学 位 論 文

Study of Transfer Learning on medical information processing by explainable artificial intelligence

（説明可能な人工知能による医療情報の転移学習に関する研究）

張　洪健

北海道大学大学院保健科学院
保健科学専攻保健科学コース

２０２３年度

Index

Chapter 1 Background

1.1 Development of Artificial Intelligence

In 1956, McCarthy, Minsky, and other scientists met at Dartmouth College in the U.S. and put forward the concept of "Artificial Intelligence" for the first time, marking the birth of the discipline of Artificial Intelligence and opening the golden age of Artificial Intelligence. The world's first chat program, ELIZA, was born in this stage, which can match patterns according to set rules based on the user's questions, and then select the appropriate answer from a library of pre-written answers. In 1959, the chess program written by Arthur Samuel beat the then chess master Robert Nerai, and this program will search for all the possible jumps and find the best way to do so. "Reasoning is searching" was one of the main lines of research during this period. During the Golden Age, McCarthy developed the LISP language, which has become the dominant programming language in artificial intelligence for decades. Multilayer neural networks and back-propagation algorithms began to appear, and expert systems began to take off. However, as scientists overestimated the speed of scientific development, too much optimism sparked worldwide skepticism about AI when promises could not be fulfilled on time. In 1973, mathematician Rathir presented a study that pointedly pointed out that the seemingly grandiose goals of AI were simply unattainable. Artificial intelligence fell into its first winter in the 1970s. The biggest problem that arose during the Golden Age was the lack of arithmetic power. Many artificial intelligence scientists have found that mathematical reasoning, algebraic geometry, and human intelligence, computers can be solved with very little computational power, while for image recognition, sound recognition, and free movement, such as human beings who do not need to use their brains, relying on instinct and intuition to complete the task, the computer needs a huge amount of computation. This conclusion, on the one hand, makes people doubt the effectiveness and practicality of early neural networks; on the other hand, it also leads to the development of artificial intelligence technology in the direction of utilitarianism and pragmatization. Neural network technology, once favored, relied excessively on computational power and the amount of empirical data, and thus, did not make substantial progress for a long time.

In 1978, Carnegie Mellon University developed a program XCON that could help customers automatically select computer accessories and put it into use in 1980, which was a complete expert system containing more than 2,500 rules set, with an accuracy of more than 95%, and became a milestone in the new era. Expert systems began to exert power in specific fields and led the entire artificial intelligence technology into a boom phase. An expert system is a set of computer software that tends to focus on a single area of specialization, simulating human experts answering questions or providing knowledge to help staff make decisions. On the one hand, it requires human experts to organize and enter a huge knowledge base (expert rules); on the other hand, it needs to write a program that sets up how to reason based on the questions asked to find the answers, also known as an inference engine. Expert systems are limited to a small range, avoiding the various difficulties of general-purpose AI. However, expert systems are limited in their ability to achieve, are unable to self-learn and update the algorithms of their knowledge base, and are becoming increasingly cumbersome and costly to maintain. In addition, the problems of expert systems, such as narrow domain, lack of common sense knowledge, difficulty in knowledge acquisition, single reasoning method, lack of distributed functionality, and difficulty in compatibility with existing databases, were gradually exposed. Thus, it entered the second winter of artificial intelligence. Nevertheless, some important achievements have been made during this period. 1988, Judea Peel introduced probabilistic statistical methods into the reasoning process of artificial intelligence, which had a significant impact on the subsequent development of artificial intelligence.

Between 1993 and 2011 artificial intelligence entered a steady development phase. Due to the development of network technology, especially Internet technology, the innovative research of AI was accelerated, which prompted AI technology to move further towards practicalization.In 1997, IBM Deep Blue supercomputer defeated Kasparov, the world champion of chess.After 2008, with the outbreak of mobile Internet and cloud computing technology, a historically unimaginable amount of data was accumulated, which provided sufficient data for the subsequent development of AI. After 2011, owing to big data and improved computer skills, deep learning has been a huge success, with accuracy rates exceeding 90%.

1.1.1 History of Machine Learning

Artificial Intelligence belonged to the developmental period from the 1950s to the 1970s, and the dominant technique used was deductive reasoning using symbolic knowledge-based representation. From the mid-1970s to the 1980s, this belonged to the knowledge period, and the dominant techniques were based on symbolic knowledge representation and building expert systems by acquiring and utilizing domain knowledge. The learning period began in the 1980s, with the dominant techniques being symbolist learning and neural-network-based connectionist learning.

The neural network algorithm, named for a structure that mimics a human neural network, learns by training on data, allowing the construction of a neural network that can achieve classification. However, this algorithm was abandoned in the 1970s because of the problem of vanishing or exploding gradients as the number of layers increased as well as the large amount of data and arithmetic power required, leading to poor classification accuracy. Since 2011, significant advances in computer power and data volume, as well as improvements in algorithms, have led machine learning into the era of deep learning, which has resulted in classification accuracies of more than 90%, exceeding human discrimination levels. Although deep learning, or neural network algorithms, has come a long way away, there is still an unavoidable problem: neural network algorithms are black-box algorithms. It is difficult to determine the basis of a neural network program's judgments, and it is impossible to modify the reasons for those judgments. However, in the medical field, interpretability is a very important criterion that makes it difficult to apply deep learning directly to medical inferences.

