{T}}\, act\,(W_1 k + W_2 q + b)$	[12]
multiplicative (dot-product)	$f(q,k) = q^{\mathsf{T}} k$	[12]
scaled multiplicative	$f(q,k) = \frac{q^{\mathsf{T}} k}{\sqrt{D}}$	[15]
general	$f(q,k) = q^{\mathsf{T}} W k$	[13]
concat	$f(q,k) = v^{\mathsf{T}}\, act\,(W[k;q] + b)$	[13]
location-based	$f(q,k) = f(q)$	[13]
bias	$f(q,k) = k^{\mathsf{T}}(Wq + b)$	[43]
similarity	$f(q,k) = \frac{k \cdot v}{\lVert k \rVert \cdot \lVert v \rVert}$	[36]

After that, energy scores (e) are mapped to attention weights (a) through attention distribution function (g)

$$a = g(e), \tag{1.11}$$

The distribution function g corresponds to the softmax in RNNsearch, which normalizes all energy scores to a probability distribution. This method is the most common method, and some researchers have modified it according to specific tasks. A limitation of the softmax function is that the resulting probability distribution always has full support, *i.e.*, $\text{softmax}(z) > 0$ for every term of z . This is a disadvantage in applications where a spares probability distribution is desired, in which case, Martins and Astudillo [47] proposed *sparsemax* that may assign exactly zero probability to some of its output variables. Kim et al. [48] used another distribution function, the logistic sigmoid, which scales energy scores between 0 and 1. And the two functions perform better or worse on different tasks in the author's experiment. Besides, there are many attention architectures based on positions and their weight allocation methods are different, which will be explained in Section 1.4.1.

Next, the attention weight a is combined with the values V to obtain a new weighted representation Z, with

$$z_i = a_i v_i. \tag{1.12}$$

Each element of V corresponds to one and only one element of K. And in many architectures, the two are the same representation of the input data, just like the annotations in RNNsearch. Based on previous work [49–52], Daniluk et al. [53] hypothesized that such overloaded use of these representations makes training the model difficult and proposed a modification to the attention mechanism that explicitly separates these functions. They use different representations of the input to compute the attention distribution and the contextual information. In other words, V and K are different representations of the same data in their *key-value pair* attention mechanism. In particular, Q, K, and V are three different representations of the same data in the self-attention mechanism [15].

The last step is to combine the new weight representation Z to generate the *context vector c*

$$c = \sum_{i=1}^{n} z_i. \tag{1.13}$$

The above describes the most common architectures in the attention model. Quoting Vaswani et al. [15], the attention mechanism "can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key."

1.4 Taxonomy of the Attention Mechanism

We divide the attention mechanism into four categories according to four aspects. In this section, we introduce the different types of attention mechanism in detail in several seminal papers.

In addition, we would like to emphasize that these categories are not mutually exclusive, and there may be multiple types of attention in an attention model (as shown in Table 1.2).

1.4.1 Positions

The attention proposed by Bahdanau et al. [8], as mentioned above, belongs to *soft* (deterministic) attention, which uses a weighted average of all keys to build the context vector. For soft attention, the attention module is differentiable with respect to the inputs, so the whole system can still be trained by standard back-propagation methods.

Correspondingly, the *hard* (stochastic) attention is proposed by Xu et al. [10], where the probability distribution over weights is used to stochastically sample keys as context vector. In this way, the distribution function in Section 1.3.2 may be replaced by the following:

$$\tilde{a} = g(e), \tag{1.14}$$

and

$$a \sim Multinoulli(\{\tilde{a}_i\}). \tag{1.15}$$

Since making a hard decision at every position of the input renders the resulting module non-differentiable and difficult to optimize, the whole system can be trained by maximizing an approximate variational lower bound or equivalently by REINFORCE [56].

Luong et al. [13] proposed the *global* attention and *local* attention mechanism for machine translation. Global attention is similar to the soft attention. Local attention can be viewed as an interesting blend between the hard and soft attention, in which only a subset of source words is considered at a time. This approach is computationally less expensive than global or soft attention. At the same time, unlike hard attention, this approach is differentiable almost everywhere, making it easier to implement and train.

In addition, Hermann et al. [57] proposed two attention models for natural language reading comprehension tasks, the attentive reader and the impatient reader corresponding to *static* attention and *dynamic* attention, respectively. Static attention computes a weight distribution probability for the whole sentence, while dynamic attention computes weight distribution probability for each token dynamically, like soft attention mechanism.

1.4.2 Forms

The attention mechanisms can be divided into *item-wise* attention and *location-wise* attention, according to the forms of the input sequence for different tasks. The *item-wise* attention can be used in tasks where each input item is given or extracted. For example, the item could be a vector, a matrix, or feature map. However, *location-wise* attention is related to tasks where input items are hard to get. Therefore, we accept a feature map (like vision tasks) or make a transformation on that feature map to help the encoder utilize it at a later phase.

Another difference is how it is calculated when combined with soft/hard attention. The item-wise soft attention calculates a weight for each item and then makes a linear combination. The location-wise soft attention accepts an entire feature map as input and generates a transformed version through the attention module. Instead of a linear combination of all items, the

Table 1.2 Examples of combinations between different categories of attention. ‘ - ’ means not applicable.

Reference	Application	Category			
		Positions	Froms	Input Representation	Output Representation
Bahdanau et al. [8]	Machine Translation	soft	item-wise	distinctive	-
Mnih et al. [7]	Image Classification	hard	location-wise	distinctive	-
Vaswani et al. [15]	Machine Translation	soft	item-wise	distinctive	multi-head
Hu et al. [54]	Image Classification & Object Detection	soft	item-wise	distinctive	-
Lu et al. [42]	Visual Question Answeringn	soft	item-wise	co-attention & hierarchical input	-
Shen et al. [55]	Language Understandingg	soft	item-wise	distinctive	multi-dimensional
Jaderberg et al. [34]	Image classification	soft	location-wise	distinctive	-
Yang et al. [46]	Document Classification	soft	item-wise	hierarchical input	-

item-wise hard attention stochastically picks one or some items based on their probabilities. The location-wise hard attention stochastically picks a sub-region as input, and the location of the sub-region to be picked is calculated by the attention module.

1.4.3 Input Representation

There are two features about the input representation in most of the attention models mentioned above:

(1) These models include a single input and the corresponding output sequence;
(2) The keys and queries belong to two independent sequences.

This kind of attention is called *distinctive* attention [58]. In addition, the attention mechanism has many different forms of input representation.

Lu et al. [42] presented a multi-inputs attention model for visual question answering task, the *co-attention*, that jointly reasons about image and question attention. And co-attention can be performed in parallel or alternatingly. The former generates an image and question attention simultaneously, and the latter sequentially alternates between generating an image and question attentions. Furthermore, co-attention can be coarse-grained or fine-grained [59]. Coarse-grained attention computes attention on each input, using an embedding of the other input as a query. Fine-grained attention evaluates how each element of an input affect each element of the other input. And co-attention has been used successfully in a variety of tasks, including sentiment classification [60], text matching [61], named entity recognition [62], entity disambiguation [63], emotion cause analysis [64], and sentiment classification [65].

Wang et al. [66] presented an *inner* (*self*) attention that computes attention only based on the input sequence. In other words, the query, key, and value are different representations of the same input sequence. Such a model has proven to be effective by several authors, who have exploited it in different ways [15, 67–71]. A well-known application is Transformer [15], the first

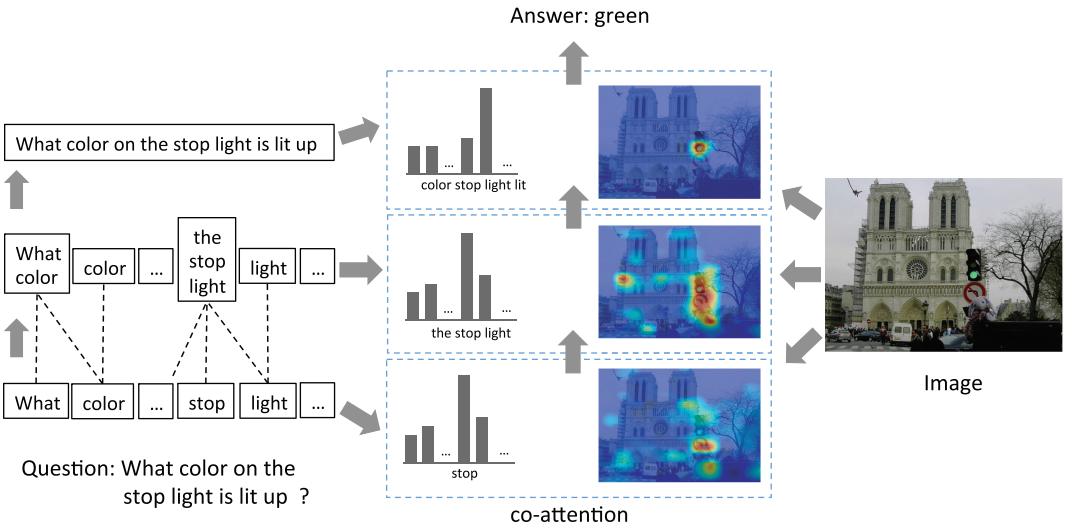


Figure 1.3 An example of hierarchical attention and co-attention [42].

sequence transduction model based entirely on self-attention without RNNs. Applications of this model in different fields will be described in Section 1.5.

Attention weight can be computed not only from the original input sequence but also from different abstraction levels, which we refer to as *hierarchical-input* attention. Yang et al. [46] proposed a hierarchical attention network (HAM) for document classification, which has two levels of attention mechanisms applied at the word level and sentence level. The hierarchical-input attention allows the HAM to aggregate important words into a sentence and then aggregate important sentences into a document. And the hierarchy can be extended further. Wu et al. [72] added a user level on top, applying attention at the document level. Contrary to the above model weight learning from a lower to a higher level, the model proposed by Zhao and Zhang [44] also uses hierarchical-input attention, but the attention weight is learned from a higher to a lower level. Except for natural language processing, hierarchical-input attention can also be used in computer vision. For instance, Xiao et al. [73] proposed a two-level attention method: object-level and part-level attention. It is the first fine-grained image classification method that does not use additional part information and only relies on the model to generate attention weight.

1.4.4 Output Representation

In many applications of convolutional neural networks (CNNs), it has been proved that multiple channels can express input data more comprehensively. Also, in attention models, in some cases, using single attention distribution of the input sequence may not suffice for downstream tasks. Vaswani et al. [15] proposed *multi-head* attention that linearly projects the input sequence (Q, K, V) to multiple subspaces according to learnable parameters, then applies scaled dot-product attention to its representation in each subspace, and finally concatenates their output. This allows the model to jointly attend to information from different representation subspaces at different positions. Li et al. [74] proposed three disagreement regularizations to augment the multi-head attention model on the subspace, the attended positions, and the output representation. This approach encourages diversity among attention heads so that different heads can learn distinct features.

Another approach is the *multi-dimensional*, proposed by Shen et al. [55], which computes a feature-wise score vector for k_i by replacing weight scores vector a with a matrix A. This is especially useful for natural language processing where word embeddings suffer from the polysemy problem. Lin et al. [67] and Du et al. [75] have successfully applied it to various tasks, including author profiling, sentiment classification, textual entailment, and distantly supervised relation extraction. Furthermore, they apply regularization penalties to enforce the distinction between models of relevance Eq. 1.16

$$P = \left\| (AA^{\mathsf{T}} - I) \right\|_F^2, \tag{1.16}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix and I is an identify matrix.

Besides, Barbieri et al. [76] proposed *label-wise* attention, which computes a separate attention distribution for each label. In addition to improving performance, this method can provide a degree of interpretability about how different features are used for predictions.

1.5 Applications

In previous sections, we have shown that attention models have been widely applied to various tasks. Here we introduce some specific applications of attention models in computer vision and natural language processing. Moreover, we do not offer a comprehensive account of all neural architectures that use attention mechanisms, and that would be impossible and rapidly become obsolete. In the following, we focus on several general attention methods.

1.5.1 Computer Vision

In this part, we introduce attention mechanisms in computer vision through several typical papers in different dimensions of neural architectures.

Spatial attention allows neural networks to learn where they should be focused. This attention mechanism transforms the spatial information in the original picture into another space and the key information is retained. Mnih et al. [7] presented a spatial attention model that is formulated as a single RNN, which takes a glimpse window as its input and uses the internal state of the network to select the next location to focus on and to generate control signals in a dynamic environment. Jaderberg et al. [34] introduced a differentiable spatial transformer network (STN), which can determine the areas that need to be paid attention in the feature map through transformations such as cropping, translation, rotation, scale, and skew. Unlike pooling layers, the spatial Transformer module is a dynamic mechanism that can actively spatially transform an image (or a feature map) by producing an appropriate transformation for each input sample.

Channel attention allows neural networks to learn what should be focused on. Hu et al. [54] proposed the squeeze-and-excitation (SE) network that adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. In their SE module, they used global average-pooled features to compute channel-wise attention. Li et al. [77] proposed SKNet that improves the efficiency and effectiveness of object recognition by adaptive kernel selection in a channel attention manner. Besides, Stollenga et al. [78] proposed a channel hard attention mechanism that improves classification performance by allowing the network to iteratively focus on the attention of its filters.

Capturing long-range dependencies is of central importance in deep neural networks, which is beneficial to visually understanding problems. Wang et al. [70] applied the self-attention mechanism to the computer vision task to solve this problem, called *non-local attention*. They proposed the non-local module, which obtains attention masks by calculating the correlation matrix between each spatial point in the feature map, and then the attention guides dense contextual information aggregation. However, this method also has the following problems: (1) Only the positional attention module is involved, not the commonly used channel attention mechanism. (2) When the input feature map is very large, there is a problem of low efficiency. Although there are other methods to solve this problem, such as scaling, it will lose information and is not the best way to deal with it. To address these problems, the researchers improved the non-local method and combined the channel attention to propose *mixed attention* [79–85].

CCNet [86] used *positional-wise attention*, different from previous studies, to generate a huge attention map to record the relationship between each pixel-pair in the feature map. The crisscross attention module collected contextual information in horizontal and vertical directions to enhance

pixel-wise representative capability. And the recurrent crisscross attention allows to capture dense long-range contextual information from all pixels with less computing and memory costs.

1.5.2 Nature Language Processing

In this part, we first introduce some attention methods used in different tasks of natural language processing, and then introduce some general pre-training word representations suitable for natural language processing tasks.

Machine Translation uses neural networks to translate text or speech from one language to another. Bahdanau et al. [8] introduced the attention mechanism into the neural network to improve neural machine translation by selectively focusing on parts of the source sentence during translation. Afterward, several works have been improved on this basis. Luong et al. [13] proposed the local/global approach. Lee et al. [87] presented interactive attention, which can keep track of the interaction history. Vaswani et al. [88] proposed supervised attention learned with guidance from conventional alignment models. Liu et al. [15] presented the Transformer model composed entirely of a self-attention mechanism.

Text Classification aims at assigning labels to text and has broad applications, including topic labeling [89], sentiment classification [90, 91], and spam detection [92]. Zhang et al. [93] proposed a new CNN model that jointly exploits labels on documents and their constituent sentences. Letarte et al. [30] proposed the self-attention network (SANet) that models the interactions between all input word pairs. Yang et al. [46] proposed a hierarchical-input attention network with two levels of attention mechanisms applied at the word and sentence level.

Text Matching is also a core research problem in NLP and Information Retrieval, which includes question answering, document search, entailment classification, paraphrase identification, and recommendation with reviews. Many researchers have come up with new approaches in conjunction with attention comprehensions, such as memory networks [38], attention-over-attention [94], inner attention [66], structured attention [48], and co-attention [42, 61].

Pre-trained word representations are a key component in many neural language understanding models. However, previous studies [95, 96] only determined one embedding for the same word, which could not achieve contextual word embedding. Peters et al. [97] introduced a general approach of context-dependent representation with Bi-LSTM to solve this problem. Inspired by the Transformer model [45], the researchers proposed the BERT [98] and GPT [99–101] methods according to the encoder and decoder parts. BERT is a bidirectional language model with two pre-training tasks: (1) masked language model (MLM). Simply masking some percentage of the input tokens randomly, then predicting those masked tokens; and (2) next sentence prediction. They use a linear binary classifier to determine whether two sentences are connected. GPT is a one-way model, and its training method roughly uses the previous word to predict the next word. Experiments show large improvements when applying them to a broad range of NLP tasks.

1.6 Attention for Interpretability

Interpretability is an particular concern for many current deep learning models. As they become increasingly complex and learn decision-making functions from data, ensuring our ability to

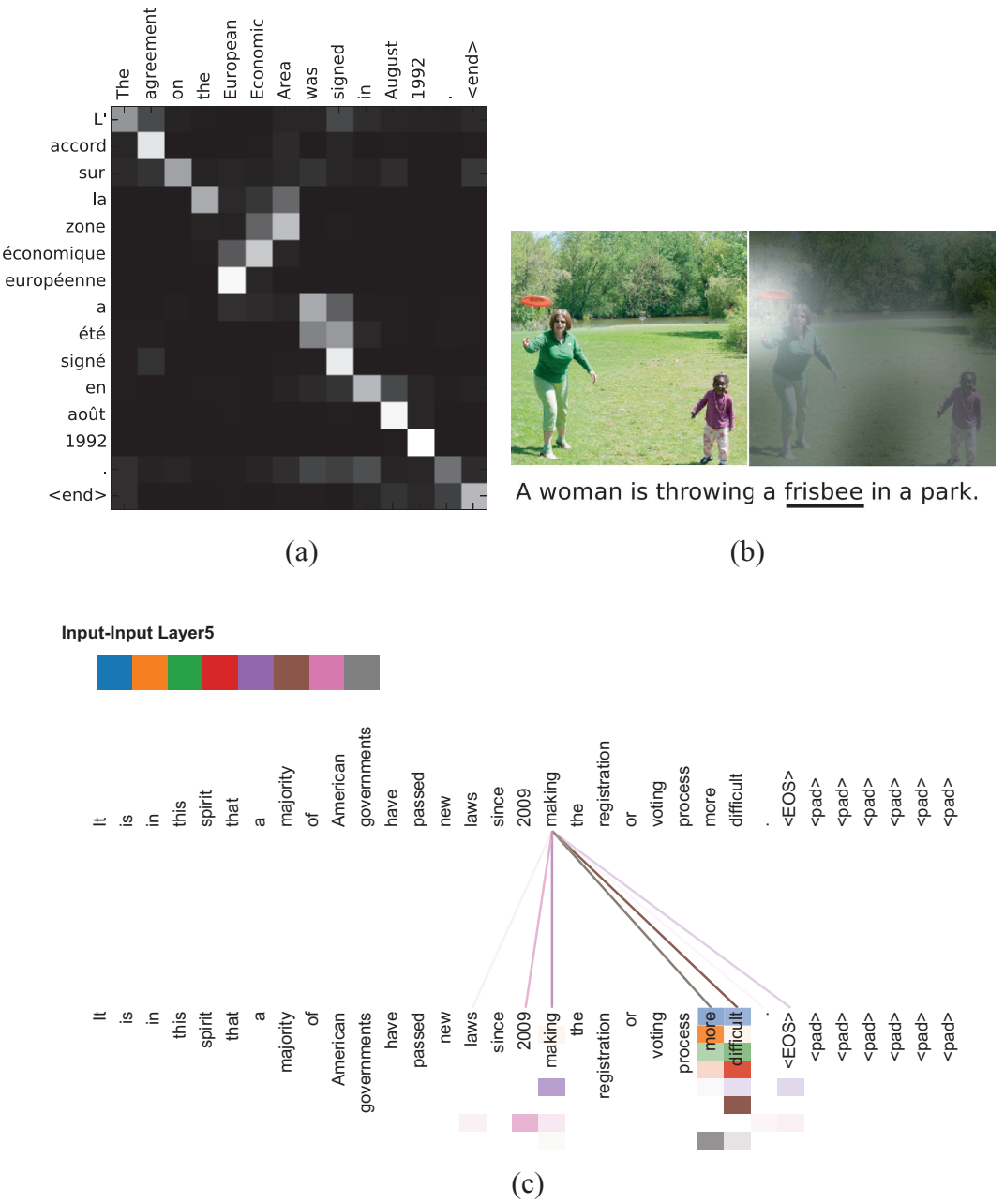


Figure 1.4 Examples of visualizing the weight of the attention mechanism. (a) Alignment of French and English sentences in MT [8] (b) Relevant image regions for image captioning [13] (c) An example of visualizing the weight of the word "making" in the Transformer model [15].

understand why a particular decision occurred is critical [27]. The attention layer in a neural network model provides a way to reason about the model behind its predictions [30–33, 102], but this is often criticized as opaque [28, 29].

As shown in Figure 1.4(a), Bahdanau et al. [8] visualized attention weights annotations that clearly show the (soft-)alignment between words in a generated translation (French) and those

in a source sentence (English). Figure 1.4(b) shows the attended regions corresponding to the underlined word in the neural image caption generation process [10]. As we can see, the model learns alignments that correspond very strongly with human intuition. Finally, Figure 1.4(c) show an example of visualization in the encoder self-attention layer in the Transformer model proposed by Vaswani et al. [15]. Different colors represent different heads.

Chan et al. [20] observed that the content-based attention mechanism can identify the start position in the audio sequence for the first character correctly. Voita et al. [103] evaluated the contribution made by individual attention heads to Transformer model performance on translation. They discovered that different heads have different importance through pruning operations. Besides, Lee et al. [87] and Liu et al. [104] provided some tools for visualizing attention weights of deep neural networks. Furthermore, Sen et al. [105] proposed novel attention-map similarity metrics to quantify similarities between human and attention-based neural network models.

However, some recent studies [28, 29] suggested that attention cannot be considered a reliable means to interpret deep neural networks. Jain and Wallace [28] performed extensive experiments across a various NLP tasks and proved that attention is inconsistent with other explainability metrics. They found that it is often possible to construct adversarial attention distributions, which means that different attention weight distributions yield equivalent predictions. Serrano and Smith [29] applied a different analysis based on intermediate representation and found that attention weights are only noisy predictors of even intermediate importance of components. On the contrary, Wiegreffe and Pinter [33] have made a rebuttal to their work with four alternative tests. Whether or not attention is used to explain neural networks is currently an open debate.

1.7 Conclusion and Prospects

Attention models have become ubiquitous in deep neural networks. In this chapter, we summarize the network architectures and various variants of the attention mechanism, define a unified attention model and propose our taxonomy. Since Bahdanau et al. [8] used the attention mechanism for alignment in machine translation task, various variants have emerged. The Transformer model, which only uses self-attention mechanism proposed by Vaswani et al. [15], is an important milestone in the attention mechanism, and its variants [98, 100, 106–108] have been successfully applied in various fields. Recent studies [109, 110] have shown that there is much room for improvement in the design of the attention mechanisms. We hope that survey will provide a better understanding of the various ways to improve attention mechanisms.

References

[1] Maurizio Corbetta, and Gordon L. Shulman. Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews Neuroscience*, 3(3):201–215, 2002.

[2] Ronald A Rensink. The dynamic representation of scenes. *Visual Cognition*, 7(1–3):17–42, 2000.

[3] John K. Tsotsos, Scan M. Culhane, Winky Yan Kei Wai, Yuzhong Lai, Neal Davis, and Fernando Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78(1–2):507–545, 1995.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734. ACL, 2014.

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

[6] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[7] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[8] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. CoRR abs/1409.0473, 2014.

[9] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, pages 3242–3250. IEEE Computer Society, 2017.

[10] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org, 2015.

[11] Gang Liu, and Jiabao Guo. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

[12] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[13] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421. The Association for Computational Linguistics, 2015.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[16] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*, pages 4263–4270. AAAI Press, 2017.

[17] Yan Tian, Wei Hu, Hangsen Jiang, and Jiachen Wu. Densely connected attentional pyramid residual network for human pose estimation. *Neurocomputing*, 347:13–23, 2019.

[18] Pengfei Zhang, Jianru Xue, Cuiling Lan, Wenjun Zeng, Zhanning Gao, and Nanning Zheng. Adding attentiveness to the neurons in recurrent neural networks. In *ECCV (9)*, volume 11213 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2018.

[19] Kaikai Song, Ting Yao, Qiang Ling, and Tao Mei. Boosting image sentiment analysis with visual attention. *Neurocomputing*, 312:218–228, 2018.

[20] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, pages 4960–4964. IEEE, 2016.

[21] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *CoRR*, abs/1412.1602, 2014.

[22] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. Self-attentional acoustic models. In *Interspeech*, pages 3723–3727. ISCA, 2018.

[23] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *AAAI*, pages 2532–2539. AAAI Press, 2018.

[24] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention networks. In *IJCAI*, pages 3926–3932. ijcai.org, 2018.

[25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio' and Yoshua Bengio. Graph Attention Networks. ArXiv abs/1710.10903, 2017.

[26] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *CoRR*, abs/1804.00823, 2018.
[27] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):93:1–93:42, 2019.
[28] Sarthak Jain, and Byron C. Wallace. Attention is not explanation. In *NAACL-HLT (1)*, pages 3543–3556. Association for Computational Linguistics, 2019.
[29] Sofia Serrano and Noah A. Smith. Is Attention Interpretable? ArXiv abs/1906.03731, 2019.
[30] Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. Importance of self-attention for sentiment analysis. In *BlackboxNLP@EMNLP*, pages 267–275. Association for Computational Linguistics, 2018.
[31] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. What does BERT with vision look at? In *ACL*, pages 5265–5275. Association for Computational Linguistics, 2020.
[32] Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. Attention interpretability across NLP tasks. *CoRR*, abs/1909.11218, 2019.
[33] Sarah Wiegreffe, and Yuval Pinter. Attention is not explanation. In *EMNLP/IJCNLP (1)*, pages 11–20. Association for Computational Linguistics, 2019.
[34] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
[35] Shaonan Wang, Jiajun Zhang, and Chengqing Zong. Learning sentence representation with guidance of human attention. In *IJCAI*, pages 4137–4143. ijcai.org, 2017.
[36] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turning machines. *CoRR*, abs/1410.5401, 2014.
[37] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1378–1387. JMLR.org, 2016.
[38] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.
[39] J. Weston, S. Chopra, A. Bordes, Memory networks, CoRR abs/1410.3916, 2014.
[40] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 2017.
[41] Mike Schuster, and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
[42] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297, 2016.
[43] Alessandro Sordoni, Philip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 2016.
[44] Shenjian Zhao, and Zhihua Zhang. Attention-via-attention neural machine translation. In *AAAI*, pages 563–570. AAAI Press, 2018.
[45] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention, please! A critical review of neural attention models in natural language processing. *CoRR*, abs/1902.02181, 2019.
[46] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489. The Association for Computational Linguistics, 2016.
[47] André F. T. Martins, and Ramón Fernández Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1614–1623. JMLR.org, 2016.
[48] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. *arXiv: Computation and Language*, 2017.
[49] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *NIPS*, pages 4331–4339, 2016.

[50] Çaglar Gülçehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic neural turing machine with soft and hard addressing schemes. *CoRR*, abs/1607.00036, 2016.

[51] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, pages 1400–1409. The Association for Computational Linguistics, 2016.

[52] Scott E. Reed, and Nando de Freitas. Neural programmer-interpreters. In *ICLR*, 2016.

[53] Michal Daniluk, Tim Rocktäschel, Johannes Welbl and Sebastian Riedel. Frustratingly Short Attention Spans in Neural Language Modeling. ArXiv abs/1702.04521, 2017.

[54] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023, 2020.

[55] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455. AAAI Press, 2018.

[56] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[57] K. M. Hermann, T. Kocisky´, E. Grefenstette, L. Espeholt,W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, CoRR, abs/1506.03340 (2015). arXiv:1506.03340.

[58] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. An attentive survey of attention models. *CoRR*, abs/1904.02874, 2019.

[59] Feifan Fan, Yansong Feng, and Dongyan Zhao. Multi-grained attention network for aspect-level sentiment classification. In *EMNLP*, pages 3433–3442. Association for Computational Linguistics, 2018.

[60] Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *AAAI*, pages 3316–3322. AAAI Press, 2017.

[61] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*, pages 4425–4431. ijcai.org, 2018.

[62] Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. Adaptive co-attention network for named entity recognition in tweets. In *AAAI*, pages 5674–5681. AAAI Press, 2018.

[63] Feng Nie, Yunbo Cao, Jinpeng Wang, Chin-Yew Lin, and Rong Pan. Mention and entity description co-attention for entity disambiguation. In *AAAI*, pages 5908–5915. AAAI Press, 2018.

[64] Xiangju Li, Kaisong Song, Shi Feng, Daling Wang, and Yifei Zhang. A co-attention neural network model for emotion cause analysis with emotional context awareness. In *EMNLP*, pages 4752–4757. Association for Computational Linguistics, 2018.

[65] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. Attentive gated lexicon reader with contrastive contextual co-attention for sentiment classification. In *EMNLP*, pages 3443–3453. Association for Computational Linguistics, 2018.

[66] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *ACL (1)*. The Association for Computer Linguistics, 2016.

[67] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR (Poster)*. OpenReview.net, 2017.

[68] Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*, pages 5852–5859. AAAI Press, 2018.

[69] John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. Deeper attention to abusive user content moderation. In *EMNLP*, pages 1125–1135. Association for Computational Linguistics, 2017.

[70] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803. IEEE Computer Society, 2018.

[71] Lijun Wu, Fei Tian, Li Zhao, Jianhuang Lai, and Tie-Yan Liu. Word attention for sequence to sequence text understanding. In *AAAI*, pages 5578–5585. AAAI Press, 2018.

[72] Chuhan Wu, Fangzhao Wu, Junxin Liu, and Yongfeng Huang. Hierarchical user and item representation with three-tier attention for recommendation. In *NAACL-HLT (1)*, pages 1818–1826. Association for Computational Linguistics, 2019.

[73] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, pages 842–850. IEEE Computer Society, 2015.

[74] Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. Multi-head attention with disagreement regularization. In *EMNLP*, pages 2897–2903. Association for Computational Linguistics, 2018.

[75] Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. Multi-level structured self-attentions for distantly supervised relation extraction. In *EMNLP*, pages 2216–2225. Association for Computational Linguistics, 2018.

[76] Francesco Barbieri, Luis Espinosa Anke, José Camacho-Collados, Steven Schockaert, and Horacio Saggion. Interpretable emoji prediction via label-wise attention lstms. In *EMNLP*, pages 4766–4771. Association for Computational Linguistics, 2018.

[77] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*, pages 510–519. Computer Vision Foundation/IEEE, 2019.

[78] Marijn F. Stollenga, Jonathan Masci, Faustino J. Gomez, and Jürgen Schmidhuber. Deep networks with internal selective attention through feedback connections. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13, 2014, Montreal, Quebec, Canada*, pages 3545–3553, 2014.

[79] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27–28, 2019*, pages 1971–1980. IEEE, 2019.

[80] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*, pages 3146–3154. Computer Vision Foundation/IEEE, 2019.

[81] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*, pages 6450–6458. IEEE Computer Society, 2017.

[82] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018.

[83] Yuhui Yuan, and Jingdong Wang. Ocnet: Object context network for scene parsing. *CoRR*, abs/1809.00916, 2018.

[84] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada*, pages 6511–6520, 2018.

[85] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part IX*, volume 11213 of *Lecture Notes in Computer Science*, pages 270–286. Springer, 2018.

[86] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*, pages 603–612. IEEE, 2019.

[87] Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. Interactive visualization and manipulation of attention-based neural machine translation. In Lucia Specia, Matt Post, and Michael Paul, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, 2017 - System Demonstrations*, pages 121–126. Association for Computational Linguistics, 2017.

[88] Lemao Liu, Masao Utiyama, Andrew M. Finch, and Eiichiro Sumita. Neural machine translation with supervised attention. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11–16, 2016, Osaka, Japan*, pages 3093–3102. ACL, 2016.

[89] Sida I. Wang, and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8–14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 90–94. The Association for Computer Linguistics, 2012.

[90] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19–24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics, 2011.

[91] Bo Pang, and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2007.

[92] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, volume 62, pages 98–105. Madison, Wisconsin, 1998.

[93] Ye Zhang, Iain James Marshall, and Byron C. Wallace. Rationale-augmented convolutional neural networks for text classification. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1–4, 2016*, pages 795–804. The Association for Computational Linguistics, 2016.

[94] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In Regina Barzilay, and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30–August 4, Volume 1: Long Papers*, pages 593–602. Association for Computational Linguistics, 2017.

[95] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings*, 2013.

[96] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014.

[97] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018.

[98] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[99] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[100] A. Radford, K. Narasimhan, Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[101] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[102] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016.

[103] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David R. Traum, and Llus Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28–August 2, 2019, Volume 1: Long Papers*, pages 5797–5808. Association for Computational Linguistics, 2019.

[104] Shusen Liu, Tao Li, Zhimin Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. Visual interrogation of attention-based models for natural language inference and machine comprehension. In Eduardo Blanco, and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31–November 4, 2018*, pages 36–41. Association for Computational Linguistics, 2018.

[105] Cansu Sen, Thomas Hartvigsen, Biao Yin, Xiangnan Kong, and Elke A. Rundensteiner. Human attention maps for text classification: Do humans and neural networks focus on the same words? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020*, pages 4596–4608. Association for Computational Linguistics, 2020.

[106] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28–August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics, 2019.

[107] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*. OpenReview.net, 2019.

[108] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-Transformer. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, pages 1315–1325. Association for Computational Linguistics, 2019.

[109] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in Transformer models. *CoRR*, abs/2005.00743, 2020.

[110] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*, pages 6687–6696. IEEE, 2019.

Chapter 2

Recurrent Attention Unit

Guoqiang Zhong, Guohua Yue, Zhaoyang Niu, and Xiao Ling
College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract

Recurrent neural network (RNN) has been successfully applied in many sequence learning problems, such as handwriting recognition, image caption, natural language processing, and video motion analysis. After years of development, researchers have greatly improved the internal structure of RNN and introduced many variants of it. Among others, gated recurrent unit (GRU) is one of the most widely used RNN models. However, GRU lacks the capability of adaptively paying attention to certain regions or locations, so it may cause information redundancy or loss during learning. In this chapter, we propose a novel RNN model called recurrent attention unit (RAU), which seamlessly integrates the attention mechanism into the interior of GRU by adding an attention gate.