Instead, expert systems based on rules and knowledge bases became more popular in the 1980s. Expert systems rely on rules and knowledge bases manually entered by experts in various fields to reason and provide advice. Therefore, expert systems can avoid the problems of neural-network-based algorithms, have better interpretability, and can answer the rules on which the judgment is based. However, expert systems cannot self-update their knowledge base and are expensive to maintain. Moreover, there are many rules that are difficult for humans to organize and put into a knowledge base. In addition, the rules that are available are in precise ranges and do not respond appropriately to data.

Decision tree algorithms in machine learning have appeared since the 1960s and 1970s with the ID3 algorithm proposed by J Ross Quinlan. It can be learned by training the data and extracting rules to make inferences on new data. However, the decision tree algorithm is still used in the determination of the exact number, and it is difficult to deal with some fuzzy concepts; at the same time, the decision tree algorithm's robustness is poor, and it is easy to overcome problems. The Random Forest algorithm, a group learning method for decision trees, solves the phenomenon of overfitting that ordinary decision trees are prone to and increases the accuracy. This algorithm was developed and deduced by Leo Breiman and Adele Cutler and was derived from a Randomized Decision Forest proposed by Tin Kam Ho in 1995. The Random Forest algorithm solves the overfitting problem but reduces the interpretability of the algorithm.

1.2 AI applicated in medical field

The impact of AI has been widely discussed in the medical literature[1]–[3] . AI can be used to develop sophisticated algorithms that "read" features from large amounts of healthcare data and then use the knowledge gained to aid clinical practice. AI can also be equipped with learning and self-correcting features to improve accuracy based on feedback. By presenting the most up-to-date medical knowledge from journals, manuals, and specialized programs to provide recommendations for effective patient care, AI devices[4] can support clinical decision-making. Furthermore, in human clinical practice, AI systems may help reduce unavoidable medical and therapeutic errors (i.e., be more objective and repeatable) [2]. In addition, AI systems can process valuable knowledge gathered from large patient populations to help make real-time inferences about health risk warnings and health outcome estimates[5].

With the recent re-entry of AI into scientific and public consciousness, there have been new breakthroughs in AI in healthcare, with clinical environments filled with new

AI technologies at a breakneck pace. Healthcare has been described as one of the most exciting applications of AI. Since the mid-20th century, researchers have suggested and built several systems for supporting clinical decisions . Since the 1970s, rule-based approaches have achieved many successes and have been recognized as theories for interpreting electro-cardiograms, identifying diseases, selecting optimal therapies , providing scientific-logical explanations, and assisting physicians in formulating diagnostic hypotheses and challenging patient cases. However, rule-based systems are expensive and unstable because they require explicit expression of decision rules and, like any textbook, manual revision. Additionally, higher-order interactions between various pieces of information written by different experts are difficult to encode, and the efficiency of the structure is limited by the comprehensive nature of a priori medical knowledge [6]. It is also difficult to incorporate an approach that combines deterministic and probabilistic reasoning procedures to narrow the appropriate psychological context, prioritize medical theories, and prescribe treatments [7].

The recent rebirth of Artificial Intelligence is largely due to the aggressive implementation of deep learning - including the training of multi-layer artificial neural networks (i.e., deep neural networks) on large datasets - to access a wide range of labeled data sources . Existing neural networks are becoming increasingly deeper, typically greater than 100 layers. Multilayer neural networks can model complex interactions between inputs and outputs but may also require more data, processing time, or advanced architectural design to achieve better performance. Modern neural networks typically have tens to hundreds of millions of parameters and require significant computational resources for model training [8]. Fortunately, recent developments in computer processor architectures have empowered deep learning with the required computational resources. In general, deep learning-based AI algorithms have been developed for image-based classification , diagnosis prognosis , genomic interpretation , biomarker discovery , monitoring via wearable life-record devices , and automated robotic surgeries to enhance digital healthcare [9]. Rapid advances in artificial intelligence have made it possible to use aggregated health data to generate powerful models that automate diagnostics and enable increasingly precise medical approaches through the timely and dynamic customization of treatments and targeted services with optimal efficacy.

Although AI promises to revolutionize the practice of medicine, a number of technical barriers still exist. Because deep-learning-based approaches rely heavily on the availability of large amounts of high-quality training data, care must be taken to collect data that are representative of the target patient population. For example, data from a variety of healthcare settings, including different forms of bias and noise, may result in models trained on data from one hospital not generalizing to another. In the case of diagnostic roles with incomplete inter-expert agreement, consensus diagnosis has been shown to significantly improve the training efficiency of deep learning-based models [10]. Proper data management is important for managing heterogeneous data. However, obtaining a high-quality gold standard for identifying a patient's clinical status requires physicians to independently and potentially review their clinical results repeatedly, which is prohibitively expensive on a population scale.

The most recent advances in neural networks have been limited to well-defined activities that do not require data integration across multiple modalities. Methods for applying deep neural networks to general diagnosis and treatment planning are not straightforward. Although deep learning has been effective in image detection, translation, speech recognition, sound synthesis, and even automated neural architecture search, clinical diagnostic and therapeutic tasks tend to require more attention (e.g., to patient interests, beliefs, social support, and medical histories) than a limited number of tasks in which deep learning typically excels. In addition, it is not clear whether transfer learning methods can translate models learned from a wide range of non-medical datasets into algorithms for studying multimodal clinical datasets. This suggests that more comprehensive data collection and annotation activities are needed to build end-to-end clinical AI programs.

The design of computing systems for processing, storing, and exchanging EHRs and other critical health data remains an issue [11]. Privacy-preserving approaches such as federated learning can allow secure sharing of data or models across cloud providers . However, creating interoperable systems that meet the requirements of clinical knowledge representation is important for the widespread adoption of such technologies. However, deep and seamless data integration across healthcare applications and locations remains problematic and inefficient. However, new software interfaces for clinical data are beginning to be widely adopted by multiple EHR providers, such as alternative healthcare applications and reusable technologies on rapid health interoperability resource platforms. Most AI previously developed in healthcare applications has been performed using retrospective data for proof-of-concept. Prospective studies and clinical trials are needed to evaluate the efficiency of developed AI systems in clinical settings to validate the usefulness of these healthcare AI systems in the real world. Prospective studies will help recognize the vulnerability of AI models in real-world heterogeneous and noisy clinical environments and identify ways to integrate medical AI into existing clinical workflows.

AI in medicine ultimately leads to safety, legal, and ethical challenges owing to medical negligence caused by complex decision support structures [12] and must face regulatory barriers. In the event of a medical malpractice lawsuit involving the application of medical AI, the judicial system will continue to provide specific instructions regarding which agency is responsible. Healthcare providers with medical malpractice insurance must be clear about coverage as healthcare decisions are made in part by AI programs. As automated AI is deployed for specific clinical activities, standards for diagnostic, surgical, support, and paramedical tasks will need to be revised. The functioning of healthcare practitioners will begin to change as different AI modules are implemented in the quality of care, and deviations will need to be minimized while patient satisfaction must be maximized .

1.3 Explainable AI

Currently, deep learning tools are capable of producing extremely reliable results; however, they are often highly opaque and their behavior is difficult to understand, if not completely invisible. Even for skilled experts, it can still be difficult to fully understand these so-called " black-box models. With the widespread use of these deep-learning tools, researchers and policymakers can question whether the accuracy of a given task outweighs more important factors in the decision-making process.

As part of attempts to incorporate ethical standards into the design and implementation of AI technologies, policy discussions around the world increasingly address the need for some form of trustworthy AI, including Effective AI, Responsible AI, Privacy-Preserving AI, and Explainable AI (XAI), where XAI seeks to address fundamental questions about the fundamentals of the decision-making process, both in humans. As AI approaches are used to solve problems in a variety of complex policymaking areas, as experts increasingly work with AI-enabled decision-making tools, such as in clinical research, and as people experience AI more often, these discussions become more pressing when decisions have significant impacts on real-life systems. Meanwhile, research on AI continues to advance steadily. XAI is a dynamic field, with many ongoing studies emerging, and a number of new strategies that have a huge impact on the development of AI in a variety of ways.

Although the term is used inconsistently, "XAI" refers to a class of systems that provides insight into how AI systems make decisions and predictions. XAI explores the reasoning behind the decision-making process, demonstrates the strengths and weaknesses of the system, and provides a glimpse into how the system will function in the future. By providing easy-to-understand explanations of how AI systems conduct research, XAI allows researchers to understand the insights that come from the research results. For example, interpretable agent modules can be added to the learning model to achieve a more transparent and trustworthy model. In other words, for traditional machine or deep learning models, only the generalization error is considered, while the addition of an interpretable agent allows for both the generalization error and human experience to be considered, leading to validated predictions. By contrast, a learning black-box model without an interpretable agent module raises end-user concerns, even though the performance of the learning model may be high. Such black-box models always cause confusion, e.g., "Why did you do that?" , "Why don't you do this?" , "When do you succeed or fail?" , "How do I correct my mistakes?" and "Can I trust predictions?" However, XAI-driven models can also be used to make predictions. On the other hand, XAI-driven models can provide clear and transparent predictions to ensure "I understand why." , "I understand why not." "I know why you succeed or fail." , "I know how to correct a mistake." and "I understand, therefore I believe." A typical feedback loop for XAI development includes seven steps: training, quality assurance

(QA), deployment, prediction, split testing (A/B testing), monitoring, and debugging.

Various terms are used in research, public, and policy debates to define certain desired characteristics of XAI systems, including:

- Explainability: This implies an understanding of how AI technology works.
- Interpretability: This explains how decisions are made for a wider range of users.
- Transparency: This measures the level of accessibility of data or models.
- Legitimacy: This demonstrates an understanding of the case that supports a particular outcome.

In a broad sense, XAI can be categorized as a model-specific or model-agnostic method. Furthermore, these methods can be categorized as local or global and can be intrinsic or post hoc[13]. Essentially, many machine learning models are inherently interpretable, such as linear models, rule-based models, and decision trees, also known as transparent or white-box models. However, the likelihood performance of these relatively simple models is relatively low. For more complex models, such as Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Integration Models, we can design model-specific and post-hoc XAI strategies for each of them. Common strategies include simplified interpretation, architectural modification, feature relevance interpretation, and visual interpretation[13]. Obviously, these more complex models can achieve better performance, while interpretability decreases.

Recently, great attention has been paid to model agnostic based approaches that rely on simplified proxy functions to interpret predictions[14]. Model-agnostic methods are not attached to a specific machine-learning model. In other words, such techniques distinguish between predictions and interpretations. Model agnostic representations are usually post hoc and commonly used to explain deep neural networks with interpretable agents that can be local or global [15].