Key Words: Deep learning, Attention mechanism, Gated recurrent unit

2.1 Introduction

Recurrent neural network (RNN) is a widely used method in the field of deep learning. It has been proven to be very successful in both classification and generation tasks (see, e.g. [1–3]). Unlike other artificial neural networks, such as deep belief nets [4] and convolutional neural networks [5], RNN is commonly used to deal with sequence modeling problems. One of the core ideas of RNN is to connect previous neurons with the current processing unit, to have a memory function for the previous information. Theoretically, RNN can capture any long-term dependency information. However, it is difficult to do so in practical applications due to the gradient exploding and vanishing problems [6–8]. One of the most promising ways to solve this problem is to change the architecture of RNN, e.g., by using a gated activation function to achieve the trade-off between the old time and new time information. Both long short-term memory (LSTM) [9] and gated recurrent unit (GRU) [10] are RNN extensions. They use the gated activation functions and, to some extent, alleviate the gradient vanishing and exploding problems.

However, compared with LSTM, GRU reduces the number of gate control units from three to two, and the model is simpler and has higher efficiency. Józefowicz *et al.* [11] compared GRU with LSTM, finding that GRU can achieve results equivalent to LSTM on many issues and is easier to train. Therefore, GRUs are increasingly used in natural language processing tasks.

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

For example, Shang *et al.* [12] used GRUs in neural response machines, and Kiros *et al.* [13] replaced the traditional LSTM with GRU in the implementation of a language model. In this work, we focus on GRUs because they have been shown to be easier to train than LSTM and have good performance.

Attention mechanism is a resource allocation mechanism. For example, in human visual processing, even if human eyes have the ability to accept a large visual field, one's eyes are usually fixed on a certain part. That is, with limited visual resources, one needs to first select a specific part of the visual field and then process it. For example, when one is reading, the word to be read is usually focused on a specific time. Just like people's perception mechanisms, computer vision should also pay attention to a specific part of the input rather than assigning all inputs the same weight. Therefore, the attention mechanism is a simulation of human attention allocation mechanism and is used to allocate limited resources to important parts. The nature of attention has been studied extensively in the previous literatures [14–18]. The attention mechanism is originally applied to image classification tasks [16] in computer vision and is subsequently applied to image caption problems [19]. In natural language processing, the attention mechanism is usually used by combining with the encoder–decoder model [2, 20], and the application is very extensive.

We have found that although GRU solves the problem of memorizing longer-term information, it does not highlight the important content of the information. Fortunately, the learning model based on the attention mechanism can pay close attention to the part that needs attention and suppresses unimportant information. Motivated by it, we propose the recurrent attention unit (RAU), a novel RNN architecture that combines the strengths of both GRU and the attention mechanism. In RAU, we apply the attention model to the interior of GRU by seamlessly adding an attention gate. The attention gate enhances RAU's ability to remember long-term information and helps memory cells quickly discard unimportant content.

The contributions of this chapter can be summarized as follows:

1. We propose a novel RNN architecture called RAU. Usually, attention mechanisms are implemented as an additional layer connected to the original RNN. The attention module may require additional training. In contrast, RAU seamlessly adds the attention gate in the memory cell of GRU, which makes the model simpler and easier to train.
2. Compared with LSTM and GRU, the attention gate can help RAU better remember the more important long-term information and discard the less important information in the sequential data. Extensive experiments demonstrate that RAU outperforms both LSTM and GRU models.
3. As a substitute for LSTM and GRU, RAU can be applied to many sequential data learning problems. Experiments show that RAU is suitable for image classification, language modeling, and sentiment classification tasks.

2.2 Related Work

RNN and its variants have been successfully applied to a variety of tasks, especially with time-dependent data. Handwriting recognition [21], image caption generation [19, 22], natural language processing generation [2, 3, 23], video caption generation [24], and so on, all need a model that can

learn sequence data. Therefore, research on RNN has always played a very important role in the field of deep learning.

2.2.1 Recurrent Neural Networks

Theoretically, an RNN can capture any long-term dependence of the input sequence. But in practice, an RNN must maintain an activation vector for each time step, which makes RNN very deep. This depth, in turn, makes the RNN suffer from gradient vanishing and exploding problems [7], resulting in the difficulty of training an RNN. There have been a number of attempts to address the problem of training RNNs. LSTM is an improved RNN model proposed in [9]. It can effectively alleviate the long-distance dependence problem of RNN by constructing special memory units to store historical information so that each time state preserves the previous input information. Later, Cho *et al.* [10] proposed a simpler and more convenient GRU architecture, which not only has the advantage of LSTM but also is easier to train and performs better.

Except for LSTM and GRU, there are many other variations in RNN research. In 2015, Le *et al.* [25] proposed a new structure IRNN, which can significantly improve RNN's performance by making only minor changes to RNN. The key to the practice is to initialize the weight of RNN to a unit matrix and use rectified linear unit (ReLU) as a non-linear activation function. Józefowicz *et al.* [11] proposed three different variants of GRU: using tanh in the gate to increase nonlinearity, removing the dependence on the hidden state in the gate, and using the above changes in the calculation of the hidden state. Li *et al.* [26] proposed an independent recurrent neural network (IndRNN). It replaces the matmul product with the Hadamard product in the recurrent input and uses the unsaturated function, such as ReLU, to handle longer sequences. Bradbury *et al.* [27] proposed a new network architecture called QRNN that replaces the traditional recurrent structure (vanilla RNN, LSTM, GRU) with a convolution operation. QRNN can be thought of as a special structure between RNN and convolutional neural network (CNN). Since the convolution operation has no time dependence on the recurrent structure, the calculation parallelism of QRNN is high. There are also many studies that use the hidden layer unit of LSTM to construct a multi-dimensional [28] or grid-structured [29] LSTM. Most of RNN's variants focus on improving its structure to achieve better results, such as changing its activation function, adding convolution operations, and adding connections. These variants yield better results than GRU in some scenarios, but none of them can always outperform GRU.

In this work, we improve the working ability of RNNs from a new perspective. Based on the advantage of the attention mechanism to adaptively extract important information, we design an efficient and simple attention gate that enables the network to quickly discard unwanted information and emphasize important information.

2.2.2 Attention Mechanisms

The attention mechanism is similar to the human visual attention mechanism. The core goal is to select information that is more critical to the current mission. The attention mechanism was originally used for computer vision [30]. With the development of deep neural networks, attention mechanism has become an important concept in the field of neural networks. Recently,

variants of the attention model have been used to address unique features in different application areas, such as image caption [19, 31], text classification [32], machine translation [2, 3, 33–35], action recognition [36–38], and sentiment analysis [39].

Wang *et al.* [19] introduced soft attention and hard attention to the image caption field and used visualization to understand the effects of the attention mechanism. The soft attention mechanism assigns an attention weight of 0–1 to each pixel of the image. Soft attention is a completely differentiable deterministic mechanism that can propagate through the attention mechanism while propagating to other parts of the network. In contrast, hard attention is a random process, which randomly selects an area of the image as the attention area, the attention weight of this part is 1, and the attention weight of other areas is 0. In order to achieve backpropagation of the gradient, Monte Carlo sampling is needed to estimate the gradient of the module. Luong *et al.* [20] proposed the global attention and local attention mechanisms for machine translation. In the encoder–decoder model of machine translation, the global attention mechanism considers all hidden states of the encoder when calculating semantic vectors. In local attention, only a part of the encoder hidden state of each target word is concerned. Sutskever *et al.* [35] proposed the self-attention mechanism for machine translation. For a sentence, each word in the sentence is evaluated with attention to all words in the sentence. The purpose is to learn the word dependence within the sentence and capture the internal structure of the sentence. Song *et al.* [37] proposed a combination of temporal attention and spatial attention for human action recognition. Temporal attention is used to assign appropriate importance to different frames. Spatial attention is used to assign appropriate importance to different joint points.

The attention model mentioned previously focuses on designing appropriate weights for specific task information (such as pixels or blocks of the input image, words of the input sentence, frames of input video, or bone points) as input to the RNN. However, they have not paid attention to the importance of information within the RNN. In other words, their attention mechanism emphasizes the network level rather than the RNN block level. Inspired by this, we are committed to exploring the RAU with an attention mechanism. The attention mechanism can suppress the forward movement of invalid information. In fact, the attention mechanism also facilitates the backpropagation of gradient information. This is particularly effective for preventing the disappearance of gradients during the long-term memory of information in the RNN.

2.3 Recurrent Attention Unit

It has been proved that GRU can maintain a long period of input information [40]. However, the existing GRU architecture is difficult to adaptively pay attention to information exactly useful at the hidden layer state and each time step, especially when data have complex potential connections, which are common in language modeling and sentiment classification tasks. To overcome this problem, we propose a novel recurrent neural network architecture called RAU. RAU can not only quatify information to be preserved over time but also suppress redundant information. We integrate the attention mechanism seamlessly into the interior of GRU by adding an attention gate. The attention gate can enhance GRU's ability to remember long-term memory and help memory cells quickly discard unimportant content.

In the following, we introduce the basic structure of RAU based on GRU and explain why RAU can adaptively pay attention to key information.

2.3.1 Gated Recurrent Unit

LSTM is an improved RNN that can learn long-term dependency information. GRU is a variant of LSTM that makes the model simpler and has similar characteristics to LSTM.

LSTM is a complex network structure that consists of three gates: the forgotten gate, the input gate, and the output gate. These three gates are used to increase or decrease the information in the cell state. The GRU combines the forgotten gate and the input gate in the LSTM into a single update gate while also removing the cell state. GRU also has a reset gate. Its structure is shown in Figure 2.1.

In Figure 2.1, x_t is the input data at time t and h_t is the output at time t of the GRU. In the GRU structure, the update gate z_t controls how much of the past information is brought into the current state. The update gate is computed based on the previous hidden states h_{t-1} and the current input x_t as follows :

$$z_t = \sigma\left(W_z[x_t, h_{t-1}] + b_z\right), \tag{2.1}$$

where σ represents the sigmoid function, W_z represents the update gate weight, and b_z indicates the update gate bias.

In addition, r_t is a reset gate. It outputs a value ranging from 0 to 1 based on the previous hidden states h_{t-1} and the current input x_t. r_t decides whether the past information is retained or discarded. Value 0 means completely discarding, allowing it to forget the previously computed state, and 1 means full reservation.

$$r_t = \sigma\left(W_r[x_t, h_{t-1}] + b_r\right), \tag{2.2}$$

where σ represents the sigmoid function, W_r represents the reset gate weight, and b_r indicates the reset gate bias.

Then, a new candidate value vector is created and added to the state. The formula for this process is as follows:

$$\tilde{h}_t = tanh\left(W_{\tilde{h}}[x_t, r_t h_{t-1}] + b_{\tilde{h}}\right), \tag{2.3}$$

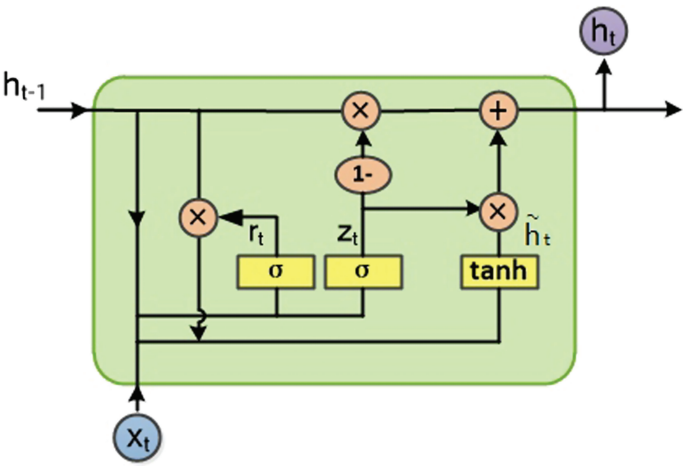


Figure 2.1 Gated recurrent unit.

where tanh is the hyperbolic tangent function, $\mathrm{W}_{\tilde{h}}$ represents the update candidate value, $\mathrm{b}_{\tilde{h}}$ refers to the update candidate bias, and $\tilde{\mathrm{h}}_t$ refers to the candidate value.

Finally, determining the value of the output, this process retains the information of the current unit and passes it to the next unit. We have

$$\mathrm{h}_t = (1 - z_t)\mathrm{h}_{t-1} + z_t\tilde{\mathrm{h}}_t. \tag{2.4}$$

2.3.2 The Recurrent Attention Unit Model

For an RAU block, we propose an attention gate to enable the RNN neurons to have the attentive capability. The response of an attention gate is a vector a_t for each input element as follows:

$$\mathbf{c}_t = \mathrm{W}_{hc}\mathrm{h}_{t-1} + \mathrm{W}_{xc}\mathrm{x}_t, \tag{2.5}$$

$$a_t^i = \frac{\exp(c_t^i)}{\sum_{i=1}^{k}\exp(c_t^i)}, \tag{2.6}$$

$$\hat{\mathrm{h}}_t = \mathrm{a}_t \odot ReLU(\mathrm{W}_{hu}\mathrm{h}_{t-1} + \mathrm{W}_{xu}\mathrm{x}_t), \tag{2.7}$$

where $\mathrm{W}_{hc}, \mathrm{W}_{hu} \in \mathbb{R}^{K\times N}$ and $\mathrm{W}_{xc}, \mathrm{W}_{xu} \in \mathbb{R}^{K\times M}$ are the learned parameters, x_t is the external m-dimensional input vector at time t, and h_{t-1} is the n-dimensional hidden state at the last moment. The attention weight a_t and input c_t are vectors. ReLU is an unsaturated activation function.

Due to the special structure of the RNN, when it is used to process data, the output h_t depends on the input x_t and the hidden information h_{t-1}. In theory, the hidden state h_t encodes the semantics of the input sequence from time 1 to time t. However, for a long-sequence problem, encoding all semantic information into a fixed-length vector will cause information loss to some extent. To solve this problem, we use attention weight to enhance the memory of key semantics and to discard useless information in time.

Figure 2.2 is a complete RAU architecture. There is an attention gate, which generates a weight vector $\mathbf{a}_t$ based on the previous hidden states h_{t-1} and the current input x_t. We use layer

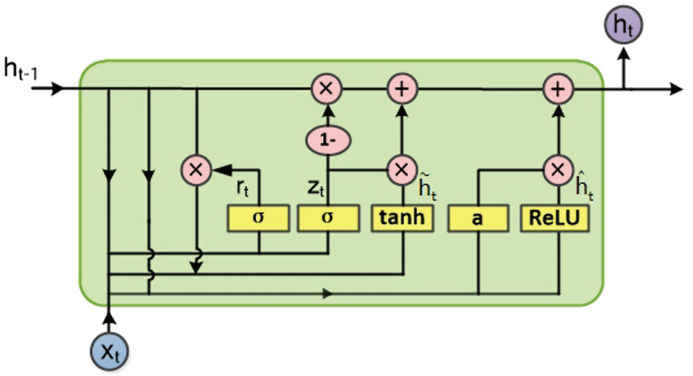


Figure 2.2 Recurrent attention unit

normalization [41] on the attention gate. $\hat{h}_t$ in Equation (2.7) is the hidden layer information after the attention is applied.

In Figure 2.2, x_t is the input data at time t, and h_t is the output at time t of RAU. In the RAU gate structure, the update gate z_t controls how much of the past information is brought into the current state. It is computed based on the previous hidden states h_{t-1} and the current input x_t as follows:

$$z_t = \sigma\left(W_z[x_t, h_{t-1}] + b_z\right), \tag{2.8}$$

where σ represents the sigmoid function, W_z represents the update gate weight, and b_z indicates the update gate bias.

In addition, $\mathbf{r}_t$ is a reset gate. It outputs a value ranging from 0 to 1 based on the previous hidden states h_{t-1} and the current input x_t. $\mathbf{r}_t$ decides whether past information is retained or discarded. Value 0 means completely discarding, allowing it to forget the previous state, and 1 means full reservation.

$$\mathbf{r}_t = \sigma\left(W_r[x_t, h_{t-1}] + b_r\right), \tag{2.9}$$

where σ represents the sigmoid function, W_r represents the reset gate weight, and b_r indicates the reset gate bias.

Then, a new candidate vector is created and added to the state. The formula for this process is as follows:

$$\tilde{h}_t = tanh\left(W_{\tilde{h}}[x_t, r_t h_{t-1}] + b_{\tilde{h}}\right), \tag{2.10}$$

where tanh is the hyperbolic tangent function, $W_{\tilde{h}}$ represents the update candidate value, $b_{\tilde{h}}$ refers to the update candidate bias, and $\tilde{h}_t$ refers to the candidate value.

Finally, for an RAU block with an attention gate, the output responses h_t at time t are calculated as

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t + \hat{h}_t. \tag{2.11}$$

The advantage of adding an attention gate is to pay more attention to important information. When the update gate is opened, there are only two terms in h_t — $\tilde{h}_t$ and $\hat{h}_t$. At this time, our attention gate will help the cell to enhance the past useful information and to prevent the loss of important information. Therefore, by using this simplified attention mechanism, the model can help memory cells quickly discard unimportant content and increase attention to important information, thus enhancing the network's ability to remember long-term memory.

2.4 Experiments

We evaluate the performance of RAU on four different datasets: Modified National Institute of Standards and Technology (MNIST), Fashion-MNIST, Penn Treebank (PTB), and Internet Movie Database (IMDb). In other words, we verify the effectiveness of RAU in image classification, language modeling, and sentiment classification tasks. Furthermore, we take LSTM and GRU as the baseline methods.

2.4.1 Application to Row-wise Sequences of the MNIST Dataset

The MNIST dataset is one of the most famous datasets in machine learning, derived from the National Institute of Standards and Technology. It consists of 0–9 handwritten numeral labels, which are divided into a training set and a test set. Figure 2.3 shows a sample diagram of MNIST data for 10 × 10. The training set contains 60,000 handwritten digital images, while the test set contains 10,000 handwritten digital images; each of the images has been normalized to a digitally centric 28 × 28 gray-scale map. We evaluate RAU against the GRU and LSTM on the MNIST dataset by generating the sequential input row-wise (one row at a time). The row-wise sequence generated from each image is a 28-element signal of length 28.

In our experiments, the networks read one row at a time in scanline order (i.e., starting at the first row of the image and ending at the last row of the image). The networks are asked to predict the category of the MNIST image only after seeing all 28 rows. This is a medium long-range dependency problem because each recurrent network has 28 time steps. All networks have 128 recurrent hidden units. We stop the optimization after it converges or when it reaches 100,000 iterations and report the results in Table 2.2 (related characteristics are listed in Table 2.1). It is clear from Table 2.2 that the proposed RAU architecture outperforms the other baseline architectures when using the same hyperparameters.

We also compare our RAU with a recurrent model of visual attention (RAM) [42] on MNIST. RAM is a visual attention model that is formulated as a single RNN. The results are shown in Table 2.3. Our RAU outperforms RAM on this dataset. Our RAU achieves 98.80% test accuracy on MNIST, while RAM achieves the highest 98.71% test accuracy.

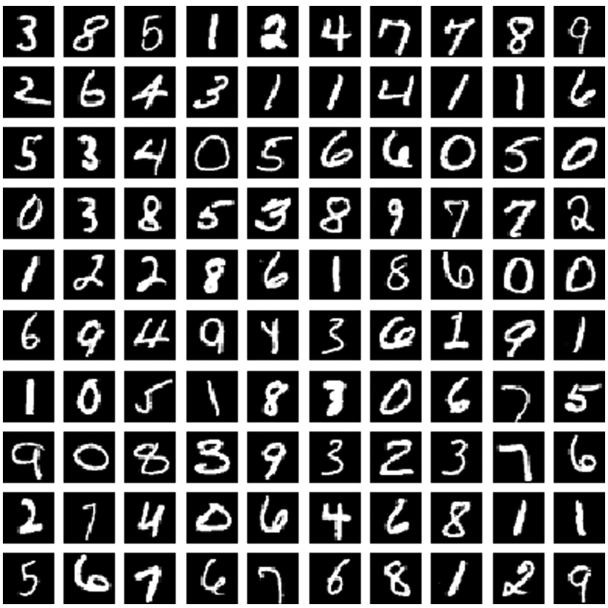


Figure 2.3 MNIST dataset examples.

Table 2.1 Network model characteristics

Model	**MNIST**	**Fashion-MNIST**	**IMDb**
Hidden Units	128	128	128
Gate Activation	Sigmoid	Sigmoid	Sigmoid
Activation	Softmax	Softmax	Softmax
Epochs	213	213	100
Optimizer	Adam	Adam	Adam
Dropout	—	—	50%
Batch Size	128	128	128

We have set the number of iterations to 100,000 steps in MNIST and Fashion-MNIST datasets, so we calculated the epoch number to be 213.

2.4.2 Application to Row-wise Sequences of the Fashion-MNIST Dataset

Fashion-MNIST is a dataset similar to MNIST, and all items are based on the classifications on the Zalando website. Each of Zalando's fashion items has a set of photographs taken by professional photographers, showing different aspects of the product, such as the appearance of the front and back, details. Fashion-MNIST is also stored in a 28 × 28 gray-scale map. It contains 70,000 pictures, 10 classes, and 7000 pictures per class. And it randomly selects 6000 pictures from each class to form training sets and the other 1000 to form test sets. Figure 2.4 shows some examples.

We also perform experiments on the Fashion-MNIST dataset. As shown in Table 2.1, we compare the three architectures with the same parameter configurations. We use the RAU, GRU, and LSTM network with a single layer of 128 hidden units to solve the classification task. Each time we read one row of an image, it is equivalent to an image being a sequence of 28 lengths. In Figure 2.5, we plot the learning curves of three models on Fashion-MNIST. We observe that RAUs tend to make faster progress over time. This behavior is observed both when the number of parameters is constrained and when the number of hidden units is constrained. In Table 2.2, we can see that our proposed RAU achieves 89.60% accuracy in the Fashion-MNIST dataset, which is 1.71% higher than GRU.

We also apply the model proposed in [42] to the Fashion-MNIST dataset, and the comparison results are shown in Table 2.4. As shown in the table, when the glimpse is set to four, the highest accuracy of 88.43% is obtained on the Fashion-MNIST dataset. But our RAU model outperforms four-glimpses RAM, which improves the accuracy to 89.60%.

2.4.3 Application to Language Modeling Task

In this chapter, we carry out a series of experiments on the Penn Treebank Corpus to verify that our proposed model can solve the language modeling task. The Penn Treebank Corpus [43] is an English tree library, established by the University of Pennsylvania, commonly used to conduct natural language processing tasks such as part-of-speech tagging and syntactic analysis. It is widely used in the language mode field.

A common indicator of the quality of a language model is perplexity. In simple terms, the perplexity value characterizes the probability of occurrence of an estimated sentence through

Table 2.2 Performance of three models on different tasks

	Problem	LSTM		GRU		RAU	
		Validation	Test	Validation	Test	Validation	Test
Row-wise Sequence	MNIST Acc. (%)	—	98.54	—	98.55	—	**98.80**
Row-wise Sequence	Fashion-MNIST Acc. (%)	—	87.89	—	88.45	—	**89.60**
Sentiment Classification	IMDB Acc. (%)	79.88	76.59	78.44	75.70	80.20	**76.92**
Language Modeling	PTB-Small (perplexity)	119.37	115.06	120.64	114.96	118.99	**113.89**
	PTB-Medium (perplexity)	86.50	83.01	87.80	83.62	86.75	**82.80**
	PTB-Large (perplexity)	83.10	78.50	82.62	78.63	82.63	**78.32**

Perplexity is a measure of the quality of the language model, the lower the better.

Table 2.3 Comparison between the recurrent model of visual attention (RAM) [42] and RAU on raw-wise MNIST data

Model	Accuracy (%)
FC, 2layers	98.65
1 Random Glimpse	57.15
RAM, 2glimpses	93.73
RAM, 3glimpses	97.3
RAM, 4glimpses	98.27
RAM, 5glimpses	98.45
RAM, 6glimpses	98.71
RAM, 7glimpses	98.53
RAU	**98.80**

For the comparison to LSTM and GRU, please refer to Table 2.2. FC denotes a fully connected network with two layers of rectifier units. Instances of the attention model are labeled with the number of glimpses.

Figure 2.4 Fashion-MNIST dataset example.

a certain language model. For example, when you already know (ω_1, ω_2, ω_3, $\cdots$, ω_m) that the sentence appears in the corpus, then the probability of this sentence would be calculated by the language model as higher as possible, while the perplexity value as small as possible. The formula for calculating the perplexity value is as follows:

$$\log(perplexity(S)) = \frac{-\sum P(\omega_i \mid \omega_1, \omega_2, \cdots, \omega_{i-1})}{m} \tag{2.12}$$

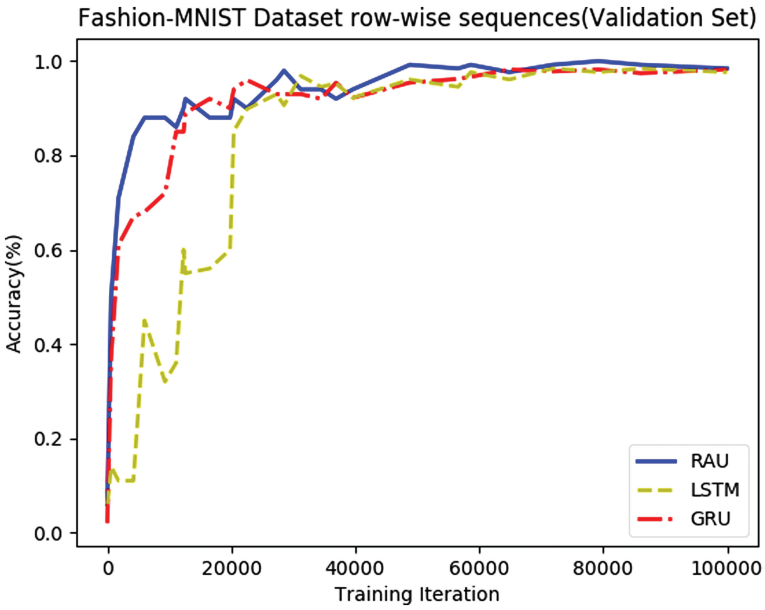


Figure 2.5 Validation learning curves of three different architectures: RAU, GRU, and LSTM with the same number of hidden units. The curves represent training up to 100,000 steps.

Table 2.4 The comparison between our model RAU and the RAM model in [42] on the Fashion-MNIST dataset

Model	Validation (%)	Test (%)
RAM, 2glimpses	82.71	82.56
RAM, 3glimpses	82.99	82.76
RAM, 4glimpses	88.01	88.43
RAM, 5glimpses	87.55	87.92
RAM, 6glimpses	87.81	87.86
RAM, 7glimpses	85.94	85.64
RAU	-	**89.60**

Instances of the attention model are labeled with the number of glimpses.

In fact, the concept represented by perplexity is the average branch factor, which is the average number of choices when the model predicts the next word. For example, if the perplexity of a language model is 83, it means that 83 words may be a reasonable choice for the next word.

We implement the three models with three different sizes of configurations in the Penn Treebank Corpus. We use the same experimental setup during the experiment set (the same training, development and test data, and the same vocabulary limit). This also helps to compare different language model techniques. We first performed the experiment of PTB using a small size: 2 layers with 200 units. The experimental results are shown in Table 2.2. In the last epoch, our proposed RAU model is reachable at 45.72 perplexity on the training set. The verification set and the test set can reach the perplexity of 118.99 and 113.89, respectively. Compared with the experiments of GRU and LSTM, it is clear that the model of RAU is more effective.

As shown in Table 2.5, in the PTB-medium experiment, we reduced the initial scale of the network weight value because the smaller initial value is beneficial to moderate training. The learning rate is unchanged, and the number of layers of the network stacked is unchanged. Here, the number of expansion steps of the gradient backpropagation is increased from 20 to 30, and the number of hidden layer units is also increased by three times. Because the number of iterations of learning increases, the decay rate of the learning rate will be smaller. As the Penn Treebank is a relatively small dataset, we prevent overfitting by using dropout technology.

In the case of a medium configuration, the loss of the three models is shown in Figure 2.6. It can be seen that the proposed model can significantly reduce the perplexity of the language model.

Table 2.5 Parameters of different sized models on PTB dataset

Model	**PTB-small**	**PTB-mediun**	**PTB-large**
Weight Initial Scale	0.1	0.05	0.04
Number Layer	2	2	2
Hidden Units	200	650	1500
Batch Size	20	20	20
Learning Rate	1	1	1
Learning Rate Decay	0.5	0.8	1/1.5
Epochs	13	35	55
Dropout	0	50%	65%
Vocabulary	10,000	10,000	10,000

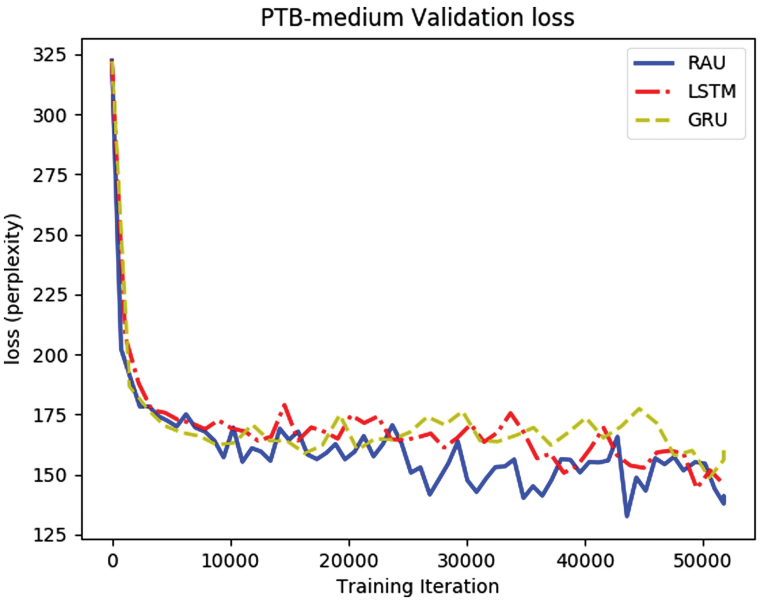


Figure 2.6 The validation loss curve of the three network models (RAU, GRU, and LSTM) in a medium configuration.

The large configuration model further reduces the initial scale of the weight values in the network while increasing the hidden layer size to 1500; because the complexity of the model increases, the dropout rate increases from 50% to 65%. Compared with LSTM and GRU in Table 2.2, we can see that the RAU is highly competitive. We observe that the performance of recurrent methods depends on the size of the hidden states: they perform better as the size of the hidden states gets larger.

2.4.4 Application to Sentiment Classification Task

We evaluate the RAU architecture on a popular document-level sentiment classification benchmark, the IMDb movie review dataset [44]. The dataset consists of a balanced sample of 25,000 positive and 25,000 negative reviews, which are divided into equal-size train and test sets, with an average document length of 231 words [45]. We have trained three different architectures using the three constant base learning rates of 10^{-3}, 10^{-4}, and 10^{-5} over 20 epochs.

In the training process, we employ a 100-dimensional architecture and adopt a batch size of 32. We have observed that, using the constant base learning rate of 10^{-5}, performance is good, whereas performance is uniformly progressing over profile-curves, as shown in Figure 2.7. Table 2.2 summarizes the results of accuracy performance, which shows comparable performance among RAU, GRU, and LSTM. Table 2.1 also lists the number of parameters in IMDb dataset.

2.5 Conclusion

Attention mechanism is an intuitive method that is widely used in the field of computer vision by giving different weights to different parts of the input. We have proposed an RAU that integrates the attention mechanism seamlessly within the GRU by adding an attention gate. It not

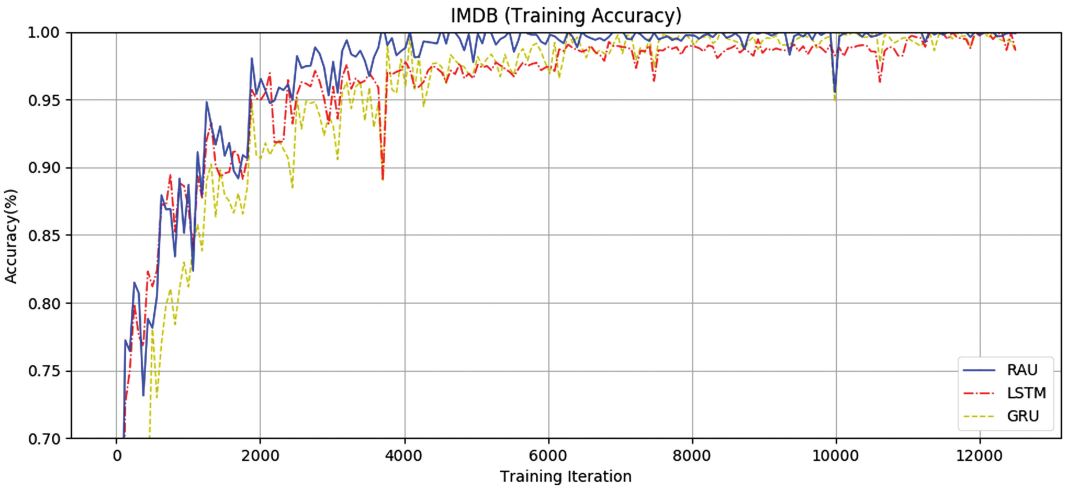


Figure 2.7 Train learning curves of three different architectures: RAU, GRU, and LSTM with the same learning rate (10^{-5}). The curves represent training up to 12,500 steps.

only amplifies the power of GRU but also enables the network to adaptively pay attention to the part that should be emphasized. We apply our model to three tasks: image classification, language modeling, and sentiment classification. The experimental results demonstrate that RAU outperforms both the LSTM and GRU. Therefore, RAU can be used as an alternative to LSTM and GRU in many application scenarios.