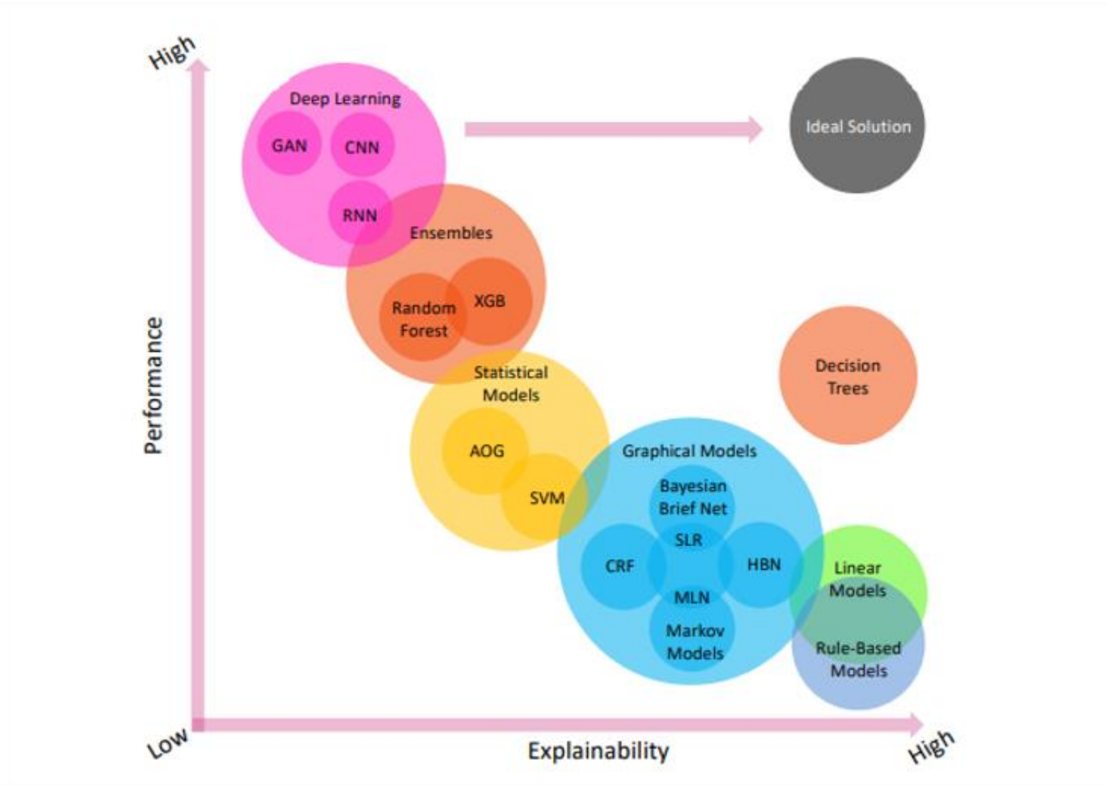


Figure 1-1. performance and explainability of often used models

1.3.1 Difference between XAI(explainable artificial intelligence) and IAI(interpretable artificial intelligence)

Although XAI and IAI have been conflated in many studies, they have been distinguished in the literature in various ways.[16], [17]

The XAI outlines why the choice was made, but does not describe the process by which the decision was made. The IAI, on the other hand, outlines how the decision was made, but does not explain why the criteria were justified.

Explainability means that a model and its results can be explained in a way that is understandable to humans. XAI allows for the explanation of the learning model and analyzes its logical flow by focusing on why the system made the given conclusions.

Interpretability allows users to understand the results of a learning model by revealing the rationale for their decisions.

1.4 XAI in medical field

Although deep-learning-based AI technologies will be used in a new era of digital healthcare, challenges remain. XAI can play a key role in development, potentially solving small-sample learning problems by filtering out clinically meaningless features. In addition, many high-performance deep learning models produce unaided discoveries that are incomprehensible to humans. While these models can produce higher efficiencies than humans, it is not easy to express intuitive explanations that justify model results, define model uncertainty, or derive additional clinical insights from these computational "black boxes." In deep learning models, it can be tricky to understand what the model sees in clinical data, such as radiographic images. For example, research investigations have explicitly stated that being a black box is a "strong limitation" of AI in dermatology, as it does not allow for personalized assessment by qualified dermatologists that can be used to clarify clinical facts[18]. This black-box design presents an obstacle to validate developed AI algorithms. It is necessary to demonstrate that a high-performance deep learning model actually recognizes the appropriate regions of an image and does not overemphasize unimportant findings. Recent methods have been developed to characterize AI models, including visualization methods. Some widely used levers include occlusion maps[19], saliency maps[20], class activation maps [21], and attention maps [22]. Since the output is an image, localization and segmentation algorithms can be more easily interpreted. However, model understanding is still much more difficult for deep neural network models trained on non-imaging data rather than images, which is an open question for current ongoing research efforts[23].

Deep learning-based AI approaches are becoming increasingly popular in medicine, with extensive work in automated triage, diagnosis, prognosis, treatment planning, and patient management[8]. We can find many open problems in the medical field that have inspired clinical trials utilizing deep learning and AI approaches. However, in the medical field, interpretable problems are far from theoretical developments. More precisely, interpretability in the clinical domain includes factors not considered in other domains, including risk and liability [24]. Lives may be at stake when making medical responses, and it would be irresponsible to leave these critical decisions to AI algorithms without interpretability and accountability . In addition to the legal issues, this is a serious vulnerability that could have catastrophic consequences if used maliciously.

Thus, several recent studies have been devoted to exploring interpretability in medical AI. More specifically, specific analyses have been investigated, such as chest radiographs, sentiment analysis in medicine, and COVID-19 detection and classification, which encourages an understanding of the importance of interpretability in the medical field. Furthermore, some study[25] argues that a certain degree of opacity is appropriate, it is more important for us to provide empirically validated and reliable findings than to obsess too much about how to unravel the black box.

One obvious approach to XAI that many researchers have adopted is to provide interpretability to their predictive models. These approaches rely heavily on keeping less complex AI models interpretable while improving their performance through refinement and optimization techniques. However, such model optimization is not always straightforward or a trivial task.

1.5 Development of Natural Language Processing (NLP)

The subject of NLP broadly consists of two main steps: firstly, the representation of the input text (raw data) into a numerical format (vectors or matrices), and secondly, the design of modeling tasks used to process the numerical data in order to achieve a desired goal or objective. The first part of this is known as word embedding and is broadly categorized into rule-based, statistical and neural network-based approaches.

In manual-based techniques, rules and features are derived by experts, e.g., trees, graphs, grammar rules, etc. Statistical and mathematical formulas are used to derive features. In neural networks, neural models automatically learn features which are categorized as context sensitive, context insensitive and pre-trained embeddings.

The field of NLP started with exact matching techniques in which context-independent grammars (CFG) are used for analysis. Search engines are mainly based on complex nested if-then rules represented by CFG. Further advances led to approximate matching, in which errors reaching a specific threshold are ignored. However, due to the ambiguity of natural language, using such techniques and designing complex rules can be more convenient, time-consuming and error-prone. Subsequently, more statistical methods followed these pattern matching approaches.

As research evolved, statistical methods were sought to focus on the frequency of words. Various techniques such as One-Heat Encoding (OHE), Bag of Words (BoW), Word Frequency (TF), Inverse Document Frequency (IDF), etc. fall into this category. These techniques are easy to implement and improve the accuracy of the model compared to grammar-based techniques. However, such representations suffer from dimensionality catastrophe and their performance suffers on large-scale datasets due to computational power limitations. Therefore, they need to be more scalable and their use is mainly limited to small-scale datasets.

Dimensionality Reduction Techniques (DRT) are used to overcome the curse of dimensionality problem. In these simplification techniques, the key idea is to find a new space or coordinate system in which the input data can be represented with fewer features, less information loss, and minimal error. Usually, two techniques are used to reduce the size. The first method is feature selection and the other is feature transformation. In feature selection, top-k items or features are selected based on certain criteria or thresholds, while the rest are simply discarded. In feature transformation, a linear or nonlinear transformation is applied to the dimensions of the original dataset and it is mapped to a lower space with fewer dimensions. In both methods, care is taken not to lose any information and to preserve or capture what is in the original data. This discrete representation does help in classification and categorization applications. However, it performs poorly for tasks such as information retrieval, chatbots, machine translation, and language generation, which require a deeper understanding of polysemous words or recognizing concepts such as analogies, synonyms, and antonyms. These limitations are represented by state-of-the-art representations, called word vectors, that are derived using neural networks.

Embeddings derived from neural networks are mapped to continuous vector spaces where each word vector is represented by an array of real numbers. In embeddings derived from such continuous vector spaces, the focus is more on computing the semantics of individual words by looking at the context in which they occur. The technique considers the influence of neighboring words on a given word and how these relationships affect the meaning of the word. These new word vectors are contextually relevant and can recognize synonyms and antonyms and construct word analogies and categories, which was not possible in earlier approaches. Word vectors capture the meanings of words (literal and implicit) and represent them using dense floating-point values. They represent semantic and syntactic aspects of words. They are typically between 100 and 500 dimensions long.

In addition, earlier methods required labeling of input data, while the derivation model handles the word vector unlabeled data in a self-supervised manner. However, more training data is needed to improve the accuracy of word vectors. In these word vectors, the rows are the number of words in the corpus and the columns are the number of categories, concepts, qualities, etc., where each feature represents some aspect of the word. For example, for a word such as "city", the important aspects may be friendliness, criminality, sociability, economy, etc. The word vectors can be used for a variety of purposes. Because of its precision, if a word embedding vector is added or subtracted, a new word vector is created, which implies something semantically. If such a "synthetic" vector is plotted on a continuous vector space, it will point to or be close to the embedding of a word. Thus, such word embeddings can be used to predict the words around them and capture the relationships between them.

1.6 Purpose

Due to the black-box problem of deep learning AI, this study proposes a method to provide interpretability to medical text processing AI models with greater simplicity compared to past text interpretability methods. It makes it possible to increase the use of AI on medical text processing tasks to draw informed conclusions without adding too much labor consumption for building AI interpretability.