References

[1] Graves, A.: Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850 (2013)

[2] Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473 (2014)

[3] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. In: *NIPS*, pp. 3104–3112 (2014)

[4] Hinton, G.E., Osindero, S., Teh, Y.W.: A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 18(7), 1527–1554 (Jul 2006)

[5] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* 86(11), 2278–2324 (1998)

[6] Hochreiter, S.: Untersuchungen zu Dynamischen Neuronalen Netzen. *Diploma, Technische Universität München* 91(1), 48–59, (1991)

[7] Bengio, Y., Simard, P.Y., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Transactions on Neural Networks and Learning Systems* 5(2), 157–166 (1994)

[8] Hochreiter, S.: The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(2), 107–116 (1998)

[9] Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural Computation* 9(8), 1735–1780 (1997)

[10] Cho, K., van Merrienboer, B., Gülcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In: *EMNLP*, pp. 1724–1734 (2014)

[11] Józefowicz, R., Zaremba, W., Sutskever, I.: An Empirical Exploration of Recurrent Network Architectures. In: *ICML*, pp. 2342–2350 (2015)

[12] Shang, L., Lu, Z., Li, H.: Neural Responding Machine for Short-Text Conversation. In: *ACL*, pp. 1577–1586 (2015)

[13] Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Urtasun, R., Torralba, A., Fidler, S.: Skip-Thought Vectors. In: *NIPS*, pp. 3294–3302 (2015)

[14] Walther, D., Itti, L., Riesenhuber, M., Poggio, T.A., Koch, C.: Attentional Selection for Object Recognition - A Gentle Way. In: *BMCV*, pp. 472–479 (2002)

[15] Itti, L., Koch, C.: Computational Modelling of Visual Attention. *Nature Reviews Neuroscience* 2(3), 194 (2001)

[16] Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent Models of Visual Attention. In: *NIPS*, pp. 2204–2212 (2014)

[17] Zhao, B., Wu, X., Feng, J., Peng, Q., Yan, S.: Diversified Visual Attention Networks for Fine-Grained Object Classification. *IEEE Transactions on Multimedia* 19(6), 1245–1256 (2017)

[18] Wang, F., Tax, D.M.J.: Survey on the Attention Based RNN Model and Its Applications in Computer Vision. *CoRR*, abs/1601.06823 (2016)

[19] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In: *ICML*, pp. 2048–2057 (2015)

[20] Luong, T., Pham, H., Manning, C.D.: Effective Approaches to Attention-Based Neural Machine Translation. In: *EMNLP*, pp. 1412–1421 (2015)

[21] Bluche, T., Louradour, J., Messina, R.O.: Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In: *ICDAR*, pp. 1050–1055 (2017)
[22] Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: *CVPR*, pp. 3156–3164 (2015)
[23] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144 (2016)
[24] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R.J., Darrell, T., Saenko, K.: Sequence to Sequence - Video to Text. In: *ICCV*, pp. 4534–4542 (2015)
[25] Le, Q.V., Jaitly, N., Hinton, G.E.: A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *CoRR*, abs/1504.00941 (2015)
[26] Li, S., Li, W., Cook, C., Zhu, C., Gao, Y.: Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN. In: *CVPR*, pp. 5457–5466 (2018)
[27] Bradbury, J., Merity, S., Xiong, C., Socher, R.: Quasi-Recurrent Neural Networks. *CoRR*, abs/1611.01576 (2016)
[28] Graves, A.: Multi-Dimensional Recurrent Neural Networks. *ICANN* 4668, 549–558 (2007)
[29] Kalchbrenner, N., Danihelka, I., Graves, A.: Grid Long Short-Term Memory. *CoRR*, abs/1507.01526 (2015)
[30] Treisman, A., Gelade, G.: A Feature-Integration Theory of Attention. *Cognitive Psychology* 12, 97–136 (1980)
[31] Lu, J., Xiong, C., Parikh, D., Socher, R.: Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *CoRR*, abs/1612.01887 (2016)
[32] Liu, G., Guo, J.: Bidirectional LSTM with Attention Mechanism and Convolutional Layer for Text Classification. *Neurocomputing* 337, 325–338 (2019)
[33] Luong, M., Pham, H., Manning, C.D.: Effective Approaches to Attention-Based Neural Machine Translation. *CoRR*, abs/1508.04025 (2015)
[34] Britz, D., Goldie, A., Luong, M., Le, Q.V.: Massive Exploration of Neural Machine Translation Architectures. *CoRR*, abs/1703.03906 (2017)
[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. *CoRR*, abs/1706.03762 (2017)
[36] Tian, Y., Hu, W., Jiang, H., Wu, J.: Densely Connected Attentional Pyramid Residual Network for Human Pose Estimation. *Neurocomputing* 347, 13–23 (2019)
[37] Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. In: *AAAI*, pp. 4263–4270 (2017)
[38] Zhang, P., Xue, J., Lan, C., Zeng, W., Gao, Z., Zheng, N.: Adding Attentiveness to the Neurons in Recurrent Neural Networks. In: *ECCV*, pp. 136–152 (2018)
[39] Song, K., Yao, T., Ling, Q., Mei, T.: Boosting Image Sentiment Analysis with Visual Attention. *Neurocomputing* 312, 218–228 (2018)
[40] Chung, J., Gülcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555 (2014)
[41] Ba, L.J., Kiros, R., Hinton, G.E.: Layer Normalization. *CoRR*, abs/1607.06450 (2016)
[42] Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent Models of Visual Attention. *CoRR*, abs/1406.6247 (2014)
[43] Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330 (1993)
[44] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning Word Vectors for Sentiment Analysis. In: *ACL*, pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA (June 2011)
[45] Wang, S., Manning, C.: Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. ACL (07 2012)

Chapter 3

Attention Mechanism-Based Long Short-Term Memory Model

Guoqiang Zhong, Xin Lin, Kang Chen, and Qingyang Li
College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract

In the deep learning area, the long short-term memory (LSTM) model could remember sequential input information, but it cannot pay special attention to the part of the sequences. That is to say, the LSTM model lacks the attention mechanism. To address this problem, we propose a novel method in which the attention mechanism is seamlessly integrated with the LSTM model, and this method is called attention mechanism-based long short-term memory (AM-LSTM). More than processing the long short-term dependencies information, the AM-LSTM model can focus on important information of the sequences with the attention mechanism inside the cell of LSTM. Extensive experiments demonstrate that the AM-LSTM model outperforms LSTM and related models on the sequence learning tasks.

Key Words: Deep learning, Sequence learning, Attention mechanism, Long short-term memory

3.1 Introduction

With attention mechanism, humans can naturally focus on important information and ignore irrelevant information during one's perception and cognition [1, 2]. Based on this fact, many brain-inspired learning models have been deeply studied and widely applied in recent years [3–10]. However, although many deep learning models can learn effective representations of data and memorize the information, they cannot pay attention to important parts of the data. For example, a long short-term memory (LSTM) [11] model is widely used for sequence learning. However, it has no attention mechanism.

To address this problem, some studies have tried to apply the attention mechanism to LSTM. Nevertheless, most of these models only add the attention mechanisms outside the LSTM cells and have not thoroughly solved the issue that LSTM has no attention mechanism itself [2, 12, 13].

Here, we propose a novel method called attention mechanism-based long short-term memory (AM-LSTM), which seamlessly integrates the attention mechanism into the inner cell of LSTM. In this case, an AM-LSTM model can remember historical information and notice crucial details in the sequences. Through the experiments, the advantage of the AM-LSTM model over LSTM is demonstrated.

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

3.2 Related Work

In this section, we review some earlier work related to AM-LSTM, including LSTM and several models using the attention mechanism.

3.2.1 LSTM

LSTM is a powerful learning model for sequential data and has been widely used in many learning tasks, such as speech recognition and handwritten character recognition [14, 15]. The cell of LSTM includes an input gate, a forget gate, and an output gate. These gates and the state of the cell can be calculated as follows:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right), \tag{3.1}$$

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right), \tag{3.2}$$

$$\tilde{C}_t = \tanh\left(W_{\tilde{c}}[h_{t-1}, x_t] + b_{\tilde{c}}\right), \tag{3.3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{3.4}$$

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right), \tag{3.5}$$

$$h_t = o_t * \tanh(C_t). \tag{3.6}$$

Here, $W_f, W_i, W_{\tilde{c}}, W_o$ are the weight parameters and $b_f, b_i, b_{\tilde{c}}, b_o$ are the biases. The forget gate f_t can change the cell state by forgetting the previous moment information. Moreover, the input gate i_t and output gate o_t can decide what information will be input to the LSTM cell and output at the current moment, respectively. These three gates are important parts of the LSTM cell, which is applied to update the current state of the LSTM cell C_t and derive new cell output h_t.

In order to optimize the performance of LSTM, many extensions of LSTM have been presented in recent years [16–19]. In [20], the spatiotemporal LSTM (ST-LSTM) cells are designed for memorizing both spatial and temporal information. [21] introduces a convolutional LSTM (ConvLSTM) model, which extends the fully connected LSTM to have convolutional architectures in both the input-to-gate and gate-to-gate transitions. In addition, [17] proposes a tensorized LSTM model, which represents the hidden states with tensors.

As mentioned earlier, LSTM and its extension models mainly focus on processing the sequential data but cannot pay attention to the important parts of the sequences. In this chapter, we present a model that can integrate the attention mechanism into the inner cell of the LSTM.

3.2.2 Models Using the Attention Mechanism

In the human cognitive system, humans can pay attention to important information and ignore unrelated information by attention mechanism [2, 22, 23]. Similarly, in the machine cognitive

and learning area, the main function of the attention mechanism is selection and allocation [1, 10]. That is, with an effective information choice ability, the attention mechanism can concentrate the computing power on the important parts of input data [1]. In recent years, the attention mechanism has been widely applied, such as in the work to resolve the human visual neural computational problem [24] and model the retrieval mechanism of associations from the associative memory [25].

Furthermore, many attention-based deep learning models have been proposed recently. For example, [26] proposes the structured attention model, which incorporates graphical models to generalize simple attention. Alternatively, [27] presents a self-attention mechanism network, which can replace the common recurrent and convolutional networks. This network completely relies on the attention mechanism to compute feature representations of its input and output. In addition, in [5], the selective attention for identification model (SAIM) is used for visual search applications. The SAIM simulates the human ability to complete translation invariant recognition of multiple scenes. In addition, in [28], a recurrent attention mechanism model is introduced. It is an end-to-end memory learning model applied to language modeling tasks.

As mentioned earlier, several attention-based models have been presented to handle visual or language processing problems. However, rare work has integrated the attention mechanism inside the cell of LSTM to improve its performance in sequence learning.

3.3 The Proposed Model

In this section, we introduce the proposed AM-LSTM model in detail, which seamlessly integrates an attention gate into the cell of LSTM. For clarity, we first depict the added attention gate in Section 3.1 and then describe the architecture and learning of the AM-LSTM model in Section 3.2.

3.3.1 The Attention Gate

Figure 3.1 shows the structure of the attention gate of the AM-LSTM model, which receives the inputs from the input gate and the forget gate. Equation (3.7) is the computational formula of the attention gate:

$$A_t = \psi\left(\hat{A}_t[f_t, i_t],\ \tilde{A}_t[f_t, i_t]\right) = \hat{A}_t \bigotimes \tilde{A}_t, \tag{3.7}$$

where $\hat{A}_t$ and $\tilde{A}_t$ are defined as follows:

$$\hat{A}_t = \sigma\left(W_{\hat{a}}[f_t, i_t] + b_{\hat{a}}\right), \tag{3.8}$$

$$\tilde{A}_t = \tanh\left(W_{\tilde{a}}[f_t, i_t] + b_{\tilde{a}}\right). \tag{3.9}$$

Here, $W_{\tilde{a}}$ and $W_{\hat{a}}$ are the weight parameters, while $b_{\tilde{a}}$ and $b_{\hat{a}}$ are the biases. The sigmoid function σ is used to compute the value of $\hat{A}_t$, which denotes the ratios of the attention elements, as

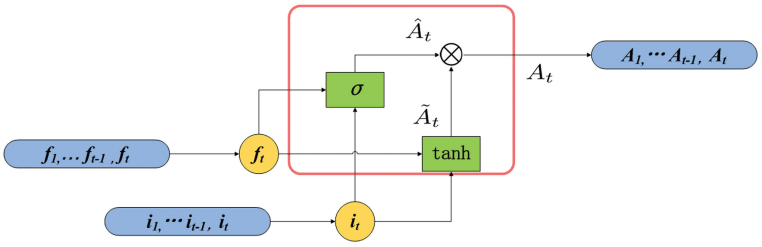


Figure 3.1 An illustration of the attention gate.

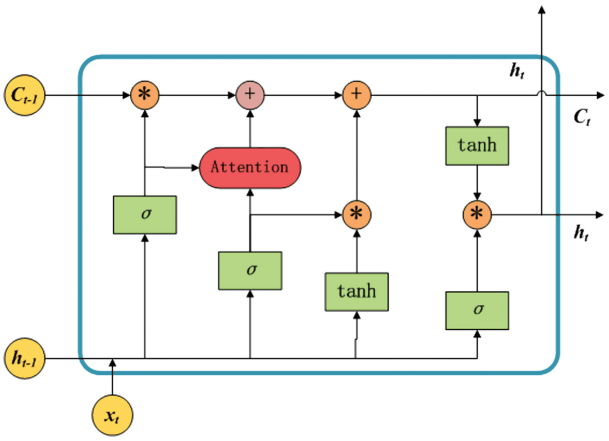


Figure 3.2 The AM-LSTM cell. The module with red color is the attention gate.

shown in Equation (3.7). Moreover, the tanh function is used to derive the candidate attention values $\tilde{A}_t$.

In Equation (3.7), $\bigotimes$ is the element-wise multiplication. We multiply the elements between $\tilde{A}_t$ and $\hat{A}_t$ to derive the output of the attention gate A_t. The attention gate determines the attention distribution of the information in the current cell. In the following, we introduce how it can be seamlessly integrated into the LSTM cell.

3.3.2 AM-LSTM

In order to seamlessly integrate the attention mechanism into the LSTM, we propose the AM-LSTM model, which integrates the attention gate introduced above into the cell of LSTM. Figure 3.2 is a diagram of the AM-LSTM cell. Particularly, the AM-LSTM model consisting of the AM-LSTM cells can focus on important information in the sequences during its learning process.

The AM-LSTM model inherits the three gates (forget gate, input gate, and output gate) of the LSTM model. For updating the AM-LSTM cell state, we can compute it as

$$\begin{aligned}\hat{C}_t &= C_t + A_t \\ &= f_t * C_{t-1} + i_t * \tilde{C}_t + A_t \\ &= f_t * C_{t-1} + i_t * \tilde{C}_t + A_t \bigotimes \hat{A}_t.\end{aligned} \tag{3.10}$$

Here, A_t is the output of the attention gate and C_t is the original LSTM cell state. In this case, we integrate the attention mechanism into the LSTM unit, such that the new AM-LSTM model can not only memorize the sequential information but also pay attention to important information in the sequences.

Accordingly, the output gate of the AM-LSTM model can be updated as

$$h_t = o_t * \tanh(\hat{C}_t). \tag{3.11}$$

In the AM-LSTM model, the attention mechanism is applied inside the LSTM cell, unlike previous attention-based LSTM models, in which attention is added after the whole sequence has been handled by all LSTM cells. Therefore, the AM-LSTM model is quite different from the LSTM model and most of its attention-based variants. The AM-LSTM model enables the sequence learning to focus on important parts of the input data and automatically ignore irrelevant parts to improve its performance.

3.4 Experiments

To evaluate the proposed AM-LSTM model, we have conducted extensive experiments on two sequence learning tasks: image classification and semantic analysis. In the following, we report the experimental settings and results.

3.4.1 Experiments on the Image Classification Task

In this section, we used the MNIST and Fashion-MNIST datasets to verify the performance of the AM-LSTM model. MNIST is a handwritten digit dataset. It contains 70,000 28×28 grayscale images, which belong to 10 classes. Of all images, 60,000 are used for training and the other 10,000 for testing [29]. Alternatively, Fashion-MNIST is an image dataset, while its image format and number are identical to the MNIST dataset [30]. In our experiments, we considered the rows of an image as sequential data to perform image classification.

To verify the performance of the AM-LSTM model, we compared it with some related models. As the AM-LSTM model integrates the attention mechanism inside the LSTM cell, the most closely related model to the AM-LSTM model is LSTM. Therefore, LSTM is our compared baseline. In addition, we compared the AM-LSTM model with the gated recurrent unit (GRU) [31], bidirectional LSTM (Bi-LSTM) [32], and nested LSTM (NLSTM) [33] models in our experiments.

Table 3.1 demonstrates the image classification results obtained by the AM-LSTM model and the compared models on the MNIST and Fashion-MNIST datasets. As we can see, the

Table 3.1 Accuracy obtained by AM-LSTM model and related models on the MNIST and Fashion-MNIST datasets

Data set	LSTM (%)	GRU (%)	Bi-LSTM (%)	NLSTM (%)	**AM-LSTM (%)**
MNIST	97.47	97.79	97.81	97.75	**97.85**
Fashion-MNIST	87.46	88.16	88.18	88.32	**86.60**

AM-LSTM model outperforms all compared models. This shows the advantage of the AM-LSTM model over LSTM and other related models during image classification learning.

To further analyze the advantage of AM-LSTM over LSTM, we draw the learning curves of LSTM and AM-LSTM models acquired on the Fashion-MNIST dataset in Figure 3.3. Figure 3.3(a) denotes the accuracy curves against the training steps, while Figure 3.3(b) is the loss curves against the training steps. As can be seen from these figures, the AM-LSTM model consistently performs better and converges faster than the LSTM model.

3.4.2 Experiments on the Sentiment Analysis Task

Sentiment analysis is an interesting task in the deep learning area [34–36]. In order to evaluate the performance of the AM-LSTM model, we conducted experiments on both the ordinary sentiment analysis and aspect-based sentiment analysis tasks.

3.4.2.1 Ordinary Sentiment Analysis

In this experiment, we used the internet movie review database (IMDb) [35] dataset to verify the effect of the AM-LSTM model on an ordinary sentiment analysis task. IMDb is a crawler dataset about internet film reviews. Based on the emotion polarities of the reviews, this dataset divides all movie reviews into positive and negative categories.

In our work, we compared AM-LSTM with LSTM and a hybrid deep belief network (HDBN) [37], which is an effective deep network for sentiment analysis. The error rates and the running time of the AM-LSTM model and the compared models are shown in Figure 3.4(a) and 3.4(b), respectively. As we can see, among the compared model, AM-LSTM obtains the best classification accuracy and uses the least running time.

3.4.2.2 Aspect-Based Sentiment Analysis

Aspect-based sentiment analysis is one of the classical tasks of semantic analysis [38]. In order to test the effect of the AM-LSTM model on aspect-based sentiment analysis, we conducted experiments on two datasets. One was the SemEval-2014 Task 4 (SemEval14) dataset [36], which contains two domains (Restaurant domain and Laptop domain). The other was the

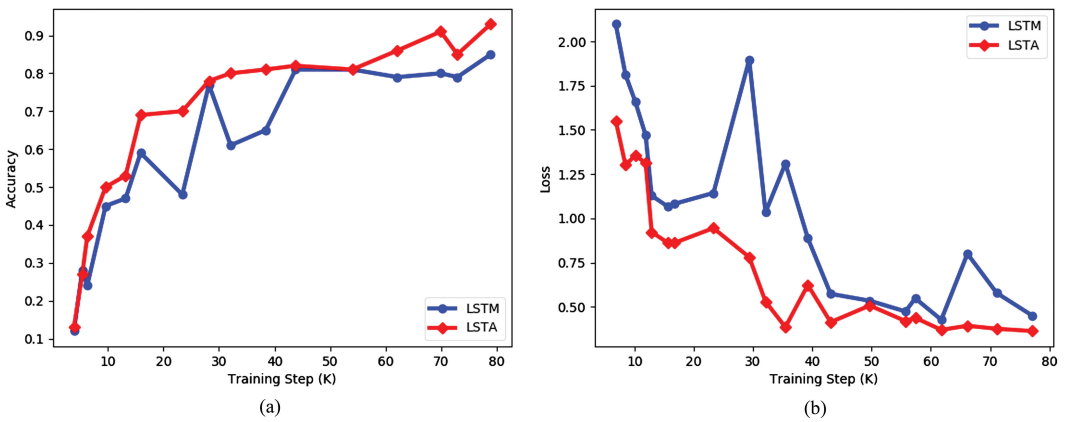


Figure 3.3 The accuracy and loss curves of LSTM and AM-LSTM model on the Fashion-MNIST dataset. (a) The accuracy curves; and (b) the loss curves.

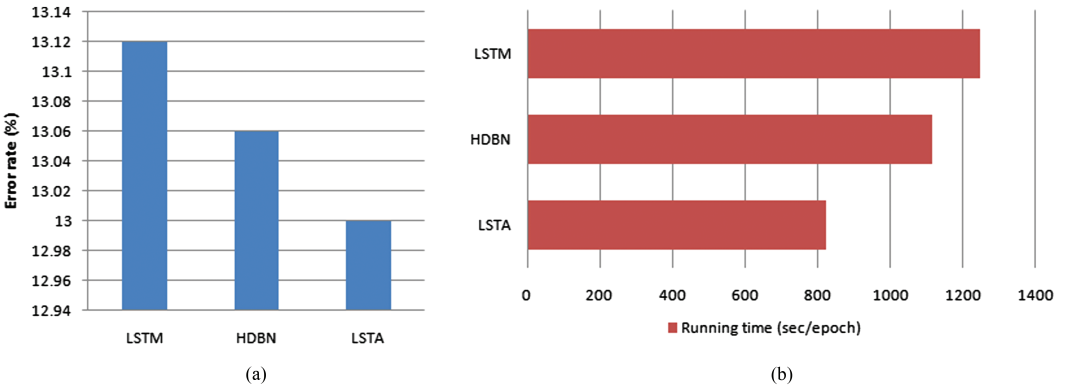


Figure 3.4 The error rates and the running time of LSTM, HDBN, and AM-LSTM model on the IMDb dataset. (a) The error rate obtained by the three models; (b) their running time.

Table 3.2 The statistics of the SemEval14 and Twitter datasets

Data set	Positive		Neutral		Negative	
	Train	Test	Train	Test	Train	Test
SemEval14(Restaurant)	2164	728	637	196	807	196
SemEval14(Laptop)	994	341	464	169	870	128
Twitter	1561	173	3127	346	1560	173

Table 3.3 The experimental results obtained on the SemEval14 and Twitter datasets

Method	SemEval14 (Restaurant)		SemEval14 (Laptop)		Twitter	
	Acc.(%)	Macro-F1	Acc.(%)	Macro-F1	Acc.(%)	Macro-F1
Cabasc	78.12	0.6743	70.84	0.6552	69.51	0.6707
ATAE-LSTM	77.86	0.6718	69.75	0.6425	69.65	0.6762
LSTM	77.41	0.6686	69.74	0.6394	68.93	0.6699
AM-LSTM	**78.57**	**0.6801**	**71.16**	**0.6559**	**69.94**	**0.6911**

Twitter dataset collected by Dong et al. [34]. In these two datasets, the aspect terms of each review are labeled by three sentiment polarities: positive, neutral, and negative. For example, in the sentence "I loved the food in this restaurant, but the service is horrible," the polarity of the aspect term *food* is positive, but for the aspect term *service*, its polarity is negative. Concretely, the statistics of the two datasets are provided in Table 3.2.

In this experiment, we used Accuracy and Macro-averaged F-measure (Macro-F1) as the metrics to evaluate the effect of AM-LSTM and the compared models [34, 39]. The experimental results obtained by AM-LSTM and the compared methods are shown in Table 3.3. In this table, "Cabasc" is a content attention model for aspect-based sentiment analysis [38] and "ATAE-LSTM" is an attention-based LSTM with aspect embedding, which can focus on the parts of a sentence when several aspects are taken as input [40]. Although there are many attention-based variants models, they are quite different from the AM-LSTM model. We can also apply the attention mechanism outside the AM-LSTM cell as same as them.

From Table 3.3, we can see that AM-LSTM performs best among the compared models. In other words, AM-LSTM outperforms both LSTM and previous attention-related models, including the model that applies the attention mechanism outside the cell of LSTM.

3.5 Conclusion

In this chapter, we propose a novel AM-LSTM model to alleviate the problem that LSTM lacks attention mechanism. The key idea behind this model is to seamlessly integrate the attention mechanism inside the cell of LSTM. Experiments show that the AM-LSTM model performs better than LSTM, variants of LSTM, and other related networks. Therefore, AM-LSTM can be seen as a substitute for LSTM in the sequence learning tasks.

References

[1] Posner, M.I.: Cognitive Neuroscience of Attention. Guilford Press (2011)
[2] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In: *ICML*, pp. 2048–2057 (2015)
[3] Aboudib, A., Gripon, V., Coppin, G.: A Biologically Inspired Framework for Visual Information Processing and an Application on Modeling Bottom-Up Visual Attention. *Cognitive Computation* 8(6), 1007–1026 (2016)
[4] Gao, F., Zhang, Y., Wang, J., Sun, J., Yang, E., Hussain, A.: Visual Attention Model Based Vehicle Target Detection in Synthetic Aperture Radar Images: A Novel Approach. *Cognitive Computation* 7(4), 434–444 (2015)
[5] Heinke, D., Backhaus, A.: Modelling Visual Search with the Selective Attention for Identification Model (VS-SAIM): A Novel Explanation for Visual Search Asymmetries. *Cognitive Computation* 3(1), 185–205 (2011)
[6] Katsuki, F., Constantinidis, C.: Bottom-Up and Top-Down Attention: Different Processes and Overlapping Neural Systems. *The Neuroscientist* 20(5), 509–521 (2014)
[7] Luo, B., Hussain, A., Mahmud, M., Tang, J.: Advances in Brain-Inspired Cognitive Systems. *Cognitive Computation* 8(5), 795–796 (2016)
[8] Taylor, J.G.: Cognitive Computation. *Cognitive Computation* 1(1), 4–16 (2009)
[9] Wang, Z., Ren, J., Zhang, D., Sun, M., Jiang, J.: A Deep-learning Based Feature Hybrid Framework for Spatiotemporal Saliency Detection inside Videos. *Neurocomputing* 287, 68–83 (2018)
[10] Wischnewski, M., Belardinelli, A., Schneider, W.X., Steil, J.J.: Where to Look Next? Combining Static and Dynamic Proto-objects in a TVA-based Model of Visual Attention. *Cognitive Computation* 2(4), 326–343 (2010)
[11] Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural Computation* 9(8), 1735–1780 (1997)
[12] Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A Structured Self-Attentive Sentence Embedding. In: *ICLR* (2017)
[13] Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: *EMNLP*, pp. 1412–1421 (2015)
[14] Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28(10), 2222–2232 (2017)
[15] Wöllmer, M., Eyben, F., Graves, A., Schuller, B.W., Rigoll, G.: Bidirectional LSTM Networks for Context-Sensitive Keyword Detection in a Cognitive Virtual Agent Framework. *Cognitive Computation* 2(3), 180–190 (2010)

[16] Graves, A., Mohamed, A.R., Hinton, G.: Speech Recognition with Deep Recurrent Neural Networks. In: *ICASSP*, pp. 6645–6649 (2013)
[17] He, Z., Gao, S., Xiao, L., Liu, D., He, H., Barber, D.: Wider and Deeper, Cheaper and Faster: Tensorized LSTMs for Sequence Learning. In: *NIPS*, pp. 1–11 (2017)
[18] Neil, D., Pfeiffer, M., Liu, S.C.: Phased LSTM: Accelerating Recurrent Network Training for Long or Event-Based Sequences. In: *NIPS*, pp. 3882–3890 (2016)
[19] Wang, P., Song, Q., Han, H., Cheng, J.: Sequentially Supervised Long Short-Term Memory for Gesture Recognition. *Cognitive Computation* 8(5), 982–991 (2016)
[20] Wang, Y., Long, M., Wang, J., Gao, Z., Philip, S.Y.: PredRNN: Recurrent Neural Networks for Predictive Learning Using Spatiotemporal LSTMs. In: *NIPS*, pp. 879–888 (2017)
[21] Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C.: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: *NIPS*, pp. 802–810 (2015)
[22] Corbetta, M., Shulman, G.L.: Control of Goal-Directed and Stimulus-Driven Attention in the Brain. *Nature Reviews Neuroscience* 3(3), 201–215 (2002)
[23] Yan, Y., Ren, J., Sun, G., Zhao, H., Han, J., Li, X., Marshall, S., Zhan, J.: Unsupervised Image Saliency Detection with Gestalt-Laws Guided Optimization and Visual Attention Based Refinement. *Pattern Recognition* 79, 65–78 (2018)
[24] Cutsuridis, V.: A Cognitive Model of Saliency, Attention, and Picture Scanning. *Cognitive Computation* 1(4), 292–299 (2009)
[25] Wichert, A.: The Role of Attention in the Context of Associative Memory. *Cognitive Computation* 3(1), 311–320 (2011)
[26] Kim, Y., Denton, C., Hoang, L., Rush, A.M.: Structured Attention Networks. In: *ICLR* (2017)
[27] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention Is All You Need. In: *NIPS*, pp. 6000–6010 (2017)
[28] Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-End Memory Networks. In: *NIPS*, pp. 2440–2448 (2015)
[29] LeCun, Y., Cortes, C., Burges, C.: MNIST Handwritten Digit Database. AT&T Labs [Online] 2 (2010)
[30] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747 (2017)
[31] Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555 (2014)
[32] Graves, A., Schmidhuber, J.: Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks* 18(5-6), 602–610 (2005)
[33] Moniz, J.R.A., Krueger, D.: Nested LSTMs. In: *ACML*, pp. 530–544 (2017)
[34] Dong, L., Wei, F., Tan, C., Tang, D., Zhou, M., Xu, K.: Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In: *ACL*, pp. 49–54 (2014)
[35] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning Word Vectors for Sentiment Analysis. In: *ACL-HLT*, pp. 142–150 (June 2011)
[36] Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., Manandhar, S.: SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In: *SemEval@COLING*, pp. 27–35 (2014)
[37] Yan, Y., Yin, X.C., Li, S., Yang, M., Hao, H.W.: Learning Document Semantic Representation with Hybrid Deep Belief Network. *Computational Intelligence and Neuroscience* 2015, 650527:1–650527:9 (2015)
[38] Liu, Q., Zhang, H., Zeng, Y., Huang, Z., Wu, Z.: Content Attention Model for Aspect Based Sentiment Analysis. In: *WWW*, pp. 1023–1032 (2018)
[39] Tang, D., Qin, B., Liu, T.: Aspect Level Sentiment Classification with Deep Memory Network. In: *EMNLP*, pp. 214–224 (2016)
[40] Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based LSTM for Aspect-Level Sentiment Classification. In: *EMNLP*, pp. 606–615 (2016)

Chapter 4

Industrial Surface Defect Detection Based on Multi-Attention Fusion Mechanism and Federated Learning

Xiaoli Yue and Guoqiang Zhong
College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract

The rise of artificial intelligence provides a new method for industrial surface defect detection. However, the lack of sufficient labeled defect samples and the desire for greater detection accuracy are the major obstacles to deep learning-based defect detection. To address the aforementioned issues, this chapter proposes the novel data augmentation algorithm mixForm and the multi-attention fusion mechanism (MAF). First, the MAF focuses on the spatial and channel features of the defect samples and integrates these features into the spatial pyramid structure with the attention mechanism, which enables the classification and localization of complex defect features and improves the accuracy of defect detection. Second, this chapter proposes a data augmentation scheme for defect detection. The spatial and shape-wise transformations for target defects improve the detection model's ability to recognize multiple types of defects. Third, this chapter considers edge computing in industrial scenarios to achieve a federated learning (FL) framework for building defect detection models in low computing power situations with multiple devices cooperatively, called split federated learning (SFL). Meanwhile, the defect detection algorithm based on the MAF module is applied to the SFL framework. This effectively improves the model training efficiency and maintains high-accuracy detection for defective samples. Finally, the experimental results show that the defect detection model with the MAF module and mixForm data augmentation achieves higher detection accuracy than the comparison scheme, reaching 82.91 mAP on the NEU-DET dataset. Compared with other attention mechanisms, the defect detection technique employing MAF improves by at least 1.89 mAP. The experiments validate that SFL achieves faster convergence and higher detection accuracy than Faster R-CNN using traditional FL.

Key Words: Surface defect detection, Attention mechanism, Federated learning, Data augmentation

4.1 Introduction

Smart manufacturing is a new manufacturing model that applies artificial intelligence to industry. Smart manufacturing can transform the original complex and labor-intensive manufacturing process into an intelligent and efficient machine-automated production process. Industrial surface defect detection is one of the key aspects of smart manufacturing. Traditional defect

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

detection is done by manual or machine vision algorithms based on hough transform [1], wavelet transform [2], and Fourier transform [3] to select products containing defects, which is not only less accurate but also time-consuming. Nowadays, the rapid development of artificial intelligence has led to the creation of new defect detection solutions. The use of object detection algorithms allows for automated and rapid inspection of defects in products [4].

The current research on defect detection based on object detection is divided into two main categories: one-stage algorithm and two-stage algorithm. Among the one-stage algorithms, typical algorithms include YOLO [5], RetinaNet [6], and SSD [7]. Such algorithms sacrifice part of the detection accuracy in exchange for having high detection efficiency. Among two-stage algorithms, typical algorithms include Faster R-CNN [8] and Cascade R-CNN [9]. These algorithms improve detection accuracy by increasing the screening of candidate frames. However, the above two types of algorithms still have the problem of low detection accuracy for the detection of complex defects. As shown in Figure 4.1, a typical defect sample of rolled steel has various types of defects, small defect targets, and inconspicuous features. This makes it difficult for the existing defect detection to accurately locate and identify the defective regions in the sample. At the same time, the lack of sufficient annotation data is also a challenge that hinders further improvement of detection accuracy. Therefore, this chapter tries to propose a new defect detection scheme from the perspective of model design and data processing.

The scheme in this chapter is divided into two parts: multi-attention fusion mechanism (MAF) and mixForm data augmentation algorithm. MAF is inspired by the attention mechanism in deep learning, and it is proposed by applying the attention mechanism that fuses multiple functions for defect detection. Because of the single function of extracting features in the previous attention mechanisms, it is difficult to acquire various features in defect samples. MAF can obtain a variety of features to enable the subsequent modules to detect defects better. Unlike the direct masking or splicing data augmentation methods such as MixUp [10] and CutMix [11], mixForm takes advantage of the more homogeneous background in the defect samples. The mixForm recombines defective regions and incorporates them into the defect-free image

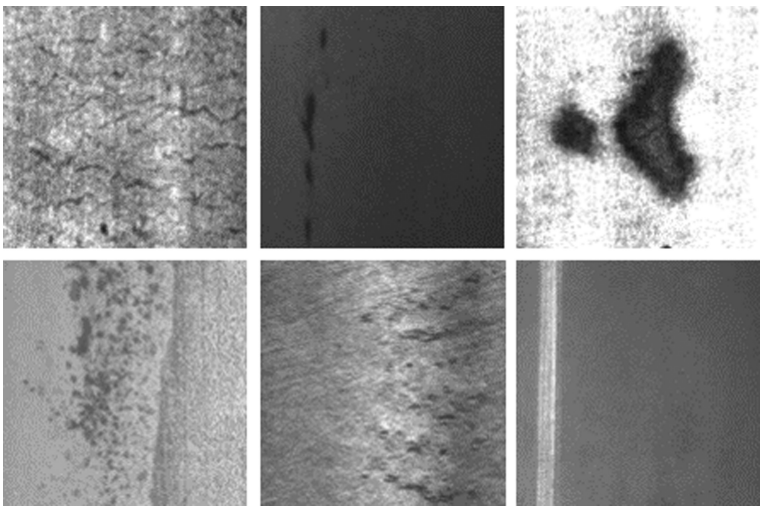


Figure 4.1 Example of defects on the steel surface.

to generate a new defect sample. Both MAF and mixForm can be successfully applied to the detection algorithm and to improve the accuracy of defect detection. In this chapter, the training problem of defect detection is further considered. Training a defect detection model using FL [12] allows the training task to be assigned to multiple smart terminals. This distributed idea can effectively save the training time of the model. Also, we optimize the traditional federation learning algorithm to propose the SFL algorithm, which greatly reduces communication overhead. The specific contributions of this chapter are as follows.

(1) To enhance the feature extraction capability of the network, this chapter proposes an attention mechanism based on multidimensional information fusion. The MAF integrates the location and shape information of defects into the defect detection model, which improves the defect detection accuracy while avoiding adding more model complexity.
(2) To fully use the limited annotation data, this chapter designs the mixForm data augmentation scheme applicable to defect detection. The mixForm expands the dataset and effectively enhances the model's ability to recognize multiple defects, improving detection accuracy.
(3) To promote the application of defect detection in industrial scenarios, this chapter considers the problem of training defect detection models collaboratively by numerous intelligent devices. Applying SFL to it, we can achieve the defect detection model with high computational efficiency and generalizability while ensuring data privacy.
(4) The experimental results show that the mixForm data augmentation algorithm effectively enriches the data diversity. The MAF with mixForm achieves higher accuracy detection of target defects and reaches 82.91 mAP on the NEU-DET dataset. Using MAF, the defect detection scheme achieves at least 1.89 mAP improvement over defect detection using SE (Squeeze-and-Excitation) [13], CBAM (Convolutional Block Attention Module) [14], and CA (coordinate attention) [15] attention. Experiments also validate the effectiveness of MAF on SFL. Compared to Faster R-CNN with FL, SFL achieves faster convergence speed and higher detection accuracy.

4.2 Related Work

In this section, we review the defect detection approaches from the perspective of manual features and deep learning.

For the traditional defect detection methods, the features used by the algorithm are broadly classified into three categories: texture, color, and shape [16]. For example, Prasitmeeboo *et al.* [17] designed a two-step SVM method using color histogram features and implemented a defect detection function for particle board using smoothing and thresholding techniques. Wang *et al.* [18] used the fast-hough transform technique to achieve defect detection on the bottle surface. Nevertheless, defect detection faces more significant challenges in natural complex industrial environments, such as diverse types of defects, slight differences from the background, difficulty in distinguishing defects, and noisy images. Traditional machine vision algorithms have the problems of low detection precision, detection of only specific features, and poor robustness.

For the deep learning methods, defect detection can be considered a practical application of object detection models in industrial scenarios to obtain accurate locations and category information. Moreover, algorithms based on deep learning gradually occupy a critical position.

Jin *et al.* [19] achieved 96.74 mAP using an improved Faster R-CNN for rail surface defect detection. Huang *et al.* [20] proposed a multi-scale feature pair approach to improve Cascade R-CNN to accomplish defect detection of metal cans. Liu *et al.* [21] used the MobileNet-SSD network to achieve 94.3 mAP on the support component dataset of a high-speed railway catenary. Zhang *et al.* [22] improved the YOLO-v3 network by introducing a focal loss function and pre-training weights, thus improving detection precision. Cheng *et al.* [23] improved the performance of the RetinaNet model by optimizing anchor configuration and introducing channel attention and adaptive feature fusion modules. However, there are still challenges in deep learning-based defect detection, such as the lack of generalization of the model and low precision of detection for small objects.

4.3 Industrial Surface Defect Detection Scheme

In this section, we introduce the proposed MAF in the scheme, the mixForm data augmentation algorithm, and the SFL framework for industrial defect detection.

4.3.1 Multi-Attention Fusion Mechanism

The detailed structure of MAF is shown in Figure 4.2, which encodes both channel and spatial dimensions. MAF helps the defect detection model find vital information in the image by focusing on "what are the features" and "where are the features." The channel attention module introduces a spatial pyramid structure that captures semantic information using different pooling sizes to construct different structural information. Moreover, the spatial attention module performs contextual modeling of spatial pixel points. The proposed attention mechanism of the scheme considers the inter-channel and spatial information interactions. It enhances the data's global and local correlation to develop a more comprehensive feature attention approach than

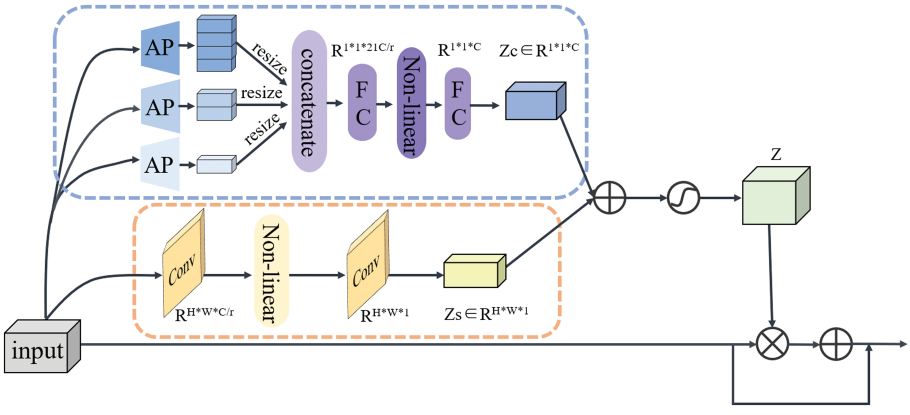


Figure 4.2 A schematic diagram of the architecture of MAF is proposed in this chapter. The blue dashed box refers to the channel dimension, and the orange dashed box refers to the spatial dimension. Here, r is the reduction factor for both channel and space branches, and we set it to 16. In addition, $\oplus$ denotes element-wise sum, and $\otimes$ denotes element-wise product.

other single-dimensional attention mechanisms in the following, we introduce the individual modules separately.

4.3.1.1 Channel Attention Module

Channel attention can effectively learn the relevance of each channel, that is, the importance of each channel, and strengthen the essential features to weaken unimportant features. Given an input feature map $X \in R^{H*W*C}$, the average pooling is performed by three different size pooling kernels (1*1, 2*2, 4*4) to obtain different scale features, preserving the structural information of the feature map. Then, the 1D feature vector is obtained by reshaping and concatenating operations. Finally, the operation of nonlinear activation can derive the output attention map of the channel module. Equation (4.1) is shown below.

$$Z_c = Conv\Big(\delta\Big(Conv\Big(C\big(R(P_1), R(P_2), R(P_4)\big)\Big)\Big)\Big), \tag{4.1}$$

where P_1, P_2, and P_4 represent the three scales of pooling operations to obtain three feature maps with $X_1 \in R^{1*1*C}$, $X_2 \in R^{2*2*C}$, and $X_4 \in R^{4*4*C}$, respectively. This reduces the loss of spatial information caused by the global average pooling operation. $R(\cdot)$ refers to the reshape operation, which yields three 1D vectors with $v_1 = c$, $v_2 = 4c$, and $v_3 = 16c$, respectively. $C(\cdot)$ refers to the concatenate operation. δ is the ReLu nonlinear activation function, and Conv is two 1*1 convolution functions. They are stitched together and convolved, and this step is an interaction of the information on multiple feature maps.

4.3.1.2 Spatial Attention Module

Because the distribution of defect features is uneven on different image pixels, spatial attention can effectively focus the model on informative features and strengthen the ability of feature representation in crucial regions, thus strengthening the regions of interest and weakening the background regions. Therefore, this scheme designs a spatial pixel attention module, focusing on generating respective weights for each pixel point. The feature map $X \in R^{H*W*C}$ is similarly fed into the 1*1 convolution and nonlinear activation function to obtain the output attention map $Z_s \in R^{H*W*1}$ of the spatial module, which can be described as follows:

$$Z_s(X) = Conv\Big(\delta\Big(Conv(X)\Big)\Big). \tag{4.2}$$

4.3.1.3 Integration Method

The above two modules are parallel. First, the obtained attention maps Z_c and Z_s are combined by add operation. Then, after the sigmoid function normalizes Z_c and Z_s to between (0,1) and obtains the whole attention map $Z(X) \in R^{H*W*C}$, it is applied to the input feature map to obtain the final weighted output feature map X'. The calculation process can be expressed as follows:

$$Z(X) = \sigma\big(Z_c(X) + Z_s(X)\big), \tag{4.3}$$

$$X' = X + X \otimes Z(X). \tag{4.4}$$

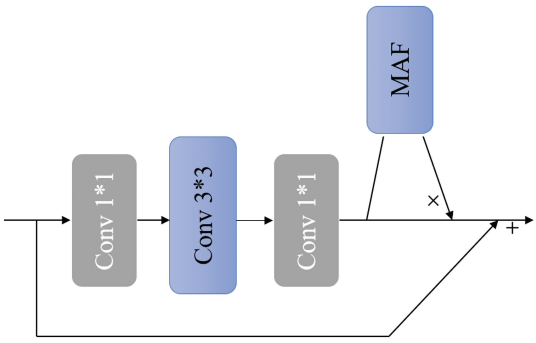


Figure 4.3 Implementation approach for ResNet network architecture.

4.3.1.4 Implementation

The attention mechanism can be flexibly inserted into the defect detection models as a plug-and-play module to improve the detection performance. In the scheme, the aim is to capture long-term dependencies and improve the backbone network to enhance feature extraction; therefore, the proposed MAF is inserted behind the residual block of ResNet [24] (*i.e.,* behind the one with more channels). Figure 4.3 shows how attention is added to the backbone.

4.3.2 MixForm Data Augmentation

Traditional augmentation methods use horizontal and vertical flips, changing brightness, and Gaussian perturbation [25]. These methods reduce the dependence of models on the location of the object's appearance and reduce the models' sensitivity to color. However, the traditional methods have limited improvement in the accuracy of the defect detection model. Furthermore, methods like MixUp [10], CutMix [11], and mosaic destroy the structural information of the image and introduce some unnecessary defects. Therefore, to get sufficient training data and make full use of the limited annotation information, this chapter proposes a data augmentation algorithm for industrial surface defects called mixForm. We consider that the image background is primarily consistent with industrial defects; so we propose a diverse combination of defect features and background fusion, and the algorithm is shown below.

First, the defective region image D_{defect} is cropped from the defective image according to the annotation information and input to the algorithm together with the clean sample dataset D_{clean}. Second, the algorithm randomly selects a clean image and multiple defective region images. Moreover, it randomly transforms these defective region images in shape and size. The algorithm calculates the percentage of the overlapping area of these defective region images to avoid adding defective region images with a too-sizeable overlapping area on the background image. Finally, to prevent the defective region images from being too prominent in the background image due to their significant differences, the algorithm employs Poisson fusion to seamlessly integrate the defective region images into the background image. This approach enables the creation of a single image that encompasses multiple defects, resulting in more realistic images that align with the industrial scenario. The dataset is expanded in the data preprocessing stage to improve the defect detection model's ability to recognize multiple defects.

Algorithm 1 MixForm data augmentation

Input: Images of all defective areas in the defect dataset D_{defect}, Clean sample dataset D_{clean}, Number of images generated by data augmentation N, Defective area overlap threshold T

Output: Data augmentation defects dataset D_{aug}

```
for i to N do
  img_c ← random_select(D_clean)
  // Randomly select one of the clean samples as the background image.
  N_defect ← random_int(2,4)
  // Randomly determine the number of defects present in an image.
  j ← 0
  while j < N_defect do
    d ← random_select(D_defect) // Randomly select a defective area image.
    s ← random_resize(d) // Randomly generate defect areas with shape and size.
    p ← get_position(d) // Generate defect areas and positions for background.
    if DecideOberlap(d, s, p, img_c) < T then
        // Calculate the overlap of the currently added defect area with the existing defects in the background.
        img_c ← seamlessClone(d, s, p, img_c)
        //Add defect areas to the background image.
        j = j + 1
      end if
  end while
  D_aug ← img_c // Add the generated images to the data augmentation dataset.
end for
return D_aug
```

4.3.3 Split Federated Learning

FL can be deployed on numerous intelligent devices to achieve distributed training in defect detection in the industrial Internet of Things. The aggregation approach demonstrated in Equation (4.5) is the core principle of the FedAvg [12] algorithm.

$$w^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t, \tag{4.5}$$

where $K\left(1 < K \le C\right)$ is the number of participants in the current communication round, n_k is the number of training data of the k-th participant, and n is the number of training data of all participants who participated in the current aggregation round.

However, due to the problems of device heterogeneity and data heterogeneity, devices with weak computational power will slow down the FL model parameter aggregation. Models with different data distributions will interfere with each other in the aggregation stage, which eventually significantly reduces the efficiency and effectiveness of FL. Therefore, SFL is inspired by transfer learning [26] to propose an FL algorithm applicable to object detection, which improves the computational efficiency and the generalization of the global model compared to FedAvg. The core principle of the algorithm is shown in Equation (4.6). The corresponding model parameters $w_{k,t}^s$ are obtained by extracting the generic part of the object detection model $f\left(\cdot\right)$ for different devices.

For example, devices with different computing power may use different numbers of ResNet blocks in the object detection model. The algorithm can select the part of the model that is

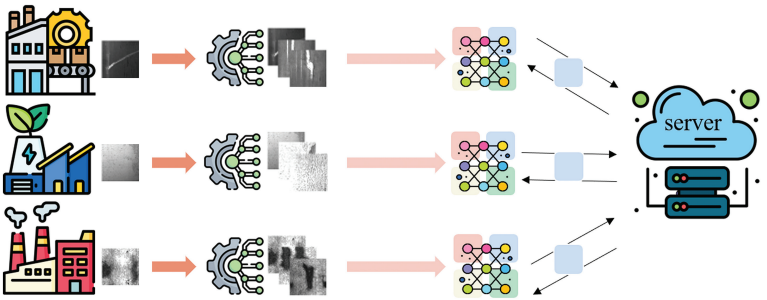


Figure 4.4 SFL defect detection process.

common and has better transferability. Then the model parameters $w_{k,t}^{s}$ are aggregated and distributed to all participants according to the FL process. The participants use the combination of local-specific model parameters $w_{k,t}^{f-s}$ and global model parameters $\sum_{k=1}^{K} \frac{n_k}{n} w_{k,t}^{s}$ as a new local model to continue the training. The SFL process is shown in Figure 4.4.

$$f_{k,t+1} = concat\left(w_{k,t}^{f-s} + \sum_{k=1}^{K} \frac{n_k}{n} w_{k,t}^{s} \right), \tag{4.6}$$

where $f_{k,t+1}$ is the local model for the k-th participant's $t+1$ communication round, and $f-s$ is denoted as the difference set between the model f and the generic part s (*i.e.*, the model-specific part). The heterogeneity problem of FL is effectively solved with the above algorithm, and the speed of model convergence is also improved.

4.4 Experiment

This section first describes the three parts of the dataset, performance evaluation, and implementation details required to conduct the experiments. The following experiment results for defect detection are shown, including comparison experiments with different algorithms, ablation experiments, and comparisons with other attention methods. Finally, defect detection results based on SFL are shown.

4.4.1 Dataset

To validate the performance of the defect detection model, we used the steel surface defect dataset NEU-DET [27] for evaluation. In addition, this dataset consisted of 1800 samples of six different types with the size of 200*200.

4.4.2 Performance Evaluation

For detection tasks, the model effectiveness is usually evaluated using the mean average precision (mAP), which is the average of the average precision (AP) over multiple categories. AP is

the area formed by the PR curve plotted against the axes based on the obtained precision and recall data; that is, it is a trade-off between these two. The above metrics can be defined as

$$mAP = \frac{1}{N}\sum AP, \tag{4.7}$$

$$AP = \int_0^1 precision\left(recall\right) d\left(recall\right), \tag{4.8}$$

$$precision = \frac{TP}{TP + FP}, \tag{4.9}$$

$$recall = \frac{TP}{TP + FN}, \tag{4.10}$$

where TP, FP, and FN represent the number of true positive, false positive, and false negative samples, respectively. N refers to the number of categories.

4.4.3 Implementation Details

4.4.3.1 Computation Platform

The model was implemented on PyCharm based on PyTorch 1.10.1, CUDA 11.3, Python 3.8, and the experiments were completed on Intel(R) Xeon(R) Silver 4214R CPU and NVIDIA RTX 3090 GPUs platform.

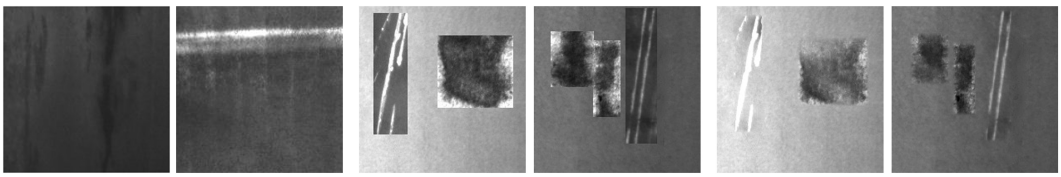
4.4.3.2 Parameter Setup

All experiments used a Faster R-CNN algorithm with COCO pre-trained ResNet50-FPN as the backbone. For fine-tuning, the learning rate was set to 0.02, and the weight decay and momentum were $1e^{-4}$ and 0.9, respectively, during the training process. Due to the limitation of GPU memory, the batch size was set to 10, and the epoch was 40. The training, validation, and test sets of the dataset were divided according to the ratio of 6.8:1.2:2.

4.4.4 Experiment Results

4.4.4.1 MixForm Data Augmentation

The mixForm data augmentation can address the drawback of having only the same type of defects in each image of the NEU-DET dataset. MixForm incorporated different defects into clean samples and improved the diversity of the training data. Figure 4.5(a) and Figure 4.5(c) compare the traditional data augmentation with mixForm. The traditional data augmentation only rotates, flips, and blurs the image, and the number of generated samples is limited and does not change much compared with the original image. MixForm can specify the number of defective features in the image, which is generally set to two to four in the experiment. Thus, a rich set of defect data can be generated, allowing the defect detection model to identify the defect location while correctly classifying the defects. In addition, the mixForm algorithm considers that multiple defects may overlap when added to the background image. Hence, the algorithm



(a) traditional (b) non background fusion (c) mixForm

Figure 4.5 Results generated by different data augmentation.

Table 4.1 Detection results on the NEU-DET dataset

Method	YOLOv3	RetinaNet	YOLOX	Faster R-CNN	DEA_RetinaNet	Ours	Ours*
mAP	69.40	75.33	74.77	75.21	79.11	**82.91**	78.98
crazing	**68.39**	53.02	48.23	40.29	60.93	55.42	43.87
inclusion	61.88	78.74	82.01	81.44	82.49	85.41	**85.74**
patches	71.44	93.33	88.95	92.93	94.27	**95.04**	94.14
pitted_surface	68.33	91.37	76.80	88.44	**95.79**	93.30	92.89
rolled-in_scale	72.66	62.00	65.71	60.47	67.16	**76.32**	61.60
scratches	73.71	73.49	89.40	96.90	74.05	**99.56**	98.49

determines whether overlap occurs between defects and thus guides the calculation of defect positions in the background image.

Figure 4.5(b) shows the data generated by mixForm without background fusion. It can be clearly seen that these defect features differ significantly from the background image after addition due to different illumination at the time of the shooting. Defect detection models can quickly identify defective regions in such images, but it is challenging to obtain accuracy improvement. Therefore, background fusion is significant for mixForm, and in Figure 4.5(c), the defects have been well integrated into the background. It is worth noting that to avoid the presence of the augmented training set data in the validation set, the model gets convincing results in prediction. Specifically, mixForm expands the data only for the training set.

4.4.4.2 Comparison of Accuracy with Different Models

To evaluate the effectiveness of our model, we compared it with several mainstream object detection algorithms. Table 4.1 shows the detection results on the NEU-DET dataset. It includes the AP values on various types of defects and the final mAP values. The compared algorithms include the one-stage object detection network represented by YOLOv3, YOLOX, and RetinaNet, and the two-stage object detection network represented by the original Faster R-CNN. The original dataset was first divided. The training set was expanded to 6120 images using mixForm and then trained using the defect detection model proposed in this study.

Table 4.1 shows the defect detection results of the methods mentioned above. Moreover, Ours* represents the detection results that do not use mixForm. Table 4.1 shows that the method proposed in this study achieved the highest precision of 82.91 mAP and the best detection results on the categories patches, rolled-in_scale, and scratches, with 95.04, 76.32, and 99.56, respectively. In the crazing category, where the defect features are less obvious, our method also significantly improves the detection precision to 55.42. Notably, the precision of our model only

using MAF has reached 78.98 mAP without expanding the dataset, and the highest detection precision of 85.74 is achieved in the inclusion category. The examples of the detection results of this scheme on the NEU-DET dataset are shown in Figure 4.6.

4.4.4.3 Ablation Studies

The ablation experiments used Faster R-CNN as a baseline to verify the effects of mixForm and MAF. Also, we verified the effect of the attention mechanism in the reduction factor r on the model performance.

MAF and mixForm. The baseline is the Faster R-CNN algorithm with ResNet50-FPN as the backbone. Table 4.2 shows that the third column is the improved backbone network using MAF. The mAP of the model is improved from 75.21 in baseline to 78.98, which effectively improves the model performance. The second column shows the results after mixForm expands the training set to 6120 images, and the number of validation and test sets remains the same, which increases 5.28 mAP compared to the baseline. When both modules are used, its mAP reaches a maximum of 82.91, which is 2.42 mAP higher than using the data augmentation scheme alone and 3.93 mAP higher than using only the attention mechanism.

The effect of *r*. The reduction factor r is a vital hyperparameter to reduce the number of channels of the model. To investigate the effect of different reduction factors of this attention mechanism on the model performance, we tried to vary the size of r on the NEU-DET dataset before and after the expansion, respectively, and observe the change in performance. We conducted experiments on the MAF-ResNet50-FPN-based architecture with r of 16, 24, and 32. As shown in Table 4.3, the model gradually performs better as r decreases on both the original and expanded datasets. The reduction factor decreases when the reduction is 16 compared to 32, leading to a more significant number of channels and the introduction of more parameters to improve the model performance. Therefore, $r = 16$ is used for all experiments in this chapter.

4.4.4.4 Comparison with Other Attention Mechanisms

We compared the MAF proposed in this study with the mainstream attention mechanisms, including the SE, CA, and CBAM in Table 4.4. The upper part of the table represents the

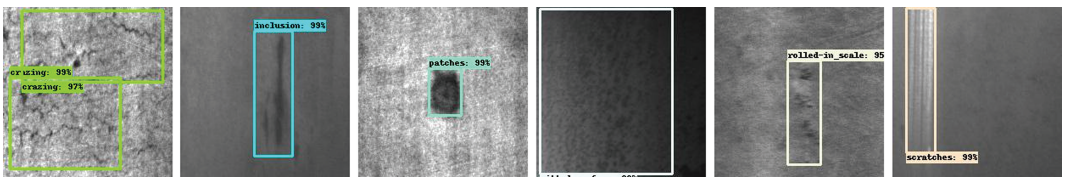


Figure 4.6 Examples of defect detection results on NEU-DET.

Table 4.2 Performance of the two module usage methods

baseline	mixForm	MAF	mAP
✓			75.21
		✓	$78.98_{+3.77}$
	✓		$80.49_{+5.28}$
	✓	✓	$82.91_{+7.70}$

Table 4.3 Comparison of the performance of MAF-based models under different *r*

Settings	*r*	mAP
baseline	—	75.21
+MAF(Ours)	16	**78.98** $_{+3.77}$
+MAF(Ours)	24	77.93 $_{+2.72}$
+MAF(Ours)	32	77.6 $_{+2.4}$
+MAF(Ours)*	16	**82.91** $_{+7.7}$
+MAF(Ours)*	24	81.84 $_{+6.63}$
+MAF(Ours)*	32	81.69 $_{+6.48}$

Table 4.4 Comparison of different attention mechanisms on the NEU-DET dataset before and after expansion

Settings	mAP
ResNet50-FPN	80.49
∼+SE	82.06 $_{+1.57}$
∼+ CBAM	80.78 $_{+0.29}$
∼+ CA	80.64 $_{+0.15}$
∼+ Ours	**82.91** $_{+2.42}$
ResNet50-FPN	75.21
∼+SE*	77.09 $_{+1.88}$
∼+ CBAM*	73.13
∼+ CA*	72.97
∼+ Ours*	**78.98** $_{+3.77}$

detection effect after expanding the training set, and the lower part is unexpanded. As seen in Table 4.4, the use of the SE attention mechanism has improved the detection performance of the model by 1.57 and 1.88, respectively. CA and CBAM contribute less to the model performance or even reduce the detection accuracy. However, when using the attention mechanism of this scheme, the defect detection model captures more semantic information than other attention mechanisms, and the best detection results are obtained experimentally. In addition, Figure 4.7 shows the loss variation using different attentions on the NEU-DET dataset.

4.4.4.5 The Performance of SFL

SFL changed the previous design of FL to upload the complete model and instead selected the common part of the defect detection model for aggregation. In the experiments, the shared one-layer SFL selected ResNet50's conv5_x layer for aggregation. The shared two layers selected the conv4_x and conv5_x layers for aggregation. Similarly, the shared three layers selected the conv3_x, conv4_x, and conv5_x layers for aggregation. In addition, the three sets of experiments mentioned earlier were compared with the shared backbone, shared full model, and Faster R-CNN using the original dataset of FL. The FL using the original dataset divided 1224 training images equally among 10 participants, and the other FL experiments all used mixForm augmented 6120 training images divided equally among 10 participants. The experiment allowed

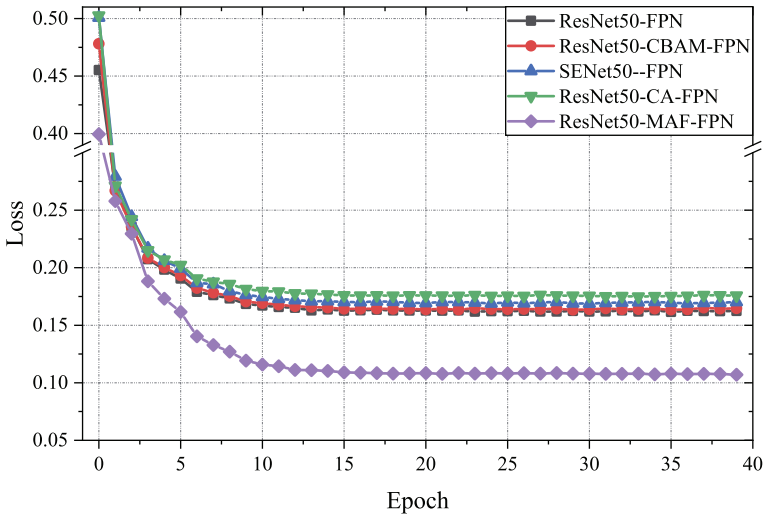


Figure 4.7 Loss changes during training on different attentions.

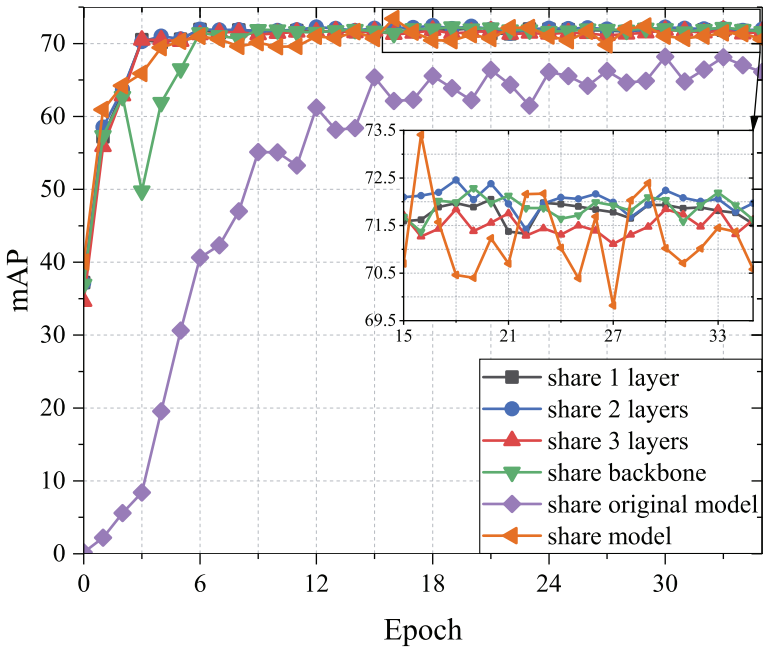


Figure 4.8 Defect detection in FL mAP.

the participants to share the model parameters with the server every three epochs. The rest of the settings were the same as in the previous chapter.

Figure 4.8 shows the average mAP of the 10 participants for every epoch. From the figure, it can be seen that SFL can achieve a similar mAP as when FL shares all model parameters. Meanwhile, SFL has a more stable training process compared to traditional FL. Compared with FL using the original dataset and Faster R-CNN, SFL demonstrates a clear advantage of faster convergence and effective improvement of defect detection accuracy on FL. The shared

Table 4.5 Comparison of the number of uploaded parameters for different sharing methods

Aggregate	One layer	Two layers	Three layers	Backbone	Model
Params	33M	49M	51M	52M	70M
ration	47.02%	69.98%	73.86%	74.58%	100%

two-layer SFL improves 6.35 mAP over the FL using the original dataset and Faster R-CNN. Table 4.5 compares the number of uploaded parameters for different sharing methods. Sharing only part of the model reduces the number of uploaded parameters to the server, significantly reducing the communication overhead and speeding up the aggregation of the global model.

4.5 Conclusion

To satisfy the requirements of industrial quality inspection, this chapter proposes a scheme for surface defect detection. This scheme effectively improves the accuracy and achieves collaborative training of the defect detection model. Experimental results show that this chapter's scheme outperforms previous schemes. The model achieves 78.98 mAP on the original NEU-DET dataset and 82.91 mAP after mixForm. In addition, SFL significantly saves computational resources and improves the generalizability of the model. In the future, we will continue to study the defect scale problem and design an effective feature fusion module to further improve the defect detection model.

References

[1] Illingworth, J., and Kittler, J., A survey of the Hough transform, *Computer Vision Graphics and Image Processing*, 44(1), 87–116 (1988).
[2] Pathak, R.S. The wavelet transform, *Springer Science & Business Media*, 4(1), 4–76 (2009).
[3] Nussbaumer, H.J. *The fast Fourier transform*, Springer, Berlin, 80–111 (1981).
[4] Kim, S., Kim, W., Noh, Y.K., Park, F.C., Transfer learning for automated optical inspection, *Proceedings of the IEEE* 23, 25172524 (2017).
[5] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., You only look once: Unified, real-time object detection, *Proceedings of the CVPR* 33, 779–788 (2016).
[6] Lin, T.Y., Goyal, P., Girshick, R., He, K., and Dollár, P., Focal loss for dense object detection, *Proceedings of the ICCV* 16, 2980–2988 (2017).
[7] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., and Berg, A. C., Ssd: Single shot multibox detector, *Proceedings of the ICCV* 14, 21–37 (2016).
[8] Ren, S., He, K., Girshick, R., and Sun, J., Faster r-cnn: Towards real-time object detection with region proposal networks, *Proceedings of the NeurIPS* 29, 28–39 (2015).
[9] Cai, Z., and Vasconcelos, N., Cascade r-cnn: Delving into high quality object detection, *Proceedings of the CVPR* 35, 6154–6162 (2018).
[10] Zhang, H., Cisse, M., Dauphin, Y.N., and Lopez-Paz, D., Mixup: Beyond empirical risk minimization, arXiv, 1710.09412, (2017).
[11] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y., Cutmix: Regularization strategy to train strong classifiers with localizable features, *Proceedings of the CVPR* 36, 6023–6032 (2019).

[12] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B.A., Communication-efficient learning of deep networks from decentralized data, *Proceedings of the AISTATS* 20, 1273–1282 (2017).
[13] Hu, J., Shen, L., and Sun, G., Squeeze-and-excitation networks, *Proceedings of the CVPR* 35, 7132–7141 (2018).
[14] Woo, S., Park, J., Lee, J.Y., and Kweon, I.S., Cbam: Convolutional block attention module, *Proceedings of the ECCV* 15, 3–19 (2018).
[15] Hou, Q., Zhou, D., and Feng, J., Coordinate attention for efficient mobile network design, *Proceedings of the CVPR* 38, 13713–13722 (2021).
[16] Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z., and Shao, L., Surface defect detection methods for industrial products: A review, *Applied Sciences*, 11(16), 7657–7671 (2021).
[17] Prasitmeeboon, P., and Yau, H., Defect detection of particleboards by visual analysis and machine learning, *Proceedings of the ICEAST* 5, 1–4 (2019).
[18] Wang, J., Fu, P., and Gao, R. X., Machine vision intelligence for product defect inspection based on deep learning and Hough transform, *Manufacturing Systems*, 51(1), 52–60 (2019).
[19] Jin, X., Wang, Y., Zhang, H., Zhong, H., Liu, L., Wu, Q. J., and Yang, Y., DM-RIS: Deep multimodel rail inspection system with improved MRF-GMM and CNN, *IEEE Transactions*, 69(4), 1051–1065 (2019).
[20] Huang, Z., Xiao, H., Zhang, R., Wang, H., Zhang, C., and Shi, X., Multi-scale feature pair based R-CNN method for defect detection, *Proceedings of the IEEE* 12, 46–51 (2019).
[21] Liu, Z., Liu, K., Zhong, J., Han, Z., and Zhang, W., A high-precision positioning approach for catenary support components with multiscale difference, *IEEE Transactions*, 69(3), 700–711 (2019).
[22] Zhang, C., Chang, C.C., and Jamshidi, M., Concrete bridge surface damage detection using a single-stage detector, *Computer-Aided Civil and Infrastructure Engineering*, 35(4), 389–409 (2020).
[23] Cheng, X., and Yu, J., RetinaNet with difference channel attention and adaptively spatial feature fusion for steel surface defect detection, *IEEE Transactions*, 70(1), 1–11 (2020).
[24] He, K., Zhang, X., Ren, S., and Sun, J., Deep residual learning for image recognition, *Proceedings of the CVPR* 33, 770–778 (2016).
[25] Shorten, C., and Khoshgoftaar, T.M., A survey on image data augmentation for deep learning, *Journal of Big Data*, 6(1), 1–48 (2019).
[26] Pan, S. J., and Yang, Q., A survey on transfer learning, *IEEE Transactions*, 22(10), 1345–1359 (2009).
[27] Song, K., and Yan, Y., A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects, *Applied Surface Science*, 285(1), 858–864 (2013).