Chapter 2. Methodology

2.1 Multi-Layer Perceptron (MLP)

MLP is the simplest neural network. A multilayer perceptron consists of multiple layers of neurons, compared to a linear model, a multilayer perceptron achieves model fitting to nonlinearities by adding at least 1 layer of hidden layers and activation functions between the input and output layers. The basic structure is shown below in Fig. 1 for an MLP that contains only 1 hidden layer. where the layer consisting of nodes that are connected to each input and each output is called a fully connected layer. The loss function is constructed by comparing the difference between the output of the neural network, i.e., the predicted result, and the actual result, as a measure of the model's performance during the training process. The smaller the value of the loss function, the better the model. The training process of a neural network model is to minimize the value of the loss function by constantly updating the parameters in the model (automatically or manually) to maximize the model performance. In the training process of neural networks, the parameters that need to be adjusted manually are called hyperparameters, which need to be determined before a training session is conducted, such as the learning rate, the number of epochs, the batch size, and the loss function used. The parameters that are constantly updated during the training process are the weight matrices between the layers.

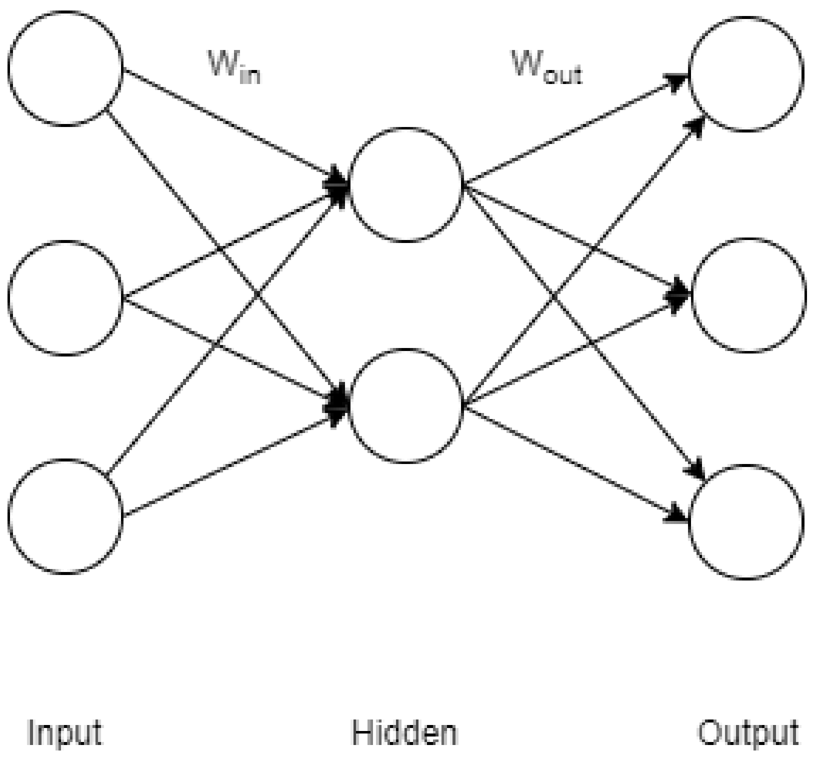


Figure 2-1 structure of MLP

2.1.1 Loss Function

The loss function is used to describe the gap between the predicted and actual results of a model. The training objective of machine learning is to find the weights W and deviations B to minimize the loss function, and this minimization is usually done using gradient descent. Forward propagation and back propagation are used in the training process, where the loss is calculated using forward propagation and the weight matrix is updated by back propagation of the gradient. The commonly used loss functions are cross entropy, mean square error etc.

2.1.2 Activation Function

Activation functions are a class of nonlinear functions that are applied to the output results of each hidden layer to make the model nonlinear. Commonly used activation functions are Sigmoid, tanh, ReLU and so on. Each activation function is shown below.

2.2 Convolutional Neural Network (CNN)

Image data samples are composed of a two-dimensional data grid, where a pixel point is one or more data (depending on the image format), and its data magnitude is larger and spatially informative compared to the usual tabular data. Convolutional Neural Networks CNNs are a powerful class of models designed to process image data. Models based on the architecture of convolutional neural networks have dominated the field of computer vision, and almost all applications of image recognition today are based on this approach. CNNs are able to utilize the spatial structure information of the image as compared to MLPs, while the convolutional structure is able to drastically reduce the amount of parameter computation as compared to the fully connected structure.CNN networks have a convolution layer and a pooling layer. The convolution layer uses the convolution kernel to correlate with the pixel information of the picture, and by moving the convolution kernel on the picture, it obtains the local spatial information of each position of the picture, and at the same time realizes the compression of the picture information. The pooling layer gradually applies the pooling kernel on the basis of the convolutional layer to calculate the maximum or tie value of all elements in the pooling window. The role of pooling layer is to reduce the sensitivity of the convolutional layer to the position, and at the same time reduce the sensitivity to the spatial downsampling. The basic calculation of the CNN structure is shown below.