Chapter 5

A Compact Object Detection Architecture with Transformer Enhancing

Liyuan Cui[1], Guoqiang Zhong[1], Xiang Liu,[1,2] and Hongwei Xu[3]

[1] College of Computer Science and Technology, Ocean University of China, Qingdao, China

[2] Innovation Center, Ocean University of China, Qingdao, China

[3] School of Control Science and Engineering, Shandong University, Jinan, China

Abstract

Transformer architecture has become the de facto standard as its advancements increase in computer vision. While the application is restricted because of its unprecedented storage, heavy reliance on data size, and intolerable computational power consumption, the performance loss is permitted by compressing the network to gain a tiny and faster model. In this chapter, we leverage the latest advances in transformer in computer vision to present an architecture enhanced by the right-size vision transformer, to undertake the backbone in the object detection network. We start with the process of slicing the design to reduce the computational burden, and the feature is delivered to GhostNet. Its proper component is replaced by the vision Transformer module. We compare with the baseline Yolov5 and show that our approach performs on par with the Transformer module using two medium-scale object detection datasets with three scales of networks. Without relying on ultra-large datasets and pre-trained models, our model presents comparable or even higher mAP metrics with only half of the model size and floating-point computation compared with the baseline Yolov5.

Key Words: Vision Transformer encoder, Lightweight network, Multi-head attention, Yolov5, GhostNet.

5.1 Introduction

Since AlexNet [1] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, and 10.9% performance transcended the second place, deep convolutional neural networks (CNNs) have rapidly garnered tremendous attention and deeper investigation. With the advancement of CNN models increasing, e.g., VGGNet [2], GoogleNet [3], and ResNet [4], the application is explored to serve different fields of computer vision tasks, including object tracking [5], object detection [6, 7], image compression [8], and instance segmentation [9]. With respect to modern computer vision, CNNs have shown unparalleled potential and proven to be a very successful paradigm. Although CNN is particularly impressive in extracting local

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

information, it is invented on pixel matrices in image processing, only considering sequential n-grams that are consecutive on the surface string and neglecting long-distance dependencies. It does not gain further achievements in a computational and parametric-efficient manner besides deepening the networks.

Transformer is a type of neural network mainly based on the self-attention mechanism [10], which can provide relationships between different features. Attention mechanism allows the modeling of dependencies regardless of their distance in the input or output sequences. It is widely used in the field of natural language processing (NLP), e.g., the famous BERT [11] and GPT-3 [12] models. The whole community is inspired by the power of these Transformer models to investigate their use for visual tasks such as DERT [12] and ViT [13]. However, it is typically not enough to train a network with good statistical prediction unless it can sufficiently be pre-trained on data with similar attributes.

We consider here to help it deliver super performance while releasing it from large-scale datasets. We find that the information across the entire image is integrated by attention even in the lowest layers, and the average distance, where the information is integrated between image space across, is parallel to the receptive field size in CNNs. This suggests that the highly localized attention may play a similar function as early convolutional layers in CNNs.

This chapter proposes a new lightweight network with rare performance loss. The entire architecture is shown in Figure 5.1. An efficient way is proposed to further improve the performance of the object detection model through a combination of Transformer module and CNNs. It establishes the balance between accuracy with computational consumption and real-time memory occupancy during training. The last module of GhostNet [14] is replaced by our Transformer module, which acts as the backbone of the network. Before entering the backbone,

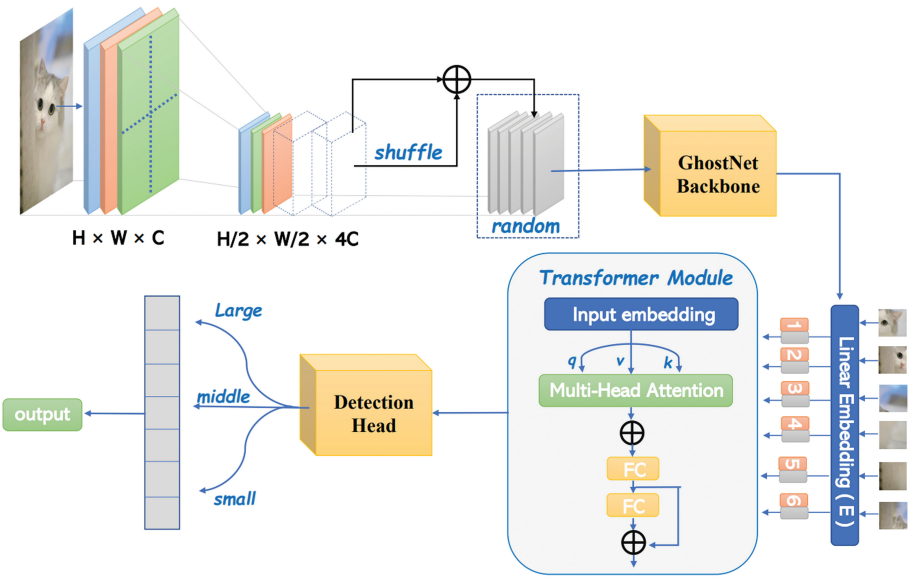


Figure 5.1 The overall structure of the proposed network. The GhostNet backbone receives ground truth after it is sliced. The Transformer module takes place of the original last module in GhostNet. The dominant component named multi-head attention is shown in detail in Figure 2.1. FC represents the fully connected layer and the output of the detection head is a vector in $n + 5$ and n is the number of identifiable objects.

the pixels of the image $x \in \mathbb{R}^{C \times H_0 \times W_0}$ are reconstructed into low-resolution images, aggregating the information of *H, W* dimension into the *C* space to attain the improvement of the perceptual field, which has a similar quality of information to the original one. Then the random images are fed into the backbone after being shuffled. At the bottom of the network, CNN forces the capture of local structure using local receptive fields, shared weights, and spatial subsampling, and thus also achieves highly localized features with less effort compared with Transformer. Further, the perspective fields are explored with network depth, which is beneficial for the Transformer module to integrate information globally.

First, a standard Transformer is utilized to process the sequence of patches corrupted by the local information of a patch. Given a 2D image $x \in \mathbb{R}^{C \times H \times W}$, we uniformly split it into N patches $x_p \in \mathbb{R}^{N \times P^2 C}$ [13], where the (P, P) is the resolution of each image patch, and $N = HW / P^2$, which serves as the effective input sequence length for the Transformer. Then, the projected tokens are flattened into 1D for subsequent process with a trainable linear projection map (Eq. 5.1),

$$I = [x_{class}; x_p^1 E^1; ...; x_p^n E^n] + E^{pos}, \tag{5.1}$$

where $E^i \in \mathbb{R}^{(P^2 \dot{C}) \times D}, i \in N$, is the mapping matrix, and $E^{pos} \in \mathbb{R}^{(N+1) \times D}$ is obtained from the linear operation as position embedding. A learnable embedding is prepended in the sequence of embedded patches $z_0 = x_{class}$. It contains latent information, and more information about the sequence is accumulated through self-attention, which is later used for classification.

The main contributions of this chapter are as follows:

1. Extending Transformer-based architecture to lightweight network for objection detection, releasing it from huge datasets.
2. Introducing a combination of Transformer and CNN based on their different internal representations when processing the image data. The experiments demonstrate that this approach improves the performance and provides network flexibility with less real-time memory and storage.

The rest of the chapter is composed of the following parts. Section 5.2 includes a review of the network lightweight and vision Transformer. Section 5.3 describes details of the proposed Transformer module and the other auxiliary lightweight work. Section 5.4 is devoted to the experiments and results to analyze the effectiveness and validity of the Transformer module. Section 5.5 includes the conclusions of this work and open research lines.

5.2 Related Works

In this section, we review some existing literature on deep network compression and vision transformer separately.

5.2.1 Deep Network Compression

Neural networks are intensive in terms of both computation and memory. Therefore, deep network compression is introduced to address this limitation. It includes heuristic methods [15], reinforcement learning methods [16], and genetic and evolutionary algorithms [17]. Knowledge

distillation [11, 18, 19] refines the knowledge in a complex machine learning model or an ensemble of models into a smaller single model that is much easier to deploy without significant loss in performance. Another approach is approximating a matrix by one whose rank is less than that of the original matrix [20]. The goal of this approach is to obtain more compact representations of the data with limited loss of information. In addition, neural structure design [21–23] has great potential for building compact deep networks.

However, the neglect of abundant, even redundant, information about these methods in the feature maps loses a comprehensive understanding of the input data. In that case, GhostNet [14] uses cheap linear transformations to generate many ghost that could fully utilize information. Therefore, we use GhostNet as our architectural bias.

5.2.2 Vision Transformer

The major success of the Transformer architectures in the field of NLP has inspired researchers to apply Transformer to computer vision tasks. In classification, Chen et al. trained a sequence Transformer to achieve auto-regressively to predict pixels [24] on image classification tasks. A model named ViT [13] was applied on sequences of image patches using a pure Transformer. Touvron et al. proposed a competitive convolution-free Transformer [25] called a data-efficient image Transformer through the help of a teacher–student strategy. Besides image classification, Transformer models have been applied to address other vision tasks, including object detection [12], video understanding [26], image processing [27], and semantic segmentation [28]. Additionally, Zhu et al. [29] extended the network slimming approach for reducing the dimensions of linear projections in both FFN and attention modules. To sum up, Transformer-based models attract more and more researchers to improve a wide range of visual tasks. However, they need the quadratic complexity resulted by the number of tokens, and they are not computationally efficient.

5.3 Approach

This section introduces our proposed model in detail, including the pre-process layer and the Transformer module, as well as its entire structure.

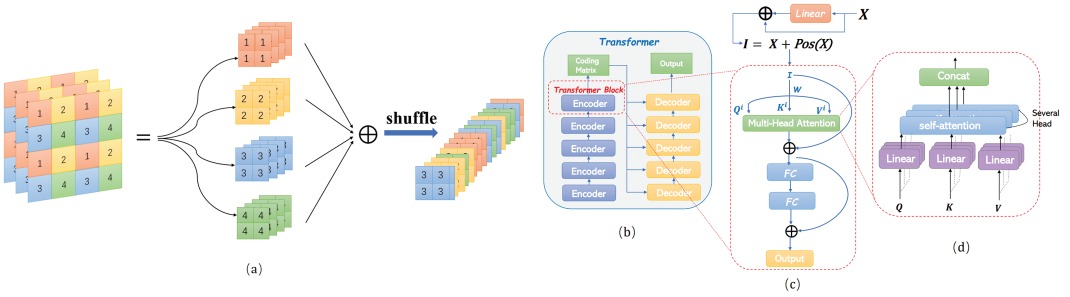


Figure 5.2 Slicing operation and the Transformer module: (a) Slicing projection is performed by drawing the pixels of the image at intervals circumferentially and concatenating it for channel expansion. (b) The structure of the Transformer in blocks. (c) A particular demonstration of the Transformer module. (d) The details of multi-head attention.

5.3.1 Pre-Process Layers

An approach is proposed to enhance the network computing speed. The pixel points of the high-resolution image are periodically drawn out to be reconstructed using the low-resolution image. For example, we start by stacking the four adjacent positions of a pixel to reduce the information from the W, H dimensions to the C dimension. It achieves a sharp decrease in the amount of computation for speeding up. Figure 2.1(a) illustrates this operation. Input images are obtained for every few pixels after being sliced (the default here is 2) to concat. This similar operation called Focus is used in previous You Only Look Once (YOLO). Three layers of convolution for downsampling from CNNs are replaced. Additionally, Focus is no longer used in the current YOLO version.

$$\begin{aligned} params &= K_h \times K_w \times C_{in} \times C_{out}, \\ FLOPs &= K_h \times K_w \times C_{in} \times C_{out} \times H \times W. \end{aligned} \tag{5.2}$$

This Focus is used to transform the image of $x \in \mathbb{R}^{C \times H_0 \times W_0}$ to $x \in \mathbb{R}^{4C \times \frac{H_0}{2} \times \frac{W_0}{2}}$. According to Equation (5.2), there is a significant reduction in computation and the number of parameters without information loss.

5.3.2 Vision Transformer Module

Two main parts are contained in our Transformer module, as shown in Figure 2.1c: multi-head self-attention (MSA) module and multi-layer fully connected (MLF) module.

After tokenization, we model the interactions between these visual tokens. In the self-attention module, the inputs $I \in \mathbb{R}^{N \times D}$ obtained in Equation 5.1 are linearly transformed to, i.e., queries $Q \in \mathbb{R}^{N \times D_k}$ and keys $K \in \mathbb{R}^{N \times D_k}$, where N is the sequence length and D and D_k are the dimensions of inputs. The scaled dot-product attention is applied to Q, K as

$$A_p = QK^T = W_q I \odot I^T W_k{}^T. \tag{5.3}$$

where W_q and W_k are the weight matrices corresponding to queries and keys of sequence, respectively. We accumulate A_p using the values $V \in \mathbb{R}^{N \times D_v}$ with its weight matrix. Finally, a linear layer named MAS is used to produce the output after the normalization with the softmax function:

$$M_p = MSA(I) = Softmax\left(\frac{A_p}{\sqrt{D}}\right) I W_v \dot{W}_o. \tag{5.4}$$

The MLF constructed with two pointwise convolutions is applied after multi-head self-attention block for feature transformation. It is parameterized by $w^1 \in \mathbb{R}^{D \times n}$, $b^1 \in \mathbb{R}^n$ and $w^2 \in \mathbb{R}^{D \times n}$, $b^2 \in \mathbb{R}^n$, where n equals the number of neurons in the middle layer of the MLF:

$$MLF\left(\sum_{i=1} M_p\right) = w^2\left(w^1 \sum_{i=1} M_p + b^1\right) + b^2. \tag{5.5}$$

We combine the output of the Transformer module with the feature map to refine its pixel–array representation as

$$O = MLF\left(\sum_{i=1} MSA(I)\right) \oplus \sum_{i=1} MSA(I). \tag{5.6}$$

$$O_1 = LN(O + FeedForwardNetwork(O)). \tag{5.7}$$

$$Module(I) = O_1. \tag{5.8}$$

CNNs neglects long-distance dependencies as they only consider sequential n-grams that are consecutive on the surface. In this way, the model is more efficient to extract high quality information with rare computation expense increase compared with using GhostNet alone.

5.3.3 The Entire Backbone Architecture

We abstract a compact form of Transformer to offset the lack of distance dependencies, which plays an important role in accessing complete information and strengthening the network. CNN is essential for its local inductive bias, and less time and data are required with a lower number of parameters to accurately fit the data. After combining both of the above in a proper position, an architecture with high-performance and cost-effective computation is designed.

5.4 Experiment

5.4.1 Dataset

Two medium-sized datasets are used for verifying the enhancement effect of the detection network: one is the vessel detection dataset named *ships* in various scales and resolutions. There are approximately 9800 images, with 8833 for training and 967 for testing.

The other dataset is named *drinks*. It is used in unmanned detection scenarios and is composed of indoor beverage pickup and vending machines. It has 6,478 images, including 5,821 images in the train set and 657 in the test set. It has unified scale and size with seven classes, representing different colors and brands of beverages, such as Coca-Cola, Pepsi, Sprite, Fanta, Vitsoy, Mizone, and Coca-Cola Bottle.

5.4.2 Data Enhancement

A range of operations is included by data enhancement, such as scale, horizontal or vertical flip, shear, rotate, transparency change, object copy–paste, Hue, Saturation, Value (HSV) change, and mixup. A fraction of samples in the total dataset is selected to investigate the optimal hyper-parameters through a genetic algorithm in the basic processing of images. Each is treated as a chromosome, which represents the parameters being optimized, and each integer within the chromosome is known as a gene. An initial population of chromosomes is generated at random, and the smallest cost values are chosen as the best and are then subjected

to operations involving reproduction, crossover, and mutation to capture the adaptation of hyper-parameters for CNNs training in different combinations. Finally, we obtain the best hyper-parameters.

Table 5.1 illustrates the effectiveness of the data enhancement to the dataset. It delivers a higher accuracy (mAP) with an improvement of almost **30%** through just training the enhanced input images. Therefore, we apply the data enhancement to both the *ships* and *drinks* datasets before feeding them to the object detection network.

5.4.3 Baseline

We use YOLOv5 as our baseline as it effectively predicts the locations and class of multiple boxes immediately and is able to model fine structures using end-to-end object detection and recognition method. Without producing the extracting proposals and the sliding window, the whole graph is used by YOLO to train the model. It is able to distinguish the target of the input data with superior speed enhanced by the time saved from generating proposals.

There are mainly three components to make up the YOLO model:

- Backbone is mainly used for the key extraction from ground truth.
- Neck is the dominant part for generating feature pyramids.
- Head uses anchor boxes to compose the vector output of the network with class probability and the coordinate bounding box scores. It mainly appears in the detection period.

5.4.4 Comparison

Table 5.2 delivers the results based on the comparison combining with parameters, model storage, GLOPS, and mAP.5 between the baseline and proposed architecture. The mAP@[.5,.95] demonstrates the confidence level of the category representing the accuracy. In the case that the parameters and calculation consumption reduce by about 50%, we achieve comparable accuracy. We get a low-storage model to speed up its reasoning and facilitate deployment. The high performance is embodied in the multi-scale and multi-resolution input images to show the

Table 5.1 Comparison results (%) between the original input images and enhanced images on *ships*. There are three scales of networks: $baseline_l$, $baseline_m$, and $baseline_s$. The data enhancement is applied to the baseline $drBaseline_{scale}$ for achieving higher performance.

Backbone	Params	GFLOPS	Storage	runtimeGPU	mAP@.5	mAP@[.5,.95]
baseline_l	20.9m	48.1	42.2M	11.2G	65.4	45.1
baseline_m	7.0m	15.9	14.4 M	6.01G	62.1	36.9
baseline_s	1.8m	4.2	3.8M	2.94G	59.5	36.4
drBaseline_l	20.9m	48.1	42.2M	11.2G	**80.2**	**58.4**
drBaseline_m	7.0m	15.9	14.4 M	6.01G	**78.6**	**56.0**
drBaseline_s	1.8m	4.2	3.8M	2.94G	**76.9**	**53.4**

Table 5.2 Experimental results (%) compared between baseline YOLOv5 and our model on the ships dataset. *Large*, *middle* and *small* are the scales of the network

Backbone	Params	GFLOPS	Storage	runtimeGPU	mAP@.5	mAP@[.5,.95]
baseline_l	20.9m	48.1	42.2M	11.2G	80.2	58.4
baseline_m	7.0m	15.9	14.4 M	6.01G	78.6	56.0
baseline_s	1.8m	4.2	3.8M	2.94G	76.9	53.4
ours_l	11.2m ↓	19.6 ↓	22.9M ↓	10.2G ↓	79.5	58.3
ours_m	4.3m ↓	8.4 ↓	9.0M ↓	5.78G ↓	78.4	**56.8**
ours_s	1.1m ↓	2.4 ↓	2.5M ↓	3.36G	76.6	53.1

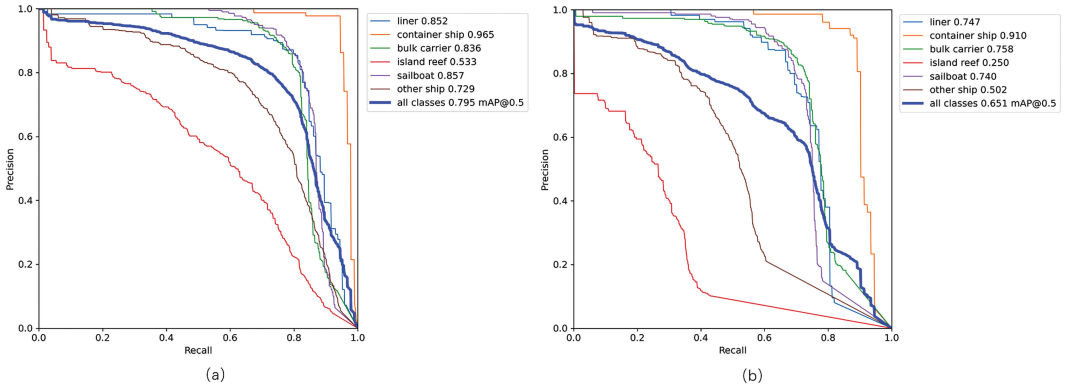


Figure 5.3 (a) The PR curve of the proposed model; (b) the PR curve of the baseline. They are both trained on the *ships* dataset on a large scale.

powerful capability for the extraction of deep feature. The PR curve shown in Figure 5.3 gives a better visualization of the performance improvement of the Transformer module.

Table 5.3 illustrates the results obtained on the *drinks* dataset compared with Table 5.2. The whole architecture doubles the speed (GFLOPS) and halves the number of parameters (params) on the multi-category datasets, with an improvement in accuracy. In the task of classification, the Transformer module presents impressive properties.

5.4.5 Ablation Experiments

Finally, we examine the function and effect of each module with ablation experiments. The results are shown in Table 5.4. There are mainly three components to be checked out in the following:

- noFocus: Replacing the Focus module with down sampling.
- noTR: There is no substitution of the last block in GhostNet instead of using the Transformer module.
- neither: Combination of the above two operations.

Table 5.3 Experimental results (%) obtained by our model and the YOLO baseline on the drinks dataset

Backbone	Params	GFLOPS	Storage	runtimeGPU	mAP@.5	mAP@[.5,.95]
baseline_l	20.9m	48.1	41.2M	10.2G	97.8	68.7
baseline_m	7.0m	15.9	14.0M	6.5G	97.9	66.5
baseline_s	1.8m	4.2	3.7M	4.3G	97.6	63.8
ours_l	11.2m ↓	19.6 ↓	22.3M ↓	7.77G ↓	**98.1**	67.5
ours_m	4.3m ↓	8.5 ↓	8.8M ↓	4.36G ↓	97.4	**66.8**
ours_s	1.1m ↓	2.4 ↓	2.5M ↓	2.41G ↓	97.1	**64.8**

Table 5.4 Experimental results (%) obtained by ablation experiments on the ships dataset to demonstrate further the effectiveness of the method proposed in this chapter

Methods	Params	GFLOPS	Storage	runtimeGPU	mAP@.5	mAP@[.5,.95]
noFocus_l	11.2m	19.6	84.9M	10.7G	79.6	57.9
noFocus_m	4.3m	8.4	33.6M	6.1G	78.2	55.6
noFocus_s	1.1m	2.4	9.1M	2.5G	75.3	52.4
noTR_l	8.5m	18.4	17.2M	10.2G	77.6	55.5
noTR_m	3.7m	8.2	7.6M	6.5G	75.8	52.6
noTR_s	0.9m	2.3	2.2M	4.3G	75.9	53.5
neither_l	8.5m	18.4	67.8M	11.2G	79.3	57.8
neither_m	3.7m	8.2	29.6M	6.7G	78.4	56.2
neither_s	0.9m	2.3	8.1M	4.5	75.8	52.9
ours_l	11.2m	19.6	22.9M	10.2G	**79.5**	**58.3**
ours_m	4.3m	8.4	9.0M	5.78G	**78.4**	**56.8**
ours_s	1.1m	2.4	2.5M	3.36G	**76.6**	**53.1**

With respect to the experience *noFocus*, it is obvious that the Focus operation lightweights the image processing (Storage). At the same time, the Transformer module achieves improvement and even realizes better detection capabilities, increasing mAP by almost 3% on both middle and large-scale networks. The loss of our network is shown in Figure 5.4.

5.5 Conclusion

In this chapter, a Transformer module is introduced to enhance the performance of a lightweight network for objection detection. We are inspired by the low relational inductive bias of convolution as it is adept at vision-based problems due to their invariance to spatial translations as well as the mechanism of attention-head to process images. Attention-head works efficiently

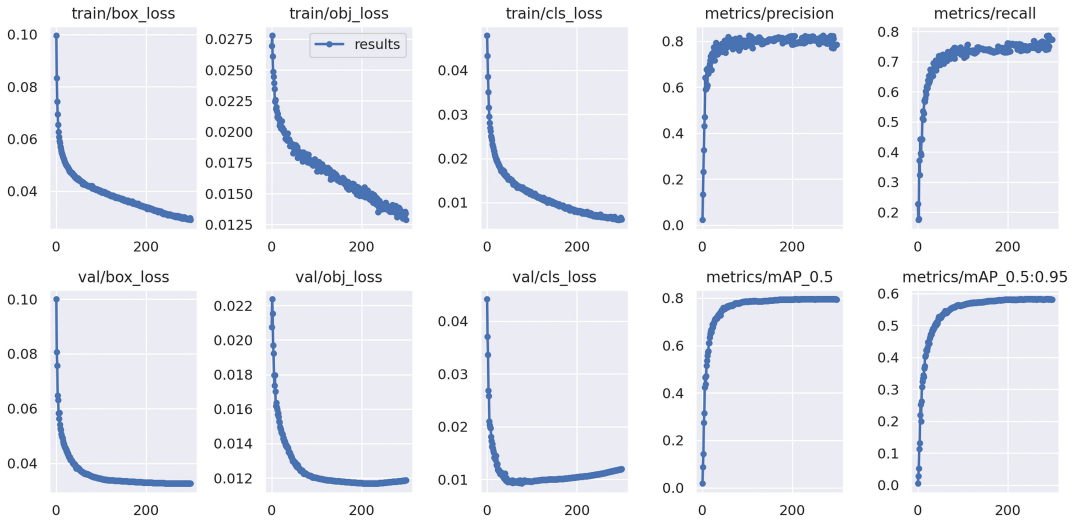


Figure 5.4 The loss of the proposed model is trained on large scale based on the *ships* dataset.

only at the top layer, while the effect is similar to that of convolution at the bottom. Obviously, the convolution has a better performance. We design architecture by properly combining the Transformer and CNN. Compared with baseline YOLOv5, we use GhostNet as a backbone, and its last module is substituted with the Transformer module for both lightweight and high performance. It is demonstrated as a computationally efficient objection network through experiments.

In our experiments, the Transformer module is verified to achieve good results, regardless of the scale, resolution of the image, and target category. The increase in accuracy and decrease in consumption is mainly contributed by the Transformer module through ablation experiments.

Recently, Dai et al. proposed CoAtNet [30], combining CNN and Transformer by stacking the two disparate blocks in a novel way. Hassani et al. introduced the ViT-Lit [31] using the inductive biases of CNN to free the Transformer from depending on large datasets for training. In future work, we are aiming to better integrate the transformer module in object detection theoretically for an integrated structure with improvements in their performance.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 79, 2015, Conference Track Proceedings*, 2015.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015*. IEEE Computer Society, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[5] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[6] Chienyao Wang, Alexey Bochkovskiy, and Hongyuan Mark Liao. Scaledyolov4: Scaling cross stage partial network. In *Proceedings of the IEEECVF Conference on Computer Vision and Pattern Recognition*, pages 13029–13038, 2021.

[7] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[8] Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Hao Li, and Rong Jin. Learning accurate entropy model with global reference for image compression. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 37, 2021*. OpenReview.net, 2021.

[9] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panopticdeeplab: A simple, strong, and fast baseline for bottomup panoptic segmentation. In *Proceedings of the IEEECVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 49, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[11] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[12] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision ECCV 2020 16th European Conference, Glasgow, UK, August 2328, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2020.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 37, 2021*. OpenReview.net, 2021.

[14] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *2020 IEEECVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 1319, 2020*, pages 1577–1586. Computer Vision Foundation IEEE, 2020.

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22–29, 2017*, pages 1398–1406. IEEE Computer Society, 2017.

[16] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2N learning network to network compression via policy gradient reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[17] Shaohui Lin, Rongrong Ji, Yuchao Li, Yong Wu jian, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 1319, 2018, Stockholm, Sweden*, pages 2425–2432. ijcai.org, 2018.

[18] Geoffrey E Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs1503.02531, 2015.

[19] Yihkai Lin, Chufu Wang, Chingyu Chang, and Haolun Sun. An efficient framework for counting pedestrians crossing a line using low-cost devices: The benefits of distilling the knowledge in a neural network. *Multim. Tools Appl.*, 80(3):4037–4051, 2021.

[20] Bin Sun, Jun Li, Ming Shao, and Yun Fu. Lprnet Lightweight deep network by lowrank pointwise residual convolution. *CoRR*, abs1910.11853, 2019.

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets efficient convolutional neural networks for mobile vision applications. *CoRR*, abs1704.04861, 2017.

[22] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6848–6856. Computer Vision Foundatio IEEE Computer Society, 2018.

[23] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, Song Han, William J Dally, and Kurt Keutzer. Squeezenet alexnetlevel accuracy with 50x fewer parameters and textless1mb model size. *CoRR*, abs1602.07360, 2016.

[24] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 1318 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 2020.

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*, volume 139, pages 10347–10357. PMLR, 2021.

[26] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. Volume, abs/1804.00819, 2018.

[27] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. volume abs/2012.00364, 2020.

[28] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H S Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6877–6886, 2021.

[29] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *CoRR*, abs/2104.08500, 2021.

[30] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet Marrying convolution and attention for all data sizes. *CoRR*, abs2106.04803, 2021.

[31] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *CoRR*, abs2104.05704, 2021.

Chapter 6

Dual-Path Mutual Attention Network for Medical Image Classification

Yixuan Lu, Guoqiang Zhong* and Yajing An
Yajing An College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract

Medical image classification plays a critical role in the computer-aided diagnosis. However, medical images are difficult to obtain due to the privacy issue. On the one hand, there are few available large-scale datasets for training deep learning models. Moreover, the intra-class variation and inter-class similarity in medical images are severe. In this chapter, we propose a dual-path architecture called dual-path mutual attention network (DPMAN) to address the above issues. We construct a dual-path architecture with a mutual learning strategy, which can simultaneously learn the prediction information of the two sub-networks. Additionally, to obtain more information on intra-class and inter-class, we propose the aggregation attention module to concentrate the information learned by the network. To solve the problem of class imbalance, we use the weighted cross-entropy loss to make the network pay more attention to class with fewer data. We do experiments on the HAM10000, APTOS, and Brain Tumor datasets. The obtained results on these three datasets show that our method has achieved higher accuracy than the state-of-the-art approaches, demonstrating the effectiveness of DPMAN.

Key Words: Deep learning, Attention mechanism, Medical image segmentation

6.1 Introduction

Deep learning is playing an increasingly important role in the field of medicine. In traditional medical diagnosis, the experience of experts is almost a decisive factor. However, there are very few experts with rich experience, and under high-intensity repetitive working conditions, doctors are prone to making mistakes due to fatigue, and it may take a long time to deal with complex images. With the rapid development of deep learning and artificial intelligence, more and more attempts have been made to realize certain medical diagnoses through deep learning methods [1–3]. However, deep learning usually requires a lot of data for training in order to achieve good results; medical images are difficult to obtain due to privacy issues in the medical

* E-mail address: gqzhong@ouc.edu.cn

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

domain [4], and the labeling of medical images requires experienced professionals so that the provided datasets are generally not large enough to sufficiently train a deep learning model [5].

To solve the problem of over-fitting caused by the lack of medical image data, Pan and Yang [6] propose transfer learning. Transfer learning is to transfer the learned model parameters to a new model for new model training. Using rich training data to help target data learning can improve task performance with limited data. However, due to the large domain mismatch between the usual natural images and the medical images, this type of transfer is difficult to succeed in the medical domain.

Hinton et al. [7] propose knowledge distillation, which has been an effective and widely used technique in recent years. The output of the pre-trained complex model (teacher model) is used as a supervision signal to train another simple network (student model). Knowledge distillation is generally on the same dataset so that the two networks are forced to be as close as possible in terms of output. It is a means to transfer the information learned from a complex network to a small network and to improve the effect of the student network. Nevertheless, the complex teacher network requires lots of computation resources. To address this problem, Zhang et al. [8] proposed mutual learning (ML), which directly discards the teacher network and allows student networks to learn from each other to improve classification accuracy. ML distils the unpretrained networks to improve their performance. In this way, it is possible to obtain a compact network with better performance than a network extracted from a strong and static teacher network. In this chapter, we propose an ML strategy that is suitable for medical image classification.

In addition to the scarcity of data, medical images also have higher intra-class variation and inter-class similarity than natural images. Specifically, two diseases may show extremely similar colors, shapes, and textures, making them more difficult to distinguish than natural images. To deal with the above problems, we propose a dual-path architecture, which can learn the prediction information and complementary inter-class knowledge of the two networks at the same time. To obtain more accurate intra-class and inter-class knowledge learned by the network, we propose the aggregation attention module (AAM) to concentrate the information learned by the network and aggregate the features from two networks. In addition, medical image datasets usually have a serious class imbalance because some diseases are common and others are rare. Directly transferring the conventional classification methods from computer vision to medical image classification, the extracted knowledge may be biased toward the majority class, leading to insufficient representation of minority classes, resulting in poor performance. Therefore, we use weighted cross-entropy loss to strengthen the networks attention to minority classes, thereby reducing the impact of class imbalance.

In summary, we propose a dual-path mutual attention network (DPMAN), which obtains the specific features of each category through ML between the two networks and concentrates the intra-class features through the AAM. At the same time, to solve the problem of class imbalance, we use weighted cross-entropy loss to add feature representation for a few categories. Our method achieves higher accuracy on the used datasets.

Our contributions can be summarized as follows:

(1) We propose a new dual-path architecture to solve the medical image classification problem.
(2) We propose the AAM, which can jointly learn intra-class and inter-class features by aggregating features from two networks and weighting attention for each network to selectively emphasize their characteristics. Meanwhile, we propose an ML strategy that can achieve good results without distilling the complex teacher network in learning.
(3) Finally, our method achieves a better performance on three used datasets, and its effectiveness is demonstrated through extensive experiments.

6.2 Related Work

In this section, we review some work related to the DPMAN model, including medical image classification, transfer learning, and attention mechanism.

6.2.1 Medical Image Classification

Medical image diagnosis usually requires careful judgment by experienced doctors to draw conclusions, especially for intractable diseases such as cancer, which even requires multiple doctors to diagnose at the same time to get the most reliable conclusion. Before the emergence of deep learning, manually defined image features (e.g., texture, shape of the image, and gray histogram of the image) were often used and then classified based on machine learning models (e.g., support vector machine, logistic regression, and random forest) after feature selection. The typical representative is the imaging histology method, which has achieved many important results in the stage of tumor typing and prognosis prediction of treatment [9]. However, the reliability and robustness of classification are largely affected by the manual definition of features and the method of feature selection.

Recently, leveraging deep learning and computer vision-based techniques to automate medical imaging diagnosis at the point of care in a robust manner has been widely studied [10–12], which can obtain more accurate and objective clinical assessments, as well as improve screening efficiency and quality control [13]. Esteva et al. [14] demonstrated dermatologist-level accuracy in detecting melanoma. Zhu et al. [15] presented a fully automated lung computed tomography (CT) cancer diagnosis system, DeepLung. DeepLung consists of two components: nodule detection (identifying the locations of candidate nodules) and classification (classifying candidate nodules into benign or malignant).

The above networks have achieved good results, but the training of deep learning models requires a lot of data, and medical imaging data are difficult to collect due to labeling difficulties and privacy regulations.

6.2.2 Transfer Learning

The large scale of deep learning models is in sharp contrast with the scarcity of medical imaging diagnostic training data, which makes transfer learning a necessary condition for this task [16, 17]. By learning the knowledge obtained in the source task, instead of using the randomly initialized weight training model from the beginning, which can achieve a better training effect when the new task has less training data. Transfer learning not only improves accuracy but also reduces training time [18, 19]. Kerman et al. [20], Ayang et al. [21], and Gu et al. [22] proposed a transfer learning method for medical image classification, which achieved good results with only a small amount of data. However, the characteristics of natural images and medical images are different, and this transfer method is difficult to convince the medical domain.

Knowledge distillation [7] is an effective and widely used technology to transfer knowledge from the teacher network to the student network on the same dataset so that the two networks are as close as possible in terms of output. The information learned by the teacher network is transferred to a student network and hopes to maintain a better effect or improve the student network's effect. Qin et al. [23] proposed the first systematic knowledge distillation framework

for medical image segmentation, incorporating cutting-edge knowledge distillation techniques from the field of artificial intelligence into medical imaging problems. This framework enables the transfer of complex models' predictive capabilities to lightweight models. However, when it comes to teacher networks, researchers often lean towards designing more intricate networks and accumulating larger datasets in pursuit of better performance. Nevertheless, the complex modules involved entail numerous parameter calculations, leading to an escalating demand for hardware resources (such as memory and GPUs), resulting in a substantial computational burden.

In contrast, ML [8] breaks this pre-defined "strong-weak relationship" by fostering a collaborative learning approach, where multiple student networks learn from and guide each other throughout the training process. This dynamic framework eliminates the need for a predefined unidirectional knowledge transfer path between teachers and students. Instead, it facilitates bidirectional knowledge exchange among the networks. Zhang et al. [24] present a novel modality-aware mutual learning (MAML) method for effective and robust multi-modal liver tumor segmentation, which utilizes ML and achieves good results. Here, we also introduce ML into our method.

6.2.3 Attention Mechanism

The attention mechanism can improve the fineness of observation in some aspects, which means, through learning and training, the deep neural networks can learn the areas that need attention in each new image. It can be noted that the essence of the attention mechanism is to obtain a set of weight distributions that can be applied to the original image through learning. The most classic and well-known mechanism is SENet [25], which effectively constructs the interdependence between channels by simply squeezing each two-dimensional feature map. CBAM [26] further advances this idea by introducing space information. Hou et al. [27] embedded location information into channel attention and proposed a novel mobile network attention mechanism, which performs well in ImageNet classification.

Attention mechanism is widely used in medical segmentation, classification, registration, etc. Sinha and Do [28] introduced an attention mechanism to fully express the context dependence on long range, which can help in the fusion of local features and global features; meanwhile, it can also filter out irrelevant noise information. Attention U-Net [29] proposes the attention gate to replace the attention method used in image classification and the external organ positioning model used in image segmentation and improves the model's sensitivity to foreground pixels. Song et al. [30] proposed a self-attention mechanism specifically for cross-modal image registration, which achieves good results in prostate cancer biopsy through accurate fusion of transrectal ultrasound (TRUS) and magnetic resonance (MR) images. MDNet [31] introduces the auxiliary attention sharpening (AAS) module to assist the attention module in generating a more effective attention map, which can automatically generate a complete diagnosis report and display the image attention area when describing the image.

6.3 Method

In this chapter, we attempt to improve classification accuracy by mainly addressing the problems of a small amount of labeled data and high intra-class variation in medical image classification. To achieve this goal, we propose the AAM and ML strategy to form a dual-path network, as shown in Figure 6.1. For an input image, the features are extracted by two

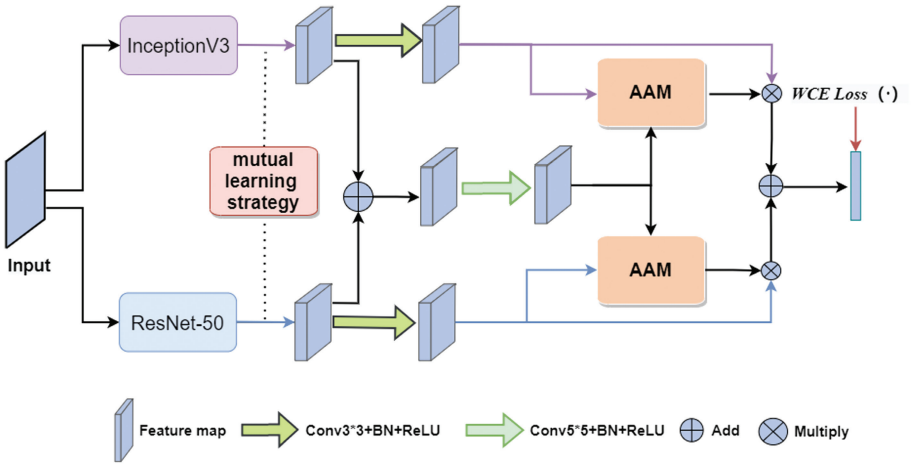


Figure 6.1 Overview of the architecture of DPMAN. The input image is fed to two pre-trained networks at the same time to extract the features, and the extracted features are added together to get fusion features. Meanwhile, the ML strategy is used within the dual path. After a series of convolutions, they are input into our proposed AAM module to output their respective feature maps. The features extracted by the pre-trained network are multiplied with their own feature maps element by element and finally fused and predicted.

different pre-trained networks, and the obtained features are fused and fed into the AAM to obtain the attention weights. At the same time, the ML strategy is used to promote the two networks to jointly learn the information they ignore. In this section, we introduce our network in detail.

6.3.1 Motivation

Compared with natural images, medical images have problems, such as a small amount of data, unbalanced classes, and high similarity between classes. To tackle these problems, predecessors have used transfer learning, distillation learning, and other methods to solve them, but these methods include a lot of parameters and calculations because they usually need a deep network for learning. On the other hand, attention mechanisms have been shown to be effective for high-level visual tasks by focusing on more informative channels [25] or spatial locations [32].

Therefore, we propose a dual-path architecture and an ML strategy that do not need a deep network as a teacher network but allow the two networks to learn from each other's distributions to correct their learned inter-class information. At the same time, to further learn to obtain the features between the dual paths, we propose an aggregate attention module to fuse and weight the features extracted from the two paths. Finally, considering the class imbalance in medical images, we use weighted cross-entropy loss to alleviate this problem.

6.3.2 Dual Path

Figure 6.1 shows the proposed dual-path structure. For a given input image, we use different pre-trained networks, ResNet-50 and InceptionV3, to extract features and obtain the features

$F_i (i \in \{1,2\})$ learned from different angles, respectively. F_1 and F_2 are added to fuse their information and obtain the fusion feature F_u. The fused feature F_u is fed into a convolutional layer with a kernel size of 5×5, and feature F_i is fed into a convolutional layer with a kernel size of 3×3. All convolutional layers are followed by a batch normalization layer.

The two sets of features are simultaneously input to our proposed aggregation module AAM to generate the corresponding attention map. F_1, F_2, and their respective attention maps are multiplied element by element; subsequently, they are added to obtain the final output after FC and sigmoid. The whole process can be expressed as the following formulas:

$$F_{a1} = AAM(f_{5\times5}(F_u), f_{3\times3}(F_1)) \cdot f_{3\times3}(F_1), \tag{6.1}$$

$$F_{a2} = AAM(f_{5\times5}(F_u), f_{3\times3}(F_2)) \cdot f_{3\times3}(F_2), \tag{6.2}$$

$$F_a = softmax(F_{a1} + F_{a2}), \tag{6.3}$$

where $f_{3\times3}$ represents a convolutional layer with a kernel size of 3×3, $f_{5\times5}$ represents a convolution with the filter size of 5×5, denotes element-wise multiplication, and *softmax* denotes the softmax function.

After obtaining the final output from the FC and sigmoid function, we can calculate the classification loss. Because of the class imbalance, we use the weighted cross-entropy (WCE) as a loss function, which strengthens its contribution for the minority class and reduces its contribution for the majority classes. The formula of the WCE loss is as follows:

$$L_{wce}(y, f(x)) = -\sum_{i=0}^{n} \omega_i y_i \ln f(x), \tag{6.4}$$

where ω_i is calculated according to the dataset in advance. Specifically, suppose the training dataset has M classes, and the number of samples in each class is n_i. We find that the median number of these M samples n_x, and the coefficient of the corresponding category is set to n_x/n_i.

6.3.3 Aggregate Attention

As illustrated in Figure 6.2, we propose AAM to measure the contribution of each pre-trained network, fusing their features and weighting attention for them. Although the fusion feature F_u encodes both information extracted by InceptionV3 and ResNet-50, it also inevitably introduces redundant noise. Instead of obtaining straightforward prediction from F_u, we propose AAM to aggregate the intra-class and inter-class information. From the work of CBAM [26], we can draw a conclusion that using both average-pooling and max-pooling operations simultaneously can greatly improve the representation ability of networks rather than using each independently. Thus, we use these two operations in our AAM.

For F_i extracted by a pre-trained network, such as InceptionV3 or ResNet-50, we use global average pooling to prevent losing too much high-dimensional information, and transfer the overall feature information to the next module for feature extraction. For the fusion feature F_u, we use global max-pooling to filter more useless information to select the features with better classification discrimination. After pooling, the features are extracted by 1×1 convolution. In order to further obtain useful information, we use 3×3 dilation convolution to expand the

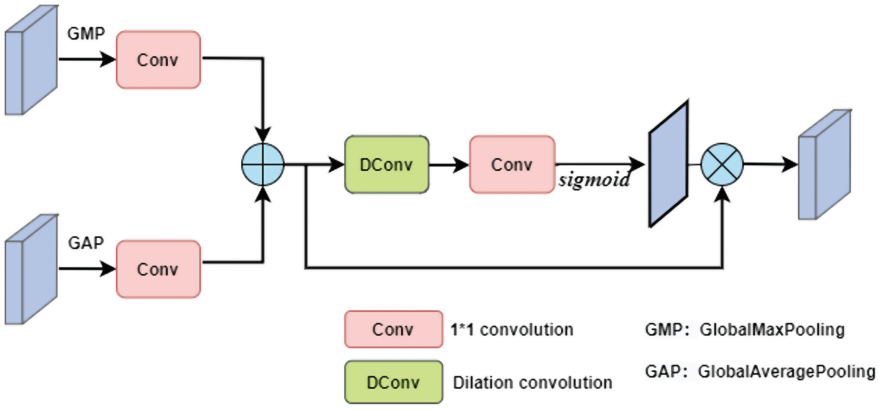


Figure 6.2 The architecture of our proposed aggregation attention module. The features extracted by pre-trained networks are aggregated with the fusion feature, where we use 3 × 3 dilation convolution to expand the receptive field. Finally, the added features are multiplied with the attention map to generate the outputs of AAM.

receptive field and obtain the attention map A_i by 1×1 convolution with the sigmoid function. Finally, the added features are multiplied with the attention map element by element to get the output of attention. Specifically, the aggregation attention is computed as

$$F_a = f_{1\times1}(GMP(F_i)) + f_{1\times1}(GAP(F_u)), \tag{6.5}$$

$$A_i = \sigma(f_{1\times1}(f_{dc}(F_a))), \tag{6.6}$$

$$F_{aam} = A_i \cdot F_a. \tag{6.7}$$

In the above formulas, $f_{1\times1}$ represents a convolution with the filter size of 1×1, f_{dc} represents a dilation convolution with the filter size of 3×3, GMP means global max pooling, GAP means global average pooling, and σ denotes the sigmoid function.

6.3.4 Mutual Learning Strategy

In order to solve the problem of lack of medical data, we introduce an ML strategy, as shown in Figure 6.3. For the two branches, InceptionV3 and ResNet50, we use softmax to get the prediction and the WCE loss as their own supervised learning loss. In addition, to not only learn the network's prediction of images but also pay attention to the error category information learned by the network, we use the soft Softmax, which is inspired by the distillation network [7], to get more useful information, such as intra-class variance and inter-class distance. The soft SoftMax is formulated as

$$p_i = \frac{exp(z_i / T)}{\sum exp(z_j / T)}, \tag{6.8}$$

where T is a temperature that is set to 2 in our method.

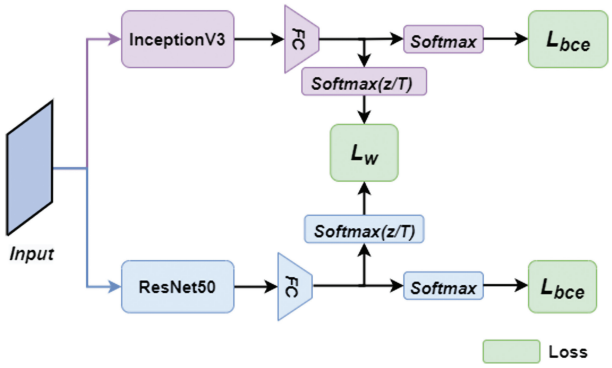


Figure 6.3 The architecture of our proposed mutual learning strategy. Each network is trained with weighted cross-entropy loss as a supervised learning loss, and a Wasserstein distance-based mimicry loss to match the probability estimates of its peers.

We use the Wasserstein distance to measure the difference between the probability distributions of the two branches. Wasserstein distance is a measure of the difference between two probability distributions by computing the cost of transform from one distribution to another. Compared to KL, even at low latitudes and when there is no overlap between distributions, the value of the Wasserstein distance still provides useful information. The Wasserstein distance loss L_w is shown as follows:

$$W(p_1, p_2) = inf_{\gamma S(p_1, p_2)} E(x, y)\,[\| x - y \|]. \tag{6.9}$$

We minimize the Wasserstein distance loss to promote learning from each other to achieve the effect of getting twice the result with half the effort. In summary, the loss we use to optimize our DPMAN contains two parts: one is cross-entropy loss, which keeps the prediction close to the ground truth, and the other is the Wasserstein distance loss, which makes the features learned by the two networks more similar. The total loss is shown as follows:

$$L = \alpha W(p_1, p_2) + \beta L_{wce}(y, f(x)), \tag{6.10}$$

Where α and β are the hyper-parameters.

6.4 Experiments

This section introduces our experimental settings and results. First, we report the details of our experiment setup. To verify the universality of our method in medical image classification, we report experimental results on three datasets with four comparative methods as follows: Yan et al. [33] (who utilized prior information by regularizing attention maps with regions of interest (ROIs) for melanoma recognition); Zhang et al. [34] (who proposed an attention residual learning convolutional neural network model, the proposed attention learning mechanism aims to exploit the intrinsic self-attention ability of DCNNs, i.e., using the feature maps learned by a high layer to generate the attention map for a low layer for skin lesion classification in

dermoscopy images); Xiao et al. [35] (who designed global and local networks, respectively, based on the combination of the LBP texture features and the deep Conv-features derived from DCNNs, and conducted several fusion strategies on the Global-DNN and Local-DNN for better performance); and Datta et al. [36] (who proposed soft-attention to boost the value of important features and suppress the noise-inducing features).

The three datasets used are the APTOS, HAM10000, and Brain Tumor datasets. From the perspective of image type, these datasets contain fundus photography, dermatoscopic images, and MRI. From the point of view of the lesion site, these datasets include eye, skin, and brain. In addition, we perform ablation experiments to prove the effectiveness of our AAM and ML strategy.

6.4.1 Implementation Details

We used the Adam [37] optimizer with an initial learning rate of 0.0001 to update the model weights during the training. In addition, when the accuracy of the validation set does not decrease within 10 epochs, we reduce the learning rate. We evaluated classification performance by overall accuracy (ACC), balanced multi-class accuracy (BMA), average precision (AP), and F1 score. Due to class imbalance, BMA was the most important indicator to measure multi-class tasks, which is defined as

$$\begin{aligned} BMA &= \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{RP_c} \\ &= \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FN_c}, \end{aligned} \tag{6.11}$$

where RP_c represents the total positive samples available for class c in the dataset, TP_c represents the true positives for class c, and FN_c represents false negatives for class c.

All the compared methods were randomly initialized. We used the Keras framework to conduct all experiments, which were implemented on the NVIDIA GeForce GTX 1080ti platform.

6.4.2 Results on HAM10000

The HAM10000 skin cancer dataset consists of 10,015 dermoscopic images. Cases include a representative collection of all important diagnostic categories in the realm of pigmented lesions: Actinic keratoses and intraepithelial carcinoma/Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines/seborrheic keratoses and lichen-planus–like keratoses, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv), and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc) [38, 39]. More than 50% of the lesions are confirmed through histopathology; the ground truth for the rest of the cases is either follow-up examination, expert consensus, or confirmation by in-vivo confocal microscopy.

We used 193 images provided by Kaggle as the validation set; for the other 10,015 images, we used 85% of these images as the training set and 15% as the test set. The original image size

Table 6.1 Results of the proposed network and other state-of-the-art methods on HAM10000. The best results are highlighted using boldface.

Model	Year	Accuracy(%)	BMA (%)	AP(%)	F1(%)
InceptionV3	2015	80.1	60	78.1	78.4
ResNet50	2016	81.5	61.4	80.0	80.2
Yan et al. [33]	2019	83.6	73.9	78.7	80.1
Zhang et al. [34]	2019	84.7	74.1	85.1	82.4
Xiao et al. [35]	2020	85.1	74.6	85.5	85.1
Datta et al. [36]	2021	85.3	72.4	85.9	85.3
Ours	2021	**86.7**	**75.8**	**86.8**	**85.6**

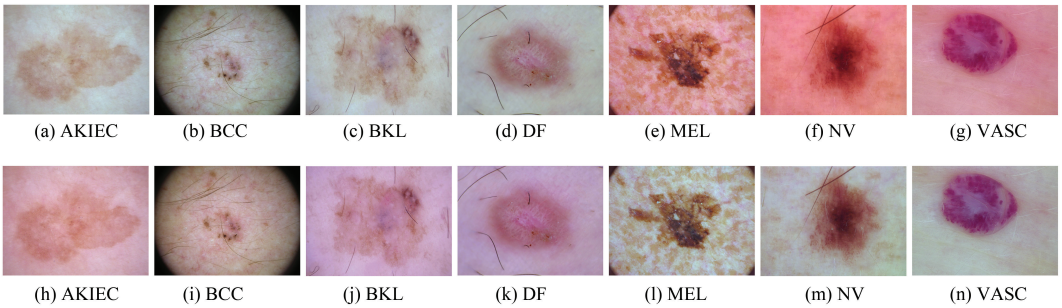


Figure 6.4 The (a)–(g) images are the original image in the dataset. Due to factors such as lighting and equipment, the color of the lesion area is very close to that of the background area. The (h)–(n) images are the processed images after the color constancy algorithm. As we can see, it is easy to distinguish between lesions and background.

is 600 × 450. Each image was resized to 256 × 256 in our experiments. In addition, to reduce the image noise caused by equipment lighting, we processed the dataset with a color constancy algorithm, which can help in clearly distinguishing the pathological area of the image from the background, as shown in Figure 6.4. For data augmentation, we adopted on-the-fly random shift, rotation, and horizontal flip for all training data to alleviate the over-fitting problem.

The class imbalance phenomenon in the HAM10000 dataset is serious. To alleviate this problem, we used a WCE loss, as described in Section 6.3.2, and the loss weights used in training were AKIEC 1.57, BCC 1.0, BKL 0.47, DF 4.49, MEL 0.46, NV 0.08, and VASC 3.63, respectively.

We evaluated the performance of our proposed method with the backbone we considered: ResNet50 and InceptionV3 and the state of-the-art methods on HAM10000. As shown in the following table 6.1, the proposed method outperforms existing methods with an improvement of 1.9%, 1.7%, and 3.4% in BMA, further validating the effectiveness of our proposed method.

6.4.3 Results on APTOS

The APTOS blindness detection dataset is captured by Aravind technicians of Aravind Eye Hospital and then relies on highly trained doctors to review the images and provide a diagnosis.

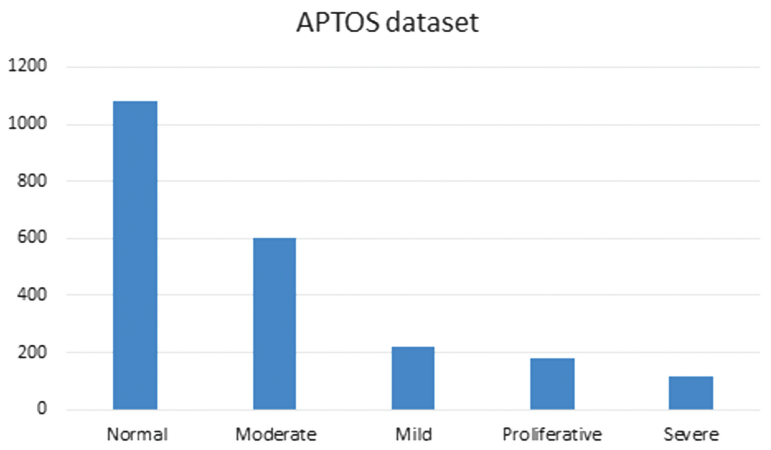


Figure 6.5 The data distribution of the APTOS dataset.

Table 6.2 Results of the proposed network and other state-of-the-art methods on the APTOS dataset. The best results are highlighted in boldface.

Model	Year	Accuracy(%)	BMA (%)	AP(%)	F1(%)
InceptionV3	2015	79.4	59.8	78.2	78.5
ResNet50	2016	79.1	56.0	78.0	77.4
Yan et al. [33]	2019	80.7	63.2	81.4	79.1
Zhang et al. [34]	2019	83.2	64.6	80.1	81.9
Xiao et al. [35]	2020	82.6	46.1	81.4	81.3
Datta et al. [36]	2021	79.1	64.2	80.2	79.5
Ours	2021	**83.4**	**65.7**	**82.6**	**82.8**

This dataset contains 3662 fundus images, which are classified into five grades according to the severity of diabetic retinopathy (DR), namely, normal cases, mild DR, moderate DR, severe DR, and proliferative DR cases [40]. We divided the dataset by 6:2:2, including 60% training set, 20% validation set, and 20% test set. Because the original image size is inconsistent, the original images were resized to 256 × 256. The data augmentation method was the same as in Section 6.4.2.

As shown in Figure 6.5, the class imbalance phenomenon in the APTOS dataset is serious; it can be clearly seen that the category with the most data volume is nearly 10 times different from the category with the least data volume. To alleviate this problem, we used a WCE loss, as described in Section 6.3.2, and the loss weights used in training are normal cases 0.2, mild DR 1.0, moderate DR 0.37, severe DR 1.93, and proliferative DR cases 1.25, respectively.

We compared the same methods as in Section 6.4.2. The classification results are summarized in the table 6.2. Under the same experimental setup and data augmentation, our method performs the best in all indicators.

6.4.4 Results on the Brain Tumor Dataset

The brain tumor dataset consists of 3264 MRI images, including four categories: glioma tumor, meningioma tumor, no tumor, and pituitary tumor [41]. We divided the datasets into train (60%),

validation (20%), and test (20%) splits. The datasets were resized to 256 × 256. To ensure fairness, we adopted the same comparison method as in Section 6.4.2.

We compared the same methods as in Section 6.4.2. The table 6.3 shows our experimental results on this dataset. It can be seen from the table that the task is not difficult because its category is balanced; even so, our method has improved other state-of-the-art approaches in all indicators.

6.4.5 Ablation Study

We conducted an ablation study to demonstrate the effectiveness of each component (AAM and ML strategy) of our method on the APTOS dataset.

In order to validate the effectiveness of the AAM, we removed the AAM and simply predicted with the fusion feature F_u, which was obtained by features extracted from the two pretrained networks: ResNet50 and InceptionV3. The results are shown in Table 6.4. Obviously, compared with our complete architecture, we can see that the AAM is necessary and plays a great role.

To validate the effectiveness of the ML strategy, we removed the interrelation and used only the L_{wce} as the total loss of the network, which can be expressed as $L = L_{wce}$. We conducted experiments on the APTOS dataset and compared the results with the proposed DPMAN, which are shown in Table 6.4. Our ML strategy leads to better performance with higher classification accuracy. The results suggest that ML strategy is a feasible means to obtain good representations for the dual-path architecture.

Table 6.3 Results of the proposed network and other state-of-the-art methods on the Brain Tumor dataset. The best results are highlighted in boldface.

Model	Year	Accuracy(%)	BMA (%)	AP(%)	F1(%)
InceptionV3	2015	90.5	90.9	90.6	90.5
ResNet50	2016	93.0	93.0	93.0	92.9
Yan et al. [33]	2019	95.7	95.2	95.4	95.1
Zhang et al. [34]	2019	96.5	96.8	96.8	96.9
Xiao et al. [35]	2020	95.1	95.6	95.5	95.1
Datta et al. [36]	2021	94.0	94.5	94.1	94.0
Ours	2021	**97.3**	**97.6**	**97.3**	**97.2**

Table 6.4 Ablation study for the mutual learning strategy and AAM

Mutual learning strategy	AAM	Accuracy	BMA	AP
		79.7	60.1	79.9
✓		81.4	62.9	80.4
	✓	81.7	60.9	81.0
✓	✓	**83.4**	**65.7**	**82.6**

6.5 Conclusion

Based on analyzing the problems existing in medical image classification, such as the small number of labeled images, high intra-class similarity and inter-class variance, and serious class imbalance, we propose a dual-path architecture and ML strategy to achieve better results with a small amount of data and prevent from overfitting due to small amount of data. Further, in order to solve the problem of high intra-class similarity and inter-class variance, we propose an AAM to fuse and weight the features extracted by dual paths to learn the features within the classes better. In addition, to solve the problem of data imbalance in medical images, we use WCE loss to drive the network training to focus more on classes with few samples so as to prevent the network predictions from tending to the categories with large amounts of data. The effectiveness of our proposed method is demonstrated by several comparative experiments and ablation experiments on three datasets.

References

[1] Moumen T. El-Melegy, Doaa Mohamed, Tarek ElMelegy, and Mostafa Abdelrahman. Identification of tuberculosis bacilli in zn-stained sputum smear images: A deep learning approach. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16–20, 2019*, pages 1131–1137. Computer Vision Foundation/IEEE, 2019.

[2] Tatiana Gabruseva, Dmytro Poplavskiy, and Alexandr A. Kalinin. Deep learning for automatic pneumonia detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14–19, 2020*, pages 1436–1443. Computer Vision Foundation/IEEE, 2020.

[3] Tianyi Zhao, Kai Cao, Jiawen Yao, Isabella Nogues, Le Lu, Lingyun Huang, Jing Xiao, Zhaozheng Yin, and Ling Zhang. 3d graph anatomy geometry-integrated network for pancreatic mass segmentation, diagnosis, and quantitative patient management. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19–25, 2021*, pages 13743–13752. Computer Vision Foundation/IEEE, 2021.

[4] Uchi Ugobame Uchibeke, Sara Hosseinzadeh Kassani, Kevin A. Schneider, and Ralph Deters. Blockchain access control ecosystem for big data security. *CoRR*, abs/1810.04607, 2018.

[5] Zilong Hu, Jinshan Tang, Ziming Wang, Kai Zhang, Ling Zhang, and Qingling Sun. Deep learning for image-based cancer detection and diagnosis-a survey. *Pattern Recognition*, 83:134–149, 2018.

[6] Sinno Jialin Pan, and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[7] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[8] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*, pages 4320–4328. Computer Vision Foundation/IEEE Computer Society, 2018.

[9] Dong Han, Qihua Li, Wei Cai, Yuwei Xia, Jia Ning, and Feng Huang. Research and application of artificial intelligence in medical imaging. *Big Data Research*, 5(1):42–70, 2019.

[10] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: General overview. *Korean Journal of Radiology*, 18(4):570–584, 2017.

[11] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[12] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19:221–248, 2017.

[13] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Vijay Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric J. Topol, Jeff Dean and Richard Socher. Deep learning-enabled medical computer vision. *NPJ Digit Med*. 2021 Jan 8;4(1): 5.

[14] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[15] Wentao Zhu, Chaochun Liu, Wei Fan, and Xiaohui Xie. Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification. In *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12–15, 2018*, pages 673–681. IEEE Computer Society, 2018.

[16] Afonso Menegola, Michel Fornaciali, Ramon Pires, Flávia Vasques Bittencourt, Sandra Avila, and Eduardo Valle. Knowledge transfer for melanoma screening with deep learning. In *14th IEEE International Symposium on Biomedical Imaging, ISBI 2017, Melbourne, Australia, April 18–21, 2017*, pages 297–300. IEEE, 2017.

[17] Eduardo Valle, Michel Fornaciali, Afonso Menegola, Julia Tavares, Flávia Vasques Bittencourt, Lin Tzy Li, and Sandra Avila. Data, depth, and design: Learning reliable models for melanoma screening. *CoRR*, abs/1711.00441, 2017.

[18] Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*, pages 4917–4926. IEEE, 2019.

[19] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*, pages 2661–2671. Computer Vision Foundation/IEEE, 2019.

[20] Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C. S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[21] Enes Ayan, and Halil Murat Ünver. Diagnosis of pneumonia from chest x-ray images using deep learning. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, pages 1–5. IEEE, 2019.

[22] Yanyang Gu, Zongyuan Ge, C. Paul Bonnington, and Jun Zhou. Progressive transfer learning and adversarial domain adaptation for cross-domain skin disease classification. *IEEE Journal of Biomedical and Health Informatics*, 24(5):1379–1393, 2019.

[23] Dian Qin, Jiajun Bu, Zhe Liu, Xin Shen, Sheng Zhou, Jing-Jun Gu, Zhihong Wang, Lei Wu and Hui-Fen Dai. Efficient Medical Image Segmentation Based on Knowledge Distillation. *IEEE Transactions on Medical Imaging* 40 (2021): 3820-3831.

[24] Yao Zhang, Jiawei Yang, Jiang Tian, Zhongchao Shi, Cheng Zhong, Yang Zhang, and Zhiqiang He. Modality-aware mutual learning for multi-modal medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 589–599. Springer, 2021.

[25] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[26] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018.

[27] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19–25, 2021*, pages 13713–13722. Computer Vision Foundation/IEEE, 2021.

[28] Ashish Sinha, and Jose Dolz. Multi-scale self-guided attention for medical image segmentation. *IEEE Journal of Biomedical and Health Informatics*, 25(1):121–130, 2021.

[29] Ozan Oktay, Jo Schlemper, Loc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018.

[30] Xinrui Song, Hengtao Guo, Xuanang Xu, Hanqing Chao, Sheng Xu, Baris Turkbey, Bradford J. Wood, Ge Wang, and Pingkun Yan. Cross-modal attention for MRI and ultrasound volume registration. In Marleen de Bruijne, Philippe C. Cattin, Stéphane Cotin, Nicolas Padoy, Stefanie Speidel, Yefeng Zheng, and Caroline Essert, editors, *Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part IV*, volume 12904 of *Lecture Notes in Computer Science*, pages 66–75. Springer, 2021.

[31] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*, pages 3549–3557. IEEE Computer Society, 2017.

[32] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. A2-nets: Double attention networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada*, pages 350–359, 2018.

[33] Yiqi Yan, Jeremy Kawahara, and Ghassan Hamarneh. Melanoma recognition via visual attention. In Albert C. S. Chung, James C. Gee, Paul A. Yushkevich, and Siqi Bao, editors, *Information Processing in Medical Imaging - 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings*, volume 11492 of *Lecture Notes in Computer Science*, pages 793–804. Springer, 2019.

[34] Jianpeng Zhang, Yutong Xie, Yong Xia, and Chunhua Shen. Attention residual learning for skin lesion classification. *IEEE Transactions on Medical Imaging*, 38(9):2092–2103, 2019.

[35] Feng Xiao, and Qiuxia Wu. Visual saliency based global-local feature representation for skin cancer classification. *IET Image Processing*, 14(10):2140–2148, 2020.

[36] Soumyya Kanti Datta, Mohammad Abuzar Shaikh, Sargur N. Srihari, and Mingchen Gao. Soft attention improves skin cancer classification performance. In Mauricio Reyes, Pedro Henriques Abreu, Jaime S. Cardoso, Mustafa Hajij, Ghada Zamzmi, Paul Rahul, and Lokendra Thakur, editors, *Interpretability of Machine Intelligence in Medical Image Computing, and Topological Data Analysis and Its Applications for Medical Data - 4th International Workshop, iMIMIC 2021, and 1st International Workshop, TDA4MedicalData 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings*, volume 12929 of *Lecture Notes in Computer Science*, pages 13–23. Springer, 2021.

[37] Diederik P. Kingma, and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio, and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, 2015.

[38] Noel C. F. Codella, David A. Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin K. Mishra, Harald Kittler, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In *15th IEEE International Symposium on Biomedical Imaging, ISBI 2018, Washington, DC, USA, April 4–7, 2018*, pages 168–172. IEEE, 2018.

[39] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1):1–9, 2018.

[40] Kaggle. Aptos 2019 blindness detection, 2019. https://www.kaggle.com/c/aptos2019-blindness-detection/data.

[41] Kaggle. Brain tumor classification (MRI), 2019. https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri.