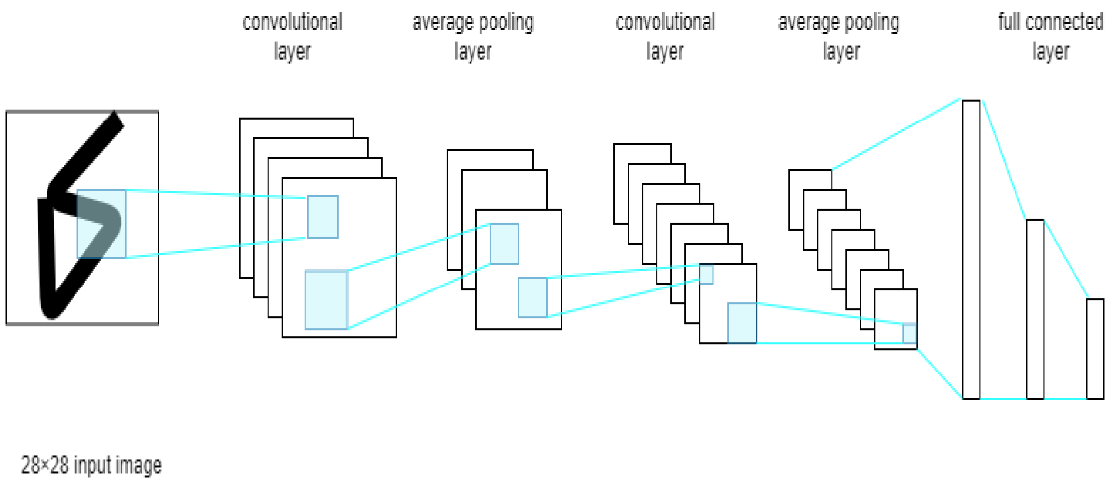


Figure 2-2 structure of LeNet

The figure above illustrates the network structure of LeNet, one of the first Convolutional Neural Networks released, which was proposed in 1989 to recognize handwritten digits in images[26]. This paper uses this to illustrate the CNN structure.

2.2.1 ResNet

The ResNet network architecture was proposed in 2015 by Kaiming He et al[27]. It won the ImageNet image recognition competition in 2015 and profoundly influenced the design of subsequent deep networks. The core idea of residual networks is that each additional layer should more easily contain the original function as one of its elements. Due to the introduction of residual blocks, the propagation speed of residual networks as well as their effectiveness for overfitting problems has improved favorably.

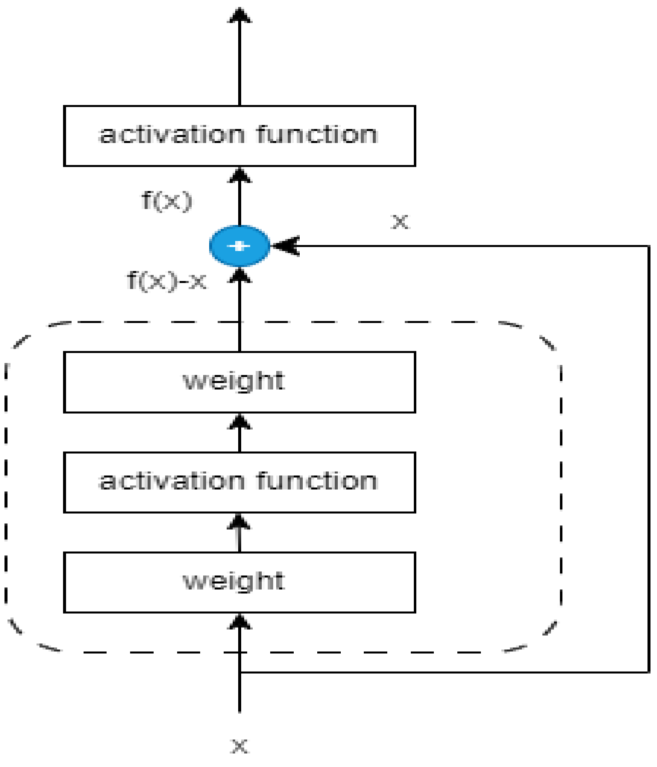


Figure 2-3 ResBlock

2.3 Class Activation Mapping (CAM)

CAM is an interpretable method for computer vision. It extracts the feature map output from the last convolutional layer of the network, and after weighted superposition of its channels, the region where its activation values are located is the region where the objects in the image are located. By overlaying this superimposed feature map on the input image, it is possible to intuitively know where the model is focusing its attention when performing classification[28]. However, CAM is only applicable to network architectures that have a globally averaged pooling layer and only one fully connected layer (the output layer). This led to the Grad-CAM method, which has two key elements, 1) an output feature map of the last convolutional layer after the input image has been processed by the CNN, and 2) a number of weights associated with the input categories that are consistent with the number of feature map channels[29]. The category saliency map can be obtained by multiplying and adding the two correspondingly. For the network structure containing multiple fully connected layers, CAM cannot be applied because it does not satisfy the second point, and if we intend to use CAM, we need to modify the network structure, which not only requires additional operations, but also has an impact on the model effect.Grad-CAM, on the other hand, chooses to use the gradient information of the feature maps as the weights, so that not only does it not need to make additional modifications to the network structure, but also can be applied to all the CNNs. Grad-CAM uses the gradient information of the feature map as weights, which makes it possible to apply it to all CNN network structures without any additional modification.