Chapter 7

AGGNN: Attention-Gated Graph Neural Network for Text Classification

Ke Xu, Guoqiang Zhong,* Zhaoyang Deng, Chenxiang Sun, and Yuxu Mao

College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract

Text classification is a fundamental task in natural language processing. There have been many graph-based neural networks for this task with the capacity to learn complicated relational information between word nodes. Many studies have shown that graph-based neural networks are effective for text classification. However, existing research still has potential deficiencies in the capacity to learn complicated relational information between word nodes. In this chapter, we propose attention-gated graph neural network (AGGNN), a novel graph-based model in which every document is represented as a text graph, which can propagate and update the semantic information of each word node from their one-hop neighbors. In addition, we propose TextPool, a text-pooling method based on an attention mechanism. Keyword nodes with discriminative semantic information are extracted via TextPool, which also aggregates document embedding. In this case, text classification is transformed into a graph classification task. Extensive experiments on four benchmark datasets demonstrate that the proposed model outperforms the state-of-the-art methods for text classification.

Key Words: Text classification, Graph neural network, Attention mechanism, Graph-based model.

7.1 Introduction

Text classification is the process of assigning tags or categories to text according to its content, which has been widely studied and addressed in many real applications, such as sentiment analysis, topic labeling, spam detection, and intent detection [1]. In the early days, text classification was mainly done by manual annotation. However, the time consumed and expenses incurred are the main disadvantages. With the occurrence of machine learning, many researchers have found that machine learning techniques for text classification can alleviate problems.

* E-mail address: gqzhong@ouc.edu.cn

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

Traditional machine learning approaches involve naive Bayes [2], support vector machine [3], and *k*-nearest neighbors [4]. However, they are dictated by handcrafted features ineffective.

With the development of deep learning technologies, text classification has benefited a lot from them due to their potential to reach high accuracy with less need of engineered features. For example, convolutional neural networks (CNNs) [5] and recurrent neural networks (RNNs) [6] have made considerable contributions to text classification. However, in CNNs, the shape of filters seriously affects the effectiveness of the acquired context information, while RNNs are biased models and result in a large weight of the following locations. In order to alleviate the above problems of CNNs and RNNs, Lai et al. [7] proposed recurrent CNNs, and Wang et al. [8] proposed convolutional RNNs for text classification. However, their work focuses on the locality of words and cannot thus model long-distance dependencies, especially for non-continuous words.

To deal with this problem, graph neural networks (GNNs) [9] are widely used in text classification, which can establish an effective relationship structure and maintain global structure information. Specifically, one text graph is constructed when words or documents are represented as nodes, and their relationship is described by edges. On this basis, Yao et al. proposed TextGCN [10], where the text graph contains words and documents from the entire corpus as nodes and transforms the text classification problem into a node classification problem. Nevertheless, TextGCN adopts the means of building one graph for the entire corpus. There are two main drawbacks. First, because the entire corpus needs to be considered, the edge needs to occupy huge memory resources. Second, the expression ability of the edges is severely restricted. To be specific, the graph constructed using TextGCN involves the global relationship between words and documents, in which a large number of edges are used to capture the global structure, but the semantic relationship is fuzzy. Therefore, a graph that relies on the whole corpus contains too much redundancy, which weakens context-aware semantic information and leads to insufficient learning.

To avoid deficiencies, we propose a novel attention-gated graph neural network (AGGNN), which can capture long-distance dependencies between discontinuous words and learn context-aware semantic information. Instead of constructing one graph for the whole corpus, we construct a separate graph for each document, where nodes represent words and edges denote the semantic interactions obtained by word co-occurrence statistics. The information of the word node is propagated in the local neighborhood through the gate mechanism to obtain effective node embedding (i.e., the semantic information is aggregated). Thereafter, we propose an attention-based pooling method labeled TextPool, implemented by global and local attention mechanisms, to extract the most important word nodes as components for document embedding since the category of the text is usually determined by several keywords. In the end, the embedding of each document is constructed from the features of the extracted keyword nodes. Comprehensive experiments on four text classification datasets demonstrate the effectiveness of our proposed model.

In a nutshell, the contributions of this chapter can be summarized as follows:

- We propose a novel AGGNN, in which each word node aggregates the semantic information from its attention-weighted local neighbors.
- An attention-based text-pooling method (i.e., TextPool) is designed to extract semantically representative keyword nodes. We devise global and local attention mechanisms to make

the pooling strategy adapt to documents of different lengths and improve the accuracy of text classification.
- Extensive experiments on several benchmark datasets demonstrate that our proposed method outperforms the state-of-the-art methods for text classification.

7.2 Related Work

In this section, we review some related text classification methods, including traditional approaches, deep learning approaches, and graph-based approaches.

7.2.1 Traditional Methods for Text Classification

Traditional methods mainly focus on feature engineering. In addition to the commonly used bag-of-words features, many researchers have made other attempts. For example, Wang et al. [11] utilized *n*-grams and Chenthamarakshan et al. [12] utilized entities of ontologies in text classification. Other researchers have converted texts into graphs and applied feature engineering to graphs [13–15]. However, they have primarily depended on the handcrafted features at the cost of manpower and resource. On the contrary, our method can learn word representations without manual intervention.

7.2.2 Deep Learning for Text Classification

With the development of deep learning, many researchers have applied deep neural networks to text classification. Kim [5] designed a CNN with one-dimensional convolutions for sentence classification. Tai et al. [16] and Liu et al. [17] presented a training framework based on LSTM for text representation. Additionally, other researchers have shown that the effectiveness of word embeddings in deep learning can significantly improve the accuracy of text classification [18, 19]. Although the above methods have achieved promising results, they have critical limitations; that is, they have mainly focused on consecutive word sequences under a local space and failed to learn the contextual semantic information of words. In order to remedy this problem, we propose our graph-based method, which can build the relationship among words via the word co-occurrence information. In this manner, it can realize the interaction among non-consecutive words and make full use of the contextual information.

7.2.3 Graph Neural Networks for Text Classification

Recently, graph neural networks have received widespread attention [20–22], which can model data in non-Euclidean space, especially for relational data. Inspired by the success of CNNs, Kipf and Welling present a graph convolutional network (GCN) for several graph datasets [23]. Since then, many GCN-based methods have emerged for text classification tasks [10, 24–28]. Yao et al. proposed TextGCN and achieved promising results on several datasets [10]. However, TextGCN considers the words and documents of the entire corpus as nodes in the

graph, resulting in information redundancy and a lack of context-aware information. Huang et al. employed a message-passing mechanism for convolution [24], and Zhang et al. employed a gated graph neural network to receive information from adjacent neighbors [28]. Their methods have produced competitive results. However, in their model, the information on word nodes is not fully updated. To solve this problem, we design an attention gate to capture the relationship between the current word node and its one-hop neighbors. In this way, the effective node embeddings and the text-level graph embeddings are learnt.

7.3 Methodology

In this section, we introduce our novel graph-based text classification model in detail. There are four key components: graph construction, AGGNN, attention-based TextPool, and readout function. The overall architecture is shown in Figure 7.1.

7.3.1 Graph Construction

The input graph is constructed in the case of a given document, in which each unique word is described as a vertex, and the co-occurrence relationship between words is represented as an edge. Formally, the graph of text is defined as $\mathcal{G}=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}\ (|\mathcal{V}|=n)$ is the set of vertices (i.e., words) and $\mathcal{E}$ is the set of edges. Let $X\in\mathbb{R}^{n\times d}$ represent the feature matrix containing all n word nodes with their features, where d denotes the dimension of each word node. We define the adjacency matrix of the input graph $\mathcal{G}$ as A, where $A_{ij}=1$, if there exists an edge $(v_i,v_j)\in\mathcal{E}$ and $A_{ij}=0$ otherwise. D is a diagonal degree matrix, where $D_{ii}=\sum_j A_{ij}$ and $\hat{A}=D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the

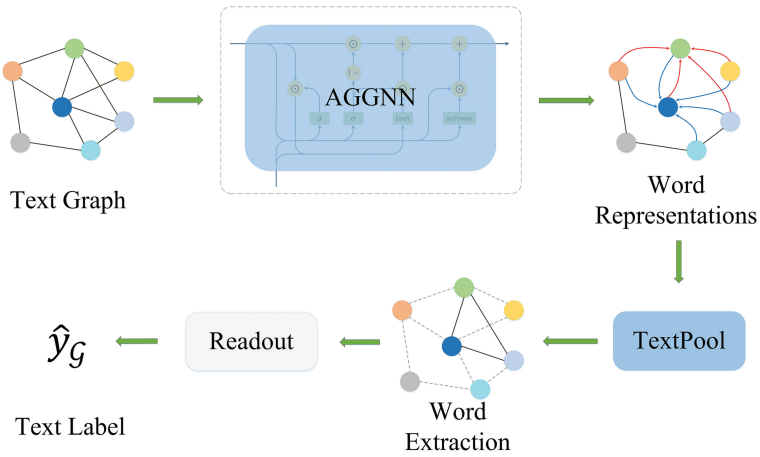


Figure 7.1 Diagram of a graph-based text classification model. For the input graph of the document, each node represents a word and is updated from its neighbors through AGGNN to generate a word representation. For example, the information propagation process of blue nodes and green nodes is represented by blue edges and red edges, respectively. The text representation is aggregated by the extracted key nodes for classification.

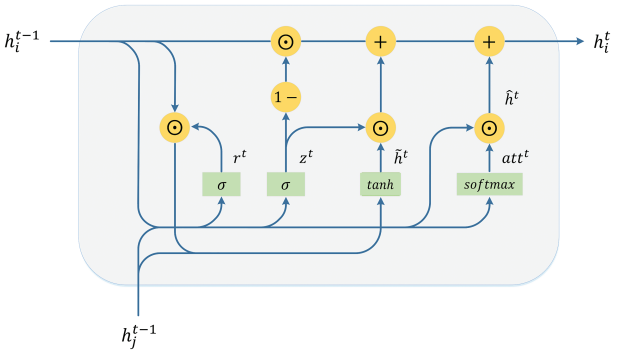


Figure 7.2 An illustration of AGGNN. The update gate z^t determines the transmission ratio of feature information of one-hop neighbor nodes, and the reset gate r^t determines the masking ratio of one-hop neighbor node feature information. In addition, the attention gate att^t enhances the propagation of one-hop neighbors' semantic information. The place where two curves merge represents the concatenation operation between matrices, and where two curves bifurcate represents the replication operation.

symmetric normalized adjacency matrix of the graph. The co-occurrence relationships describe the relationship between words that exist within a fixed-size sliding window (the size of the window is three at default) and are represented as undirected edges.

In addition, before the training phase, we utilize some preprocessing techniques on the text in a standard manner, including tokenization and stop word removal [14, 29]. The embeddings of the vertices $h \in \mathbb{R}^{n \times d}$ are initialized by the feature matrix of words $X \in \mathbb{R}^{n \times d}$. When the input graph of text is fed into AGGNN, the feature information of word nodes will be propagated to their neighbors, and the aggregating process of word representation begins.

7.3.2 Attention-Gated Graph Neural Network

Considering that long-term dependence exists among words, especially for non-consecutive words, we employ the gated graph neural networks [30] to control the information flow process of the word nodes and propose a novel AGGNN to aggregate the information of the word nodes. AGGNN comprises three gates: attention gate, update gate, and reset gate. An illustration of AGGNN is shown in Figure 7.1. Specifically, the input of AGGNN consists of two parts: the embeddings of word nodes $h^{t-1} \in \mathbb{R}^{n^{t-1} \times d}$, which are initialized by the feature matrix X, and the information a^t aggregated from every word node h_i^{t-1} and its one-hop neighbor words h_j^{t-1}, $j \in \mathcal{N}$. In addition, each word embedding can be propagated and aggregated from high-order neighbors by stacking AGGNN.

The attention gate is designed to capture the relationship between the central node and its one-hop neighbor nodes, determine how much neighbor information contributes to the central node, and will be incorporated into the central node. For each neighbor of the central node, the attention score is calculated by

$$att_i^t = softmax(h_i^{t-1}) = \frac{exp(h_i^{t-1}W)}{\sum_{j \in \mathcal{N}} exp(h_j^{t-1}W)}, \tag{7.1}$$

where $W \in \mathbb{R}^{d\times 1}$ denotes a shared linear transformation applied to each node and $\mathcal{N}$ denotes the set of one-hop neighbors of the center node (including the center node itself). The softmax function is employed to normalize the attention vector att^t.

Then, we obtain the attention-weighted embedding vector of each node v_i:

$$\hat{h}_i^t = \sum_{j\in\mathcal{N}} att_j^t \cdot h_j^{t-1}, \tag{7.2}$$

where $\mathcal{N}$ denotes the set of one-hop neighbors of node v_i (including v_i itself).

For the t-th AGGNN, the new embedding matrix h^t is updated as

$$a^t = \hat{A}^{t-1} h^{t-1} W_a, \tag{7.3}$$

$$z^t = \sigma\left(W_z a^t + U_z h^{t-1} + b_z\right), \tag{7.4}$$

$$r^t = \sigma\left(W_r a^t + U_r h^{t-1} + b_r\right), \tag{7.5}$$

$$\tilde{h}^t = \tanh\left(W_h a^t + U_h\left(r^t \odot h^{t-1}\right) + b_h\right), \tag{7.6}$$

$$h^t = \tilde{h}^t \odot z^t + h^{t-1} \odot \left(1 - z^t\right) + \hat{h}^t, \tag{7.7}$$

where z^t and r^t are the update gate and reset gate, respectively; all W and U are trainable weight matrices; all b are trainable bias vectors; $\odot$ is the Hadamard product; and σ is the sigmoid function.

The feature embedding of each node will be enhanced by the 1-hop neighbors that are closely related to the center node. Therefore, in this way, the representative nodes could be excavated in the node extraction phase.

7.3.3 Attention-based TextPool

AGGNN allows each node to update its embedding through the propagating and aggregating procedure. After that, hierarchical feature representations, which are crucial for text classification, should be learned. However, there is no natural notion of spatial locality in the text graph. In order to overcome this problem, we devise an attention-based text-pooling method (referred to as TextPool in Figure 7.1), which enables the model to focus on the most important nodes for the current text semantics.

A schematic diagram is shown in Figure 7.3. The whole process contains three steps. First, the attention score for each node is calculated via the attention function. Second, we extract the most valuable nodes based on their attention scores. Eventually, we aggregate feature embeddings from one-hop neighbors to the selected node. Therefore, even though the size and structure of the text graph are varied, TextPool will adaptively retain a portion of the nodes of the current text graph.

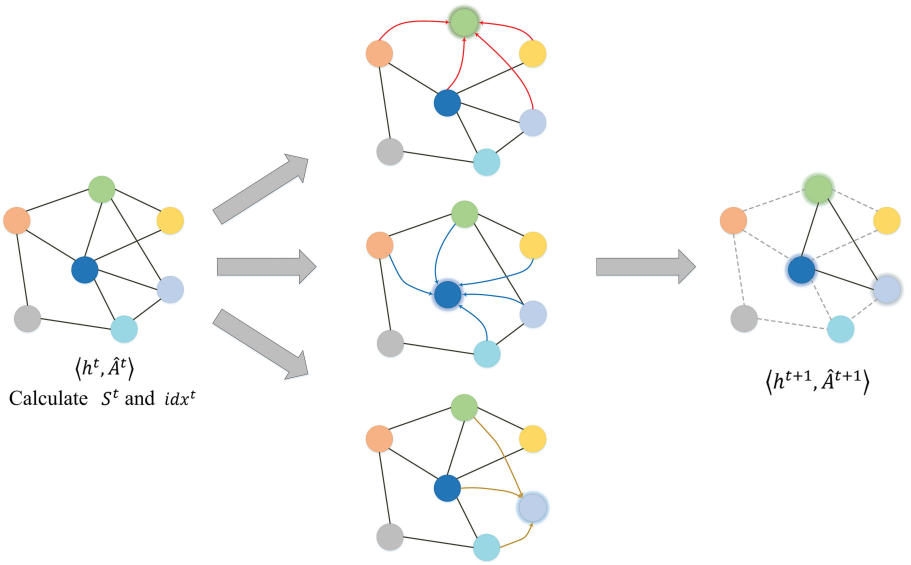


Figure 7.3 The schematic diagram of TextPool. According to the calculated attention score, the keyword nodes are extracted, and their feature embeddings are further updated. Then, a new subgraph is constructed using an adjacency matrix $\hat{A}^{t+1}$ and hidden state h^{t+1}.

We define attention function as $F(\cdot)$, under the premise of the current hidden state h^t and adjacency matrix $\hat{A}^t$, which generates a positive coefficient for each node. With a pooling ratio $\alpha \in (0,1]$, which is a hyper-parameter, the top $k = \lceil \alpha n^t \rceil$ nodes are extracted. The formulas of the process can be written as

$$S^t = F(h^t), \tag{7.8}$$

$$idx^t = top\_rank(S^t, k), \tag{7.9}$$

where $S^t \in \mathbb{R}^{n^t \times 1}$ is an attention vector and $top\_rank(\cdot)$ is a sorting function that returns the indices of top k nodes.

The hidden state of the current graph is updated with the attention score:

$$h'^t = S^t \odot h^t, h'^t \in \mathbb{R}^{n^t \times d}. \tag{7.10}$$

With the indexes of the extracted nodes, a subgraph is constructed whose adjacency matrix is described as $A_i'^t = \hat{A}_i^t, i \in idx^t$. In order to characterize the properties of the new subgraph, the hidden state of the next graph layer will be updated as

$$h^{t+1} = A^t h'^t, h^{t+1} \in \mathbb{R}^{k \times d}, \tag{7.11}$$

Since the distance between the nodes in the new subgraph represents the distance between the clusters representing the nodes in the original graph [31], the adjacency matrix is rescaled. Therefore, the new adjacency matrix is expressed as

$$\hat{A}^{t+1} = A'^{t}\hat{A}^{t}(A'^{t})^{T}, \hat{A}^{t+1} \in \mathbb{R}^{k\times k}. \tag{7.12}$$

Finally, we obtain a higher-level representation of the graph with h^{t+1} and $\hat{A}^{t+1}$. In particular, in order to generalize to graphs of different scales, the attention function $F(\cdot)$ can be implemented in two forms: global attention and local attention mechanisms.

7.3.3.1 Global Attention Mechanism

Naturally, each node needs to consider its contribution to the final text classification task. Under this circumstance, we can reckon the attention scores of all nodes in the given graph, which is referred to as the global attention mechanism. The generalized equation for calculating global attention is as follows:

$$F_i(h^t) = softmax_i(h^t W) = \frac{exp(h_i^t W)}{\sum_{j\in n} exp(h_j^t W)}, \tag{7.13}$$

where $W \in \mathbb{R}^{d\times 1}$ is a shared linear transformation.

For text graphs with small scale, global attention effectively extracts the representative nodes in semantics and facilitates the training process of the model.

7.3.3.2 Local Attention Mechanism

The word information contained in large-scale text graphs is intricate, and the global attention mechanism is hysteretic, resulting in huge time consumption. Furthermore, by propagating and aggregating information among nodes, a node will have analogous feature embeddings with its neighbors, that is, Laplacian smoothing [32]. In this case, the attention score of one node may have a close value with its neighbors, leading to the attention being stuck in a narrow region and the redundancy in a local scope. Calculating the local attention score for every node with its one-hop neighbors rather than all nodes of the graph could address this question. Therefore, we redesign the attention function called the local attention mechanism. It is defined as follows:

$$F_i(h^t) = \frac{\hat{A}_{ii} exp(h_i^t W \cdot \theta)}{\sum_{j\in\mathcal{N}} (\hat{A}_{ij} exp(h_j^t W \cdot \theta))} \sum_{j\in\mathcal{N}} \hat{A}_{ij}, \tag{7.14}$$

where $W \in \mathbb{R}^{d\times 1}$ denotes a shared linear transformation, $\mathcal{N}$ denotes the set of one-hop neighbors of each node (including the center node itself), and θ is a learnable parameter served as a regulative scalar.

In fact, the fractional part of equation (7.14) is a weighted softmax function, in which the attention score of each word node is produced with a distance between the vertex i and j (i.e., $\hat{A}_{ij}$). Moreover, we take the degree of each node into consideration to eliminate the negative impact induced by the number of neighbors and improve the applicability, that is, multiplied by the degree of each node.

7.3.4 Readout Function

When the information of word nodes is sufficiently propagated and the most significant nodes is extracted, a text-level graph representation h_G is obtained. Since the nodes in the final pooling

layer are critical for the text's category, we utilize an averaging method and a maximum method to integrate the information of each node. The readout function is defined as

$$h_{\mathcal{G}} = \frac{1}{k}\sum_{v=1}^{k} h_v^{t+1} \,\|\, \max_{v=1}^{k} h_v^{t+1}, \tag{7.15}$$

where $\|$ denotes the concatenation operation.

The predicted text label will be output based on $h_{\mathcal{G}}$:

$$\hat{y} = softmax(ReLU(Wh_{\mathcal{G}} + b)), \tag{7.16}$$

where W is a transformation matrix mapping the graph embedding into an output space, and b is a bias.

Finally, the cross-entropy loss function $\mathcal{L}$ is devised to minimize the distance between the ground truth label and the predicted label. It is denoted as follows:

$$\mathcal{L} = -\sum_{i} y_i log(\hat{y}_i), \tag{17.17}$$

where y_i is the i-th element of the one-hot ground truth label.

7.4 Experiments

In this section, we describe our experimental details and report our results. First, the benchmark datasets and related baseline models are introduced in detail. Then, we introduce the experimental setups. Third, the effectiveness of the proposed model is proved by comparing it with other graph-based models. Finally, we discuss the ablation study and parameter sensitivity.

7.4.1 Datasets

Extensive experiments were conducted to verify the performance of our graph-based model on four benchmark datasets for text classification, including Ohsumed, R8, R52, and Movie Review (MR). Table 7.1 summarizes the statistics of four datasets. It is noted that the statistics of these datasets are from Yao et al. [10].

- The Ohsumed dataset[1] is a collection consisting of medical abstracts with 23 cardiovascular disease categories. Each document in the collection has one or more associated categories. Considering that our task is a single-label text classification, we selected 7,400 documents with only one category and excluded the remaining documents. The training set contains 3,357 documents, and the test set contains 4,043 documents.

1 http://disi.unitn.it/moschitti/corpora.htm

Table 7.1 The statistics of datasets.

Dataset	Docs	Training	Test	Words	Classes	Average length
Ohsumed	7,400	3,357	4,043	14,157	23	135.82
R8	7,674	5,485	2,189	7,688	8	65.72
R52	9,100	6,532	2,568	8,892	52	69.82
MR	10,662	7,108	3,554	18,764	2	20.39

- The R8 and R52 datasets[2] are both subsets of Reuters-21578 text categorization collection. R8 has eight categories split into 5,485 training and 2,189 test documents. R52 has 52 categories split into 6,532 training and 2,568 test documents.
- MR[3] is a movie review dataset, and each review is associated with one sentiment label, which is positive or negative. The entire corpus consists of 5,331 positive and 5,331 negative reviews. In our experiments, we split them following the setups in [33].

Before the above datasets were fed into the model for training, we first pre-processed them by cleaning and tokenizing the text, similar to Kim [5]. We then removed stop words defined in Natural Language Toolkit (NLTK)[4] and low-frequency (less than five times) words in the above datasets except for MR, which is too short.

7.4.2 Baselines

We compared our method with the following baseline models:

- **TF-IDF+LR**: Bag-of-words model with term frequency-inverse document frequency weighting. The text classifier is established by Logistic Regression.
- **CNN**: Convolutional neural network defined by Kim [5], which is proposed for sentence-level classification with pre-trained word embeddings.
- **LSTM**: It is an RNN model defined by Liu et al. [17] for text classification with pre-trained word embeddings.
- **Bi-LSTM**: Bi-directional LSTM model pre-trained with word embeddings.
- **TM**: A Tsetlin Machine proposed by Yadav et al. [34], which extracts semantically related words from pre-trained word representations as input features.
- **fastText**: A model proposed by Joulin et al. [18] and treats average word or *n*-gram embeddings as document embeddings.
- **SWEM**: A simple word embedding model [35] in which simple pooling strategies are operated over word embeddings.
- **Graph-CNN**: A graph-CNN model proposed by Defferrard et al. [36], who operated convolutions over word embedding similarity graphs with Chebyshev filter.

2 https://www.cs.umb.edu/ smimarog/textmining/datasets/
3 http://www.cs.cornell.edu/people/pabo/movie-review-data/
4 http://www.nltk.org/

- **TextGCN**: A graph-based text classification method proposed by Yao et al. [10], which constructs a single large graph for the whole corpus.
- **SSGC**: This technique uses a modified Markov Diffusion Kernel to derive a variant of GCN [37].
- **Huang**: A graph-based text classification method that builds graphs for each input text with global parameters sharing [24].

7.4.3 Experimental Settings

For all above data sets, 90% of the text in the training set was randomly selected for training and the remaining 10% for validation. We utilized the Adam optimizer [38] with an initial learning rate of 0.01 and L_2 weight decay was set to 10^{-4}. The Xavier initializer [39] was used to initialize the parameters. We set the dropout rate as 0.5 and the epoch number as 400. The length of the sliding window was set to 3, and the pooling ratio α in TextPool was 0.5. For baseline models, we used default parameter setups as in their published papers. Additionally, for all models with pre-trained word embeddings in our experiments, we employed 300-dimensional pre-trained GloVe word embeddings [40] as the input word features.

In our work, we stacked two AGGNN layers to propagate the semantic information from neighbors. In the node extraction phase, the Text Pool layer is used. As with the text classifier, we constructed a two-layer fully connected layer with Rectified Linear Unit (ReLU) activation function. The number of AGGNN layers was adjusted according to the inherent attributes of different datasets, and we discuss them in the parameter sensitivity subsection.

7.4.4 Test Results

As shown in Table 7.2, it is discernible that graph-based models generally perform better than other types of models, such as TF-IDF+LR, CNN, LSTM, Bi-LSTM, and fastText, which indicate that graph-based models have a good ability to learn text information. Specifically, the graph constructed for each text contains effective word representations and relationships between words, and the graph-based model can propagate and aggregate information from various neighbors to the central node, which is a defect of other types of models.

It is worth noting that our model outperforms other graph-based models such as Graph-CNN, TextGCN, and Huang's methods. Graph-CNN utilizes a large-window bag-of-words model to represent documents, which cannot effectively distinguish information between different words. TextGCN constructs a single text graph for the entire corpus, which considers the global relationship between words and documents. However, due to the information redundancy in the process of tag message transmission, word embedding has not been fully learned. Huang proposed a message-passing mechanism to collect information from neighbors, and the edges of the graph are globally fixed between words [24]. Huang's and our graph models have promising results compared to other graph models. Because the graphs in Huang and our model are built for each document rather than the entire corpus, the burden of dependency between a single text and the entire corpus is removed. Nevertheless, Huang employs global shared edge weights in the whole corpus, resulting in the model needing to utilize an extra matrix to store the edge weights, and its shape is $|\mathcal{V}| \times |\mathcal{V}|$. Overall, our graph-based model has a more stable and excellent performance than Huang's method.

Table 7.2 Test accuracy (%) of our method and other compared models. We ran all models 10 times, and the variation range of results is reported with mean ± standard deviation. Notice that some baseline results stem from Yao et al. [10], and the best results are highlighted with boldface.

Model	Ohsumed	R8	R52	MR
TF-IDF+LR	54.66 ± 0.00	93.74 ± 0.00	86.95 ± 0.00	74.59 ± 0.00
CNN [5]	58.44 ± 1.06	95.71 ± 0.52	87.59 ± 0.48	77.75 ± 0.72
LSTM [17]	51.10 ± 1.50	96.09 ± 0.19	90.48 ± 0.86	77.33 ± 0.89
Bi-LSTM	49.27 ± 1.07	96.31 ± 0.33	90.54 ± 0.91	77.68 ± 0.86
TM [34]	-	97.50 ± 1.12	88.59 ± 1.20	77.51 ± 0.60
fastText [18]	57.70 ± 0.49	96.13 ± 0.21	92.81 ± 0.09	75.14 ± 0.20
SWEM [35]	63.12 ± 0.55	95.32 ± 0.26	92.94 ± 0.24	76.65 ± 0.63
Graph-CNN [36]	63.86 ± 0.53	96.99 ± 0.12	92.75 ± 0.22	77.22 ± 0.27
TextGCN [10]	68.36 ± 0.56	97.07 ± 0.10	93.56 ± 0.18	76.74 ± 0.20
SSGC [37]	68.50 ± 0.10	97.40 ± 0.10	94.50 ± 0.20	76.70 ± 0.00
Huang [24]	69.40 ± 0.60	97.80 ± 0.20	94.60 ± 0.30	78.86 ± 0.34
Ours	**70.26 ± 0.38**	**98.18 ± 0.10**	**94.72 ± 0.29**	**80.03 ± 0.22**

7.4.5 Ablation Study

7.4.5.1 The Attention Gate in AGGNN

In this subsection, we validated the effectiveness of the attention gate in AGGNN that utilizes a gate mechanism to update the semantic information between nodes. As exhibited in Table 7.3, the model without an attention gate in AGGNN is denoted by *w/o A*, while *w/ A* denotes the model with an attention gate in AGGNN. By observing and comparing the experimental results, we notice that the attention gate of AGGNN can significantly improve the performance of our model, no matter whether TextPool is achieved through global or local attention. Therefore, the attention gate is conducive to the dissemination of semantic information between nodes in the update phase of word embedding.

7.4.5.2 The Attention Mechanism in TextPool

As shown in Table 7.3, we discussed different attention mechanisms in TextPool, including global and local attention. As for the performance of TextPool, we can find that local attention performs better than global attention in long documents, such as on the Ohsumed dataset, while global attention is a great choice in short documents, such as in the MR dataset. The results reveal that in short documents, global attention is a simple but great choice, while in long documents, local attention pays more attention to different semantic scopes and ignores redundant semantic information.

7.4.6 Parameter Sensitivity

Figure 7.4 exhibits the performance of our model by stacking various numbers of AGGNN layers. The stacking of AGGNN layers allows for the indirect propagation of semantic information

Table 7.3 Comparisons of classification accuracy of models with attention gate (w/ A) and without attention gate (w/o A) in AGGNN

Model	Ohsumed	R8	R52	MR
TextPool-G w/o A	67.89	96.72	92.09	77.84
TextPool-G w/ A	**69.05**	**97.82**	**93.16**	**80.03**
TextPool-L w/o A	68.60	96.37	92.69	76.51
TextPool-L w/ A	**70.26**	**98.18**	**94.70**	**78.65**

TextPool-G represents the TextPool implemented by the global attention mechanism, and TextPool-L represents the TextPool implemented by the local attention mechanism. The best results are highlighted in boldface.

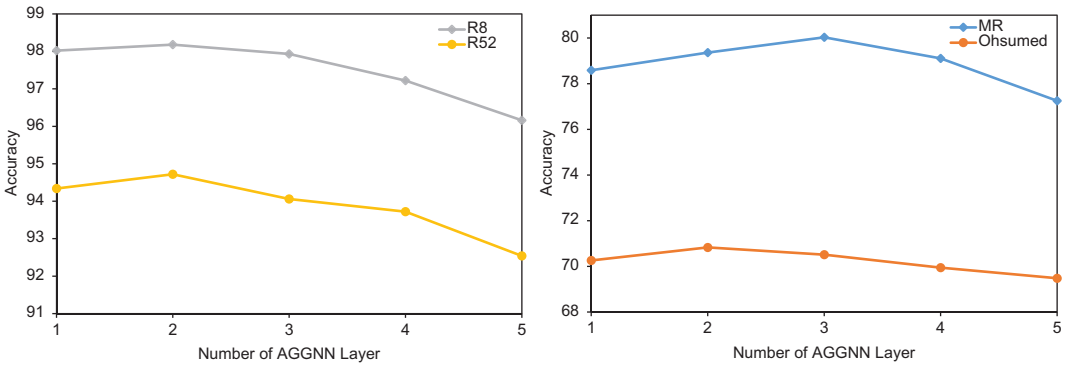


Figure 7.4 Accuracy (%) with different AGGNN layers.

from more neighbors rather than just one-hop neighbors. We find that the embeddings of nodes could be more fully updated, and the experimental results of the model are more accurate as the number of AGGNN layers increases. However, the performance tends to decline when the layer continues to increase, which may be related to excessive information diffusion, that is, over-smoothing. In Figure 7.4, the results on different datasets show similar trends. The difference is that the accuracy of MR decreases after the third stacking, while the accuracy of other datasets begins to decrease after the second stacking.

Figure 7.5 illustrates the performance of our model with different sliding window sizes on different datasets. Obviously, it can be observed that our model can achieve the best performance when the sliding window size is 3. Ultimately, we employed a sliding window with size 3 in our experiments.

7.5 Conclusion

In this chapter, we propose a novel graph-based text classification model AGGNN to construct a separate text graph for each document. In particular, the proposed AGGNN is used to learn word embedding, where the semantic information of each node is propagated and aggregated from its local neighbors through the gate mechanism. We also designed an attention-based TextPool method to select discriminative nodes based on the calculated attention scores. In

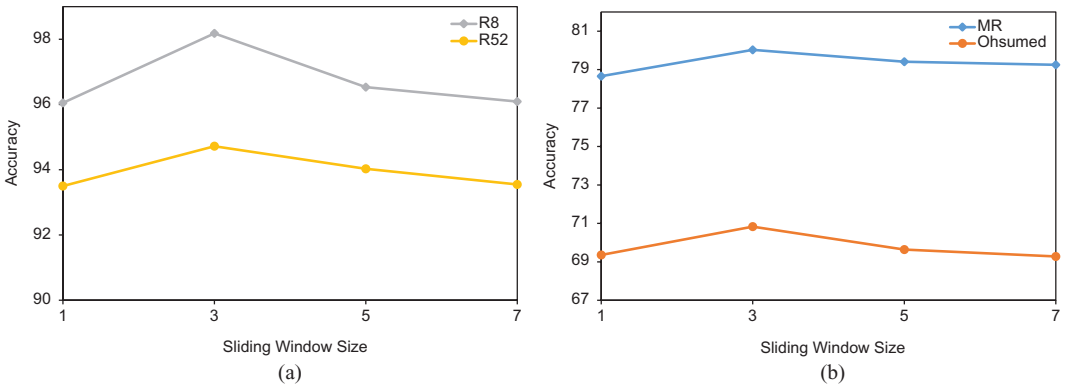


Figure 7.5 Accuracy (%) with different sliding window sizes.

addition, document embedding is obtained through an end-to-end training architecture. A large number of experiments on multiple text classification benchmark datasets demonstrate the effectiveness of our model.

References

[1] Aggarwal, C.C., Zhai, C.: A survey of text classification algorithms. In: Aggarwal, C., Zhai, C. (eds) *Mining Text Data*, pp. 163–222. Springer (2012)

[2] Peng, F., Schuurmans, D.: Combining naive bayes and n-gram language models for text classification. In: *Proceedings of the European Conference on Information Retrieval Research*, pp. 335–350 (2003)

[3] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: *Proceedings of the European Conference on Machine Learning*, pp. 137–142 (1998)

[4] Kang, H., Nam, K., Kim, S.: The decomposed k-nearest neighbor algorithm for imbalanced text classification. In: *Proceedings of International Conference on Future Generation Information Technology*, pp. 87–94 (2012)

[5] Kim, Y.: Convolutional neural networks for sentence classification. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1746–1751 (2014)

[6] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: *Proceedings of the Annual Conference of the International Speech Communication Association*, pp. 1045–1048 (2010)

[7] Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2267–2273 (2015)

[8] Wang, R., Li, Z., Cao, J., Chen, T., Wang, L.: Convolutional recurrent neural networks for text classification. In: *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–6 (2019)

[9] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. *IEEE Transactions on Neural Networks* 20(1), 61–80 (2009)

[10] Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7370–7377 (2019)

[11] Wang, S.I., Manning, C.D.: Baselines and bigrams: Simple, good sentiment and topic classification. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 90–94 (2012)

[12] Chenthamarakshan, V., Melville, P., Sindhwani, V., Lawrence, R.D.: Concept labeling: Building text classifiers with minimal supervision. In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1225–1230 (2011)
[13] Luo, Y., Uzuner, Ö., Szolovits, P.: Bridging semantics and syntax with graph algorithms – state-of-the-art of extracting biomedical relations. *Briefings in Bioinformatics* 18(1), 160–178 (2017)
[14] Rousseau, F., Kiagias, E., Vazirgiannis, M.: Text categorization as a graph classification problem. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1702–1712 (2015)
[15] Skianis, K., Rousseau, F., Vazirgiannis, M.: Regularizing text categorization with clusters of words. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1827–1837 (2016)
[16] Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 1556–1566 (2015)
[17] Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. In: *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 2873–2879 (2016)
[18] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pp. 427–431 (2017)
[19] 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. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 2321–2331 (2018)
[20] Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding. In: *Proceedings of the Web Conference 2020*, pp. 2331–2341 (2020)
[21] Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. arXiv preprint arXiv:1812.04202 (2018)
[22] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Sun, M.: Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434 (2018)
[23] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: *Proceedings of the International Conference on Learning Representations*, pp.1–14 (2017)
[24] Huang, L., Ma, D., Li, S., Zhang, X., Wang, H.: Text level graph neural network for text classification. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 3442–3448 (2019)
[25] Jose, J.M., Yilmaz, E., Magalhães, J., Castells, P., Ferro, N., Silva, M.J., Martins, F.: VGCN-BERT: Augmenting BERT with graph embedding for text classification. In: *Advances in Information Retrieval: 42nd European Conference on IR Research*, pp. 369–382 (2020)
[26] Liu, X., You, X., Zhang, X., Wu, J., Lv, P.: Tensor graph convolutional networks for text classification. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8409–8416 (2020)
[27] Li, C., Peng, X., Peng, H., Wang, L., Li, J.: TextGTL: Graph-based transductive learning for semi-supervised text classification via structure-sensitive interpolation. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2680–2686 (2021)
[28] Zhang, Y., Yu, X., Cui, Z., Wu, S., Wen, Z., Wang, L.: Every document owns its structure: Inductive text classification via graph neural networks. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 334–339 (2020)
[29] Blanco, R., Lioma, C.: Graph-based term weighting for information retrieval. *Information Retrieval* 15(1), 54–92 (2012)
[30] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. In: *Proceedings of the International Conference on Learning Representations*, pp.1–20 (2016)

[31] Huang, J., Li, Z., Li, N., Liu, S., Li, G.: Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6479–6488 (2019)

[32] Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 32(1): 3538–3545 (2018)

[33] Tang, J., Qu, M., Mei, Q.: Pte: Predictive text embedding through large-scale heterogeneous text networks. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174 (2015)

[34] Yadav, R.K., Jiao, L., Granmo, O.C., Goodwin, M.: Enhancing interpretable clauses semantically using pretrained word representation. arXiv preprint arXiv:2104.06901 (2021)

[35] Shen, D., Wang, G., Wang, W., Min, M.R., Su, Q., Zhang, Y., Li, C., Henao, R., Carin, L.: Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 440–450 (2018)

[36] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 3837–3845 (2016)

[37] Zhu, H., Koniusz, P.: Simple spectral graph convolution. In: *International Conference on Learning Representations*, pp.1–15 (2020)

[38] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: *Proceedings of the International Conference on Learning Representations* (2015)

[39] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256 (2010)

[40] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543 (2014)

Chapter 8

Stock Market Prediction Based on LSTA-GAN

Hao Li[1], Daewon Choi[2,3], and Guoqiang Zhong[1]

[1] College of Computer Science and Technology, Ocean University of China, Qingdao, China

[2] Center for BRICS Studies, Fudan University, Shanghai, China

[3] AI Economy Research Center, School of Economics, Shanghai University, Shanghai, China

Abstract

With the development of deep learning theory, models related to deep learning have started to be widely used in the field of finance. Deep finance is positioned as an important area of financial engineering because of its powerful data processing and prediction capabilities. Among them, stock price prediction is the most popular areas of interest. Economic theory assumes that stock prices can reflect all available information, but dynamic stochastic processes can generate asymmetric information for arbitrage. Compared to traditional autoregressive integrated moving average (ARIMA) models and generalized autoregressive conditional heteroskedasticity (GARCH) models, deep learning models have achieved significant improvements in stock price forecasting and possess better forecasting performance. Deep learning models are known to outperform traditional machine learning models in stock price forecasting. Most currently available deep learning models on stock price prediction use single or dual algorithms, but there have been few attempts at multiple combination algorithms. In this chapter, we propose a more novel generative adversarial network model for stock price prediction, which is a combination of long short-term attention (LSTA) as a generator—a combination of long short-term memory (LSTM) and attention mechanisms—and a multilayer perceptron (MLP) as a discriminator. The generator uses LSTA to mine the relevant data about a stock from its historical data and predict the next day's data and then uses a discriminator designed by the MLP to distinguish the generated data from the real data. We used daily stock price data from the S&P 500 index and attempted to predict the daily closing stock prices. The experimental results show that our LSTA-generative adversarial networks (GAN) model outperforms the existing deep learning models in stock price prediction. This result suggests that our LSTA-GAN model can generate novel solutions for stock price prediction and further strengthen the empirical contribution to deep learning-based asymmetric information theory.

Key Words: Stock price prediction, Attention, LSTA, GAN

8.1 Introduction

Stock prices are determined by a complex interaction of domestic and international macroeconomic and microeconomic factors, as well as commercial products and other dynamics. Stock

In: Attention Augmented Learning Machines
Editors: Guoqiang Zhong and Jinxuan Sun
ISBN: 979-8-88697-780-6

price forecasting has long been a theoretical challenge inherent in economics and business applications. At the theoretical level, the efficient market hypothesis (EMH) asserts that current stock prices should contain all available information and, therefore, there is little room for repeated arbitrage. On the other hand, incomplete information theory asserts that stock markets are complex and volatile and that prices are formed based on asymmetric information. Therefore, there is scope for dynamic price arbitrage and speculation to justify stock price forecasts.

At the technical level, the performance of stock price forecasting varies depending on the model and the technique applied. In recent decades, traditional econometric tools for time series analysis, such as autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroskedasticity (GARCH), have been widely used for stock price forecasting. Recently, however, machine learning models such as special vector machines (SVMs) and random forests, as well as deep learning tools and techniques such as convolutional neural networks (CNNs), long short-term memory (LSTM), and generative adversarial networks (GANs), have shown much higher performance compared to traditional methods.

It is worth noting in this context that deep learning models have made a significant contribution to the analysis of time series economics and have had an unparalleled impact on stock price forecasting. Over time, traditional statistical methods in economics have shifted their analytical paradigm from statistical learning to deep learning models based on big data analysis, data generation, and data transformation. Deep learning has transformed a disciplined, active set of economics into a new category of artificially intelligent economics.

Recent time-series deep learning algorithms for stock price prediction have been mainly based on LSTM, Attention, and GAN models. Some of the latest models attempt to combine them into hybrid models of LSTM-Attention (LSTA), LSTM-GAN, or Attention-GAN. However, few studies have combined these models into a single structure for time series prediction. To fill this gap, this chapter focuses on hybrid GAN analysis with LSTM-Attention as a generator and MLP as a discriminator. In the process, we will contribute to the existing literature by making several novel points as follows.

1. In contrast to traditional analyses that rely on a dual combination of LSTM-Attention or GAN-LSTM models, we devise a novel hybrid time series approach that uses an LSTA-GAN model to predict stock prices.
2. We demonstrate that our LSTA-GAN model, which combines the time series algorithm of LSTM and the encoder–decoder type algorithm of attention to develop a generator to synthesize the next day's stock price data, is distinguished from the data identification of MLPs and has a better performance than other models, such as the common model, LSTM-Attention, or GAN-LSTM models.
3. We can further improve the performance of stock price forecasting by applying novel hybrid deep learning algorithms. This confirms that deep learning not only helps to bridge the gaps in traditional economic theory, namely, efficient market hypothesis EMH, rational expectations theory, and incomplete information theory, but also yields new insights into data-driven algorithmic economics.

8.2 Related Works

8.2.1 Deep Learning Models for Stock Market Forecasts

We have identified three main trends in our compilation of existing work on deep learning models for stock price prediction. First, there has been a shift in approach from RNNs to

LSTMs due to the disappearance and explosion of gradients and the dependence on LSTMs. Some authors have introduced multi-input LSTMs, while others have used attention models in conjunction with LSTMs to improve the performance of the models. For example, Li et al. [1] used a multi-input LSTM model to extract information from low correlators, used additional input gates to discard noise, and considered other relevant stock prices to improve prediction accuracy.

Cheng et al. [2] used an attention-based LSTM model to predict stock price movements and formulate trading strategies. Kim and Kang [3] compared various deep learning models, such as MLP, one-dimensional convolutional neural network (1D CNN), stacked LSTM, attention network, and weighted attention for financial time series forecasting.

Some authors attempted to combine LSTM with wavelet transforms and gated recursive units (GRUs). For example, Qiu et al. [4] combined LSTM and attention mechanisms and used wavelet transforms to denoise historical data to train its features and build price forecasting models. Experimental results on the S&P 500 and Dow Jones Industrial Average datasets show that attention-based LSTM models exhibit higher performance with coefficients of determination above 0.94 and mean squared errors below 0.05.

Other authors attempted to use specific attention models for stock price prediction, such as temporal pattern attention (TPA) with LSTM. For example, Wei et al. [5] used temporal pattern attention and long short-term memory (TPA-LSTM) for stock price prediction to overcome the problem of long-term dependence. The results show that predicting stock index prices using the TPA-LSTM algorithm can achieve better prediction performance than traditional deep neural networks, such as RNN and LSTM.

In addition, another variation of the attention model is the use of a dual deep learning model based on hierarchical attention networks with stock data coding. For example, Huang et al. [6] applied a dual attention-based deep learning model. The first employs an article selection attention network to identify the core factors of the news and convert them into low-dimensional vectors. The second employs a time-series attention network that combines the output of the first model with trading data as input. The model learned the differences between the stock data and made more accurate predictions with higher performance.

Another interesting work is the use of multidimensional attention-based LSTM (MD-ALSTM). For example, Yu and Kim [7] applied a two-dimensional attention-based long and short-term memory (2D-Attention-LSTM) model to stock index prediction by combining an input and a temporal attention mechanism to weight stock features and time steps, respectively. The 2D-Attention-LSTM models are validated in a comparative experiment that involves two attention-based models, namely, a multi-input LSTM (MI-LSTM) and an attention-based two-stage recurrent neural network (Dual-Attention-RNN). Real stock data are used for training and evaluation. The MD-ALSTM model achieves superior performance in stock index prediction on the KOSPI100 dataset compared to MI-LSTM and DARNN. Similarly, Kim et al. [8] developed an attention-based two-dimensional model using a multi-input LSTM for time series forecasting.

As mentioned above, much innovative work has been done on LSTM-Attention models in recent years, but there is a paucity of literature on GAN-based stock price prediction analysis. Carrillo [9] provided a brief comparison of the relative performance of GAN models and ARIMA, ordinary LSTM models, and deep LSTM stock price forecasts. While all models have prediction accuracies close to or above 60%, LSTM 74.16% and GAN 72.68% have higher performance, followed by deep LSTM 62.85% and ARIMA 59.57%. However, it is not the GAN-based LSTM model that predicts stock prices.

A seminal work in some existing literature is [10] based on GANs for stock price prediction. Not only is this one of the first major contributions to GAN-based stock price prediction, but it also provides a novel insight that a hybrid model can be built on in addition to GAN, where the LSTM is used as a data generator, and the MLP is used as a discriminator, to provide higher performance. Similarly, Tovar [11] applied a hybrid GAN model consisting of a bidirectional LSTM and CNN (Bi-LSTM-CNN) for stock price prediction to generate synthetic data using stock market data, such as TSX, SHCOMP, KOSPI 200, and S&P 500, which shows higher performance than a single LSTM model with higher performance.

8.2.2 Long Short-Term Memory

The model we use to predict the closing price of stocks is based on LSTM plus an attention mechanism. LSTM has powerful time series data processing capabilities and has been widely used in many fields. We select the daily data with seven financial factors in the past 20 years to predict the future closing price. The seven financial factors are opening price, closing price, highest price, lowest price, turnover rate, trading volume, and Ma5 (Average closing price over the past five days). Because the above seven financial factors are important in price forecasting, these factors can be used as the seven characteristics of price forecasting. Suppose our input is $\mathrm{X} = \{x_1, \ldots, x_n\}$, which includes daily stock data for n days. Each x_k in X is a vector, and the specific composition is as follows:

$$[x_{k,i}]_{i=1}^{7} = [x_{k,open}, x_{k,close}, x_{k,highest}, x_{k,lowest}, x_{k,turnover}, x_{k,volume}, x_{k,Ma5}] \tag{8.1}$$

The unit of LSTM includes three gates, namely, input gate, forget gate, and output gate. These gates and units can be updated as follows:

$$f_t = \sigma\left(W_f x_t + W_f' \; h_{t-1} + b_f\right). \tag{8.2}$$

$$i_t = \sigma\left(W_i x_t + W_i' \; h_{t-1} + b_i\right). \tag{8.3}$$

$$\tilde{C}_t = \tanh\left(W_c x_t + W_c' \; h_{t-1} + b_c\right). \tag{8.4}$$

$$C_t = i_t \circ \tilde{C}_t + f_t \circ C_{t-1}. \tag{8.5}$$

$$o_t = \sigma\left(W_o x_t + W_o' \; h_{t-1} + b_o\right). \tag{8.6}$$

$$h_t = o_t * \tanh(C_t). \tag{8.7}$$

Here, W_f, W_i, W_c, W_o are the weight parameters and b_f, b_i, b_c, b_o are biases. The forget gate f_t is to decide what information we want to discard from the cell state, and the input gate i_t is to decide what information we want to store in the cell state. The output gate o_t determines what to output, which will be based on the state of the cell. These three gates are an important part of the LSTM unit used to update the current state of the LSTM unit C_t and obtain the new unit output h_t.

8.3 Our Model

In order to make the performance of LSTM more suitable for stock prediction, we introduce a novel architecture of LSTA, which we call attention mechanism-based long short-term memory (AM-LSTM) in Chapter 3. This model gives the LSTM-Attention mechanism and integrates the attention gate introduced in the LSTM unit. This section focuses on LSTM-Attention as a generator and MLP as a discriminator for hybrid GAN analysis.

8.3.1 The Generator

8.3.1.1 Attention Mechanism

Attention Mechanism is a technology that mimics human cognitive attention and is widely used in artificial neural networks. This technology can enhance the weight of neural networks on essential parts of input data and weaken the importance of other parts so that the network can focus more on the most critical parts of the data. Distribution and selection are key functions of the attention mechanism [12, 13]. It accelerates information processing and has efficient information selection and computing power, focusing on crucial tasks [12]. Attention mechanisms have also been widely used in the field of machine translation, such as attention mechanisms that are used to improve neural machine translation (NMT), which selectively focuses on certain parts of the source sentence during translation [14]. Alternatively, Vaswani et al. [15] invoked an attention mechanism model to replace some commonly used recursive and convolutional models. It relies entirely on the attention mechanism to compute the representation of inputs and outputs. Furthermore, in [16], a new approach to traffic sign detection is based on a multi-scale analysis of repeated attention and uses the local environment in the image. The effectiveness of these models using attention mechanisms has been greatly improved. Therefore, we consider to add an attention mechanism to the LSTM cell to improve its performance.

8.3.1.2 The Attention Gate

In this section, for clarity and completeness, we briefly introduce the LSTA model, which combines the attention mechanism with the traditional LSTM. In order to show the model more clearly, we first introduce the attention gate.

Figure 8.1 shows the specific structure of the attention gate. Its position in the whole model is between the forget gate and the input gate of the LSTM to accept the input of the forget gate and the input gate. Equation (8.8) is the updated formula of the attention gate:

$$A_t = \zeta\left(\bar{A}_t[f_t, i_t], \tilde{A}_t[f_t, i_t]\right) = \bar{A}_t \otimes \tilde{A}_t, \tag{8.8}$$

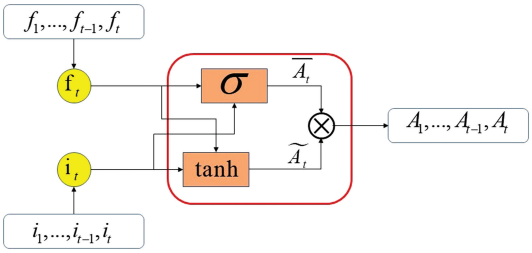


Figure 8.1 The attention gate.

where $\bar{A}_t$ and $\tilde{A}_t$ are defined as follows:

$$\bar{A}_t = \sigma\left(W_{\bar{a}} f_t + W'_{\bar{a}}\, i_t + b_{\bar{a}}\right). \tag{8.9}$$

$$\tilde{A}_t = \tanh\left(W_{\tilde{a}} f_t + W'_{\tilde{a}} i_t + b_{\tilde{a}}\right). \tag{8.10}$$

Here, $W_{\bar{a}}$ and $W_{\tilde{a}}$ are the weight matrix, while $b_{\bar{a}}$ and $b_{\tilde{a}}$ are the bias vectors. The value of $\bar{A}_t$ is calculated by the sigmoid activation function, such as the ratio of attention elements in Equation (8.8). Analogously, the tanh function is employed to compute $\tilde{A}_t$, which can be positive or negative.

8.3.1.3 LSTA

Based on the above gate mechanism, we propose the LSTA model based on LSTM. Figure 8.2 is a schematic diagram of an LSTA unit.

The LSTA receives the input gate of the LSTM and the output of the forgetting gate. In order to update its cell status, we can calculate it as

$$\begin{aligned} \hat{C} &= C_t + A_t \\ &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t + A_t \\ &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t + \tilde{A}_t \otimes \bar{A}_t, \end{aligned} \tag{8.11}$$

where C_t is the output of the original state of LSTM cells and A_t is the output of the attention gate. At this time, we have integrated the attention mechanism into the LSTM model to obtain the LSTA model so that the obtained LSTA model can not only focus on the important information in the sequence but also save the information. Ultimately, it can more accurately predict the closing price of stocks.

Therefore, the final output of LSTA can be updated as

$$h_t = o_t * \tanh(\hat{C}). \tag{8.12}$$

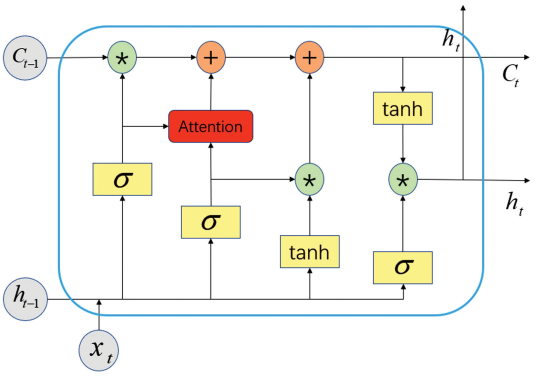


Figure 8.2 The LSTA model.

It should be noted that our attention mechanism is applied inside the LSTM unit and not added after the LSTM unit processes the complete sequence.

8.3.2 The Discriminator

We choose MLP as our discriminator to construct a function d to classify the input data. The cross-entropy loss function is used to optimize the classification process. The discriminator outputs 1 when it receives the real data; otherwise, it outputs 0. We add four hidden layers to the discriminator. The first three hidden layers t_1, t_2, and t_3 contain 72, 100, and 10 neurons, respectively, and all of them use LeakyReLU as the activation function. The final output uses the sigmoid activation function. The data generated by the generator are taken as X_{fake}, and the real data are taken as X_{real}. The output formula of the discriminator is as follows:

$$D(X_{fake}) = \sigma\left(d(X_{fake})\right). \tag{8.13}$$

$$D(X_{real}) = \sigma\left(d(X_{real})\right). \tag{8.14}$$

Among them, $D(X_{fake})$ and $D(X_{real})$ are the outputs of MLP after sigmoid activation function. $d(\cdot)$ denotes the output of MLP. The structure of the discriminator is shown in Figure 8.3.

8.4 Experiments

In this section, we introduce the selection of data sets and data processing and then input the processed data into the model as input features to obtain the final results.

8.4.1 Dataset Descriptions

In order to evaluate the LSTA-GAN model for stock price prediction, we adopt real stock data evaluation models, including the Standard & Poor's 500 (S&P 500 Index), all US stocks traded

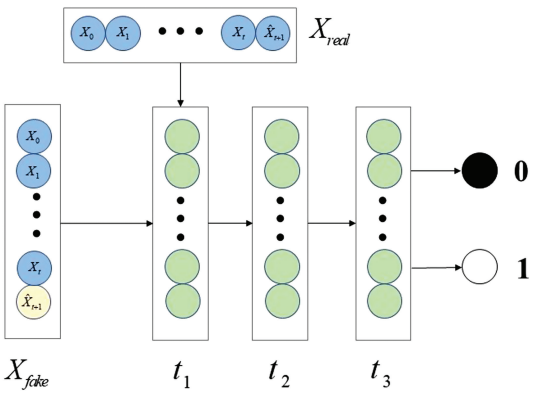


Figure 8.3 The architecture of the discriminator adopts an MLP design.

Table 8.1 Real stock data from NYSE

Name	Trade date	Open price	Highest price	Lowest price	Turnover volume
NYSE	2017/10/09	66.68	66.97	66.62	1794800
NYSE	2017/10/10	66.65	67.60	67.54	1470426
NYSE	2017/10/11	67.63	67.82	66.99	943164
NYSE	2017/10/12	67.20	67.37	66.91	1473459
NYSE	2017/10/13	66.68	66.68	66.12	1711833

on the New York Stock Exchange, NASDAQ, and NYSE markets' complete historical daily price and volume data of exchange-traded funds. We select the data of transaction dates in the past 20 years, but due to restrictions on holidays, transaction dates are not continuous. For example, some stock data are shown in Table 8.1.

We need to standardize these data, which is a crucial step. The data standardization is as follows:

$$x_i = \frac{x_i - \mu^t}{v^t}, \tag{8.15}$$

where μ^t and v^t are the mean and standard deviation of X, respectively. We choose $t = 5$ because we want to use the data of the past five days to predict the data of the next day (e.g., weekend trading is limited). For example, we first calculate the average and standard deviation of the data in the past five days to standardize the data and then use the standardized data to predict the data for the sixth day.

The goal of LSTA-GAN model is to get the closing price of the stock on the next day according to the data of seven factors in the past t days. Based on these data, the generator mines the distribution of real stock data and obtains the closing price from the generated data. We divide the data into two parts: training and testing, of which the first 90% is the training set, and the last 10% is the test set. In the training process, there is an obvious confrontation process between the discriminator and the generator. In the process of confrontation, the generator and discriminator are optimized.

8.4.2 Experimental Results

We evaluate the prediction performance of the model based on statistical regression indicators: mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE), and coefficient of determination (R^2). Among them, the smaller the value of MSE, RMSE, and MAE, the smaller the error of the model, and the closer the predicted value is to the true value; R^2 reflects the proportion of all changes in the dependent variable that can be explained by the independent variable through the regression relationship, and its value closer to 1 indicates a higher degree of fitting the model. It is assumed that the actual closing price and predicted closing price on the day i are y_i and $\hat{y}_i$, respectively. The relevant formulas are as follows:

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i|, \tag{8.16}$$

$$MSE = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2, \tag{8.17}$$

$$RMSE = \sqrt{\frac{1}{\mathrm{m}} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2}, \tag{8.18}$$

$$R^2 = \frac{\sum_{i=1}^{m} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{m} (y_i - \bar{y})^2}, \tag{8.19}$$

where m denotes the number of samples, $\hat{y}_i$ is the predicted value of the model, $\bar{y}$ is the mean value of y_i, and y_i is the real value.

We calculate the average of these evaluation indicators in four datasets as evaluation. In order to evaluate the performance of our proposed method, we use LSTM-GAN, LSTM, WLSTM, and WLSTM-Attention [4] as the baseline model to compare with our proposed model. The results of the comparison are shown in Table 8.2. Among them, boldface results represent the best ones.

Table 8.2 The evaluation result

Method	MAE	MSE	RMSE	R^2
Our Model	**0.0870**	**0.0139**	**0.1179**	**0.9745**
LSTM+GAN	0.1027	0.0187	0.1370	0.9434
WLSTM	0.2470	0.1067	0.3267	0.8965
LSTM	0.2676	0.1208	0.3475	0.8829
WLSTM+attention	0.1935	0.0546	0.2337	0.9621

The best one is highlighted in boldface.

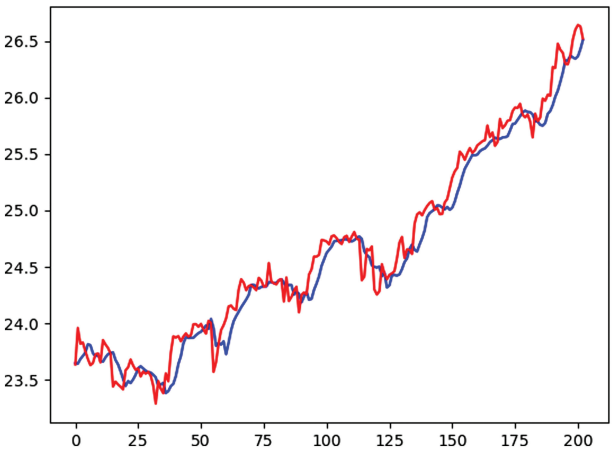


Figure 8.4 Comparison results of price forecast by the LSTM-GAN model and the actual price.

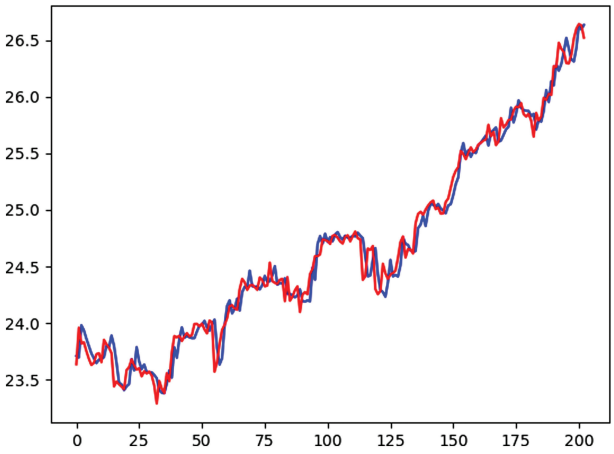


Figure 8.5 Comparison results of price forecast by the LSTM-GAN model and the actual price.

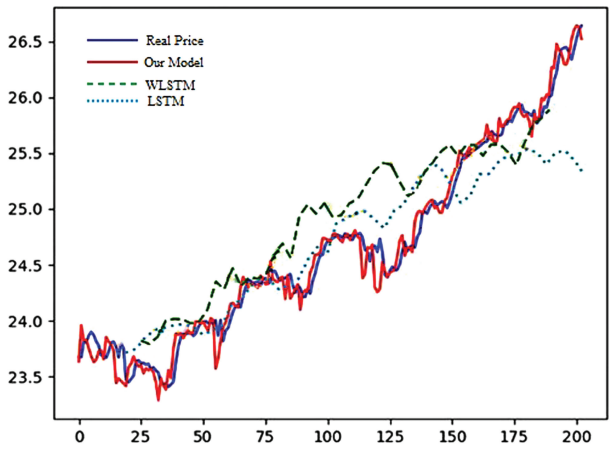


Figure 8.6 Comparison between our model and existing approaches on the stock price prediction.

In order to further analyze the difference between our model and the LSTM model, we train them, respectively, and get the results as shown in Figures 8.5 and 8.6. Among them, the red line represents the real stock price, and the blue line represents the stock price predicted by the model. From the experimental results in Figures 8.4 and 8.5, it can be seen that the fitting degree between the predicted stock price and the real stock price of our model is higher than that between the predicted data of the LSTM model and the real stock price.

Figure 8.6 uses the same training steps to conduct experiments on the dataset. Among them, the red broken line represents the real stock price data, and the blue broken line represents the stock price data predicted by the model. From the changing trend in Figure 8.6, we know that compared with the existing model, our model can achieve the best performance matching the actual price trend line.

8.5 Conclusion

In this chapter, we propose a novel approach to stock prediction by embedding an attention mechanism into an LSTM to form LSTA for stock price prediction. Our contribution is to

propose an attention mechanism that is seamlessly integrated into LSTM units to form a new stock price prediction model, namely, LSTA-GAN based on the GAN architecture. Experiments show that the LSTA-GAN stock price prediction model yields higher performance than other comparative models. In future work, we plan to explore how to extract the financial factors that affect stock prices and the relationship between the relevant stocks and real stock market price formation and optimize our model to obtain higher-accuracy stock price predictions. This work will be carried out in the broader context of introducing deep finance into mainstream economics.

References

[1] Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input lstm. In *Asian Conference on Machine Learning*, 2018.

[2] Li-Chen Cheng, Yu-Hsiang Huang, and Mu-En Wu. Applied attention-based lstm neural networks in stock prediction. *2018 IEEE International Conference on Big Data (Big Data)*, pages 4716–4718, 2018.

[3] Sangyeon Kim and Myung joo Kang. Financial series prediction using attention lstm. *ArXiv*, abs/1902.10877, 2019.

[4] Jiayu Qiu, Bin Wang, and Changjun Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLoS ONE*, 15, 2020.

[5] Xiaolu Wei, Binbin Lei, Hongbing Ouyang, and Qiufeng Wu. Stock index prices prediction via temporal pattern attention and long-short-term memory. *Advances in Multimedia*, 2020:8831893:1–8831893:7, 2020.

[6] Chao Huang, Chuxu Zhang, Peng Dai, and Liefeng Bo. Cross-interaction hierarchical attention networks for urban anomaly prediction. In *International Joint Conference on Artificial Intelligence*, 4359–4365, 2020.

[7] Yeonguk Yu and Yoonjoong Kim. Two-dimensional attention-based LSTM model for stock index prediction. *Journal of Information Processing Systems*, 15:1231–1242, 2019.

[8] Eun Been Kim, Jung Hoon Parka, Yung-Seop Lee, and Changwon Lim. Two-dimensional attention-based multi-input LSTM for time series prediction. *Communications for Statistical Applications and Methods*, 28(1), 39–57. 2021.

[9] Ricardo Alberto Carrillo. Generative adversarial network for stock market price prediction, 2019.

[10] Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock market prediction based on generative adversarial network. In *International Conference on Identification, Information, and Knowledge in the Internet of Things*, 2018.

[11] Wilfredo Tovar. Deep learning based on generative adversarial and convolutional neural networks for financial time series predictions. *ArXiv*, abs/2008.08041, 2020.

[12] Posner, Michael I. Cognitive neuroscience of attention. Guilford Press, 2011.

[13] Marco Wischnewski, Anna Belardinelli, Werner X. Schneider, and Jochen J. Steil. Where to look next? Combining static and dynamic proto-objects in a tva-based model of visual attention. *Cognitive Computation*, 2:326–343, 2010.

[14] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *ArXiv*, abs/1508.04025, 2015.

[15] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[16] Yan Tian, Judith Gelernter, Xun Wang, Jianyuan Li, and Yizhou Yu. Traffic sign detection using a multi-scale recurrent attention network. *IEEE Transactions on Intelligent Transportation Systems*, 20:4466–4475, 2019.

Contributors

Guoqiang Zhong received his PhD in Pattern Recognition and Intelligent Systems from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, in 2011. Between October 2011 and July 2013, he was a postdoctoral fellow with the Synchromedia Laboratory for Multimedia Communication in Telepresence, University of Quebec, Montreal, Canada. Since March 2014, he has been an associate professor and then a full professor at the College of Computer Science and Technology, Ocean University of China, Qingdao, China. He has published 5 books and more than 100 technical papers in the areas of artificial intelligence, pattern recognition, and machine learning. His research interests mainly focus on deep learning and its applications. He has served as chair/PC member/reviewer for many international conferences and top journals, such as *IEEE TNNLS*, *Pattern Recognition*, *ACM TKDD*, *AAAI*, and *CVPR*. He has been awarded an outstanding reviewer by *Pattern Recognition*, *Knowledge-Based Systems*, *Neurocomputing*, and *Cognitive Systems Research*. He has won the Best Paper Award of BICS2019 and the APNNS Young Researcher Award. He is a senior member of CCF, CAAI, and CSIG; member of ACM, IEEE, IAPR, and APNNS; professional committee member of CSO-AIO, CAAI-PR, CAA-PRMI, and CSIG-DIAR; and standing trustee of the Shandong Association of Artificial Intelligence.

Jinxuan Sun is currently a PhD student at the Ocean University of China. Her research interests include deep learning and image super-resolution.

Index

S

T

U

V

W

Y

Guoqiang Zhong
Jinxuan Sun
Editor

Attention Augmented Learning Machines

Theory and Applications

nova
science publishers
www.novapublishers.com

ISBN-13: 979-8-89113-161-3