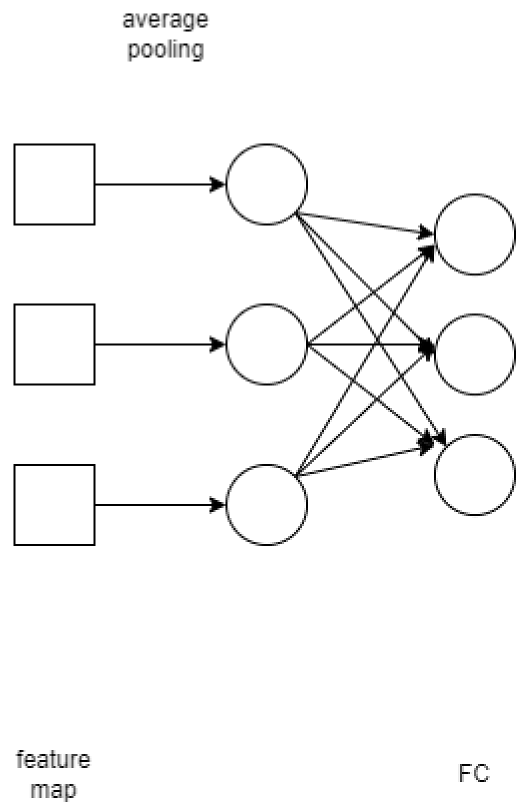


Figure 2-4 structure which can use CAM

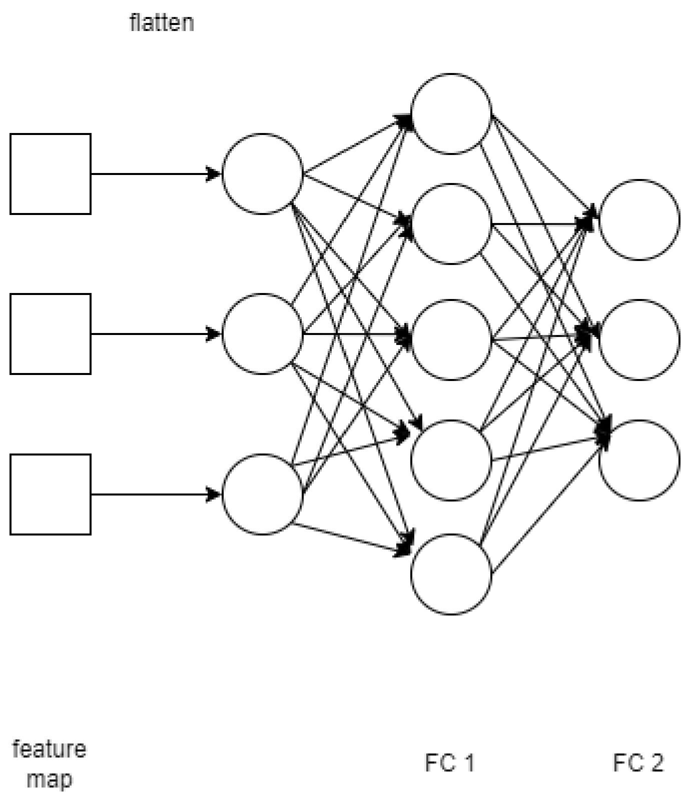


Figure 2-5 structure which can not use CAM

The CAM principle is shown in Fig. The feature map output from the last convolutional layer of the CNN is three-dimensional:[C,H,W], Let the k channel of the feature map can be denoted as $f_k(x,y)$, where x,y are the indexes on the width and height dimensions respectively. If the last convolutional layer connects a global average pooling layer, and then a fully connected output classification result, the computation process of the confidence score (before Softmax mapping) from the output feature map of the last convolutional layer to the c category in the output layer can be expressed as:

$$S_c = \sum_k w_k^c \sum_{x,y} f_{k(x,y)} = \sum_{x,y} \sum_k w_c f_k(x,y) \quad (1)$$

Indicates that each output channel of the feature map is first averaged to a value, and each channel gets a value, and then these values are weighted and summed to get a number, which is the confidence score of the first class, characterizing the likelihood that the class of the input image is the size.

Indicates that each channel of the feature map is first weighted and summed to obtain a two-dimensional feature map (channel dimension collapse), and then this two-dimensional feature map is averaged to obtain the confidence score of the first class.

From the derivation of Eq. (1), it can be seen that the global average pooling of the feature map, followed by weighted summation to obtain the confidence score of the category, is equivalent to the weighted summation of the channel dimensions of the feature map, followed by the global average pooling.

After this equivalent transformation, it highlights the importance of the channel weighted sum of the feature map: on the one hand, the channel weighted sum of the feature map directly encodes the category information; on the other hand, and most importantly, the channel weighted sum of the feature map is two-dimensional, and also retains the spatial location information of the image. We can observe the relationship between the relative position information in the image and the category information encoded by the CNN through visualization methods. Here the channel weighted sum of the feature map is called the category activation map.

2.4 NLP

2.4.1 word2vec

word2vec is a word embedding method proposed by Google's miklov in 2013[30]. Word embedding is the representation of words as vectors, which now belongs to the basic content of natural language processing. Word embedding includes two realization modes: one-hot and dense. In one-hot mode, for a dictionary of size N, vectors of length N will be generated, and the word vector of each word in the article is the vector with the corresponding position 1 in the dictionary and the rest of the positions 0. The solo heat pattern is easy to construct, but the length of the vectors constructed by it increases with the size of the dictionary, and the consumption required for computation is high, and at the same time, the word vectors constructed by the solo heat pattern cannot accurately express the similarity between different words. Dense patterns solve these problems. word2Vec is a class of dense representation patterns. word2vec is a self-supervised method, which contains two models, skip-gram and continuous bag-of-words CBOW (Continuous Bag-Of-Words). the CBOW model predicts the center word by inputting the peripheral words, and Skip- Gram, on the contrary, predicts the possible surrounding words by inputting the center word.

2.4.1.1 Continuous Bag Of Word(CBOW)

The data used to train the CBOW model are word vectors of words that are the context of a particular word [8], and the output is the word vector of that particular word. For example, if the context range is 4 and the specific word is "learning," there are 8 context words (4 before and 4 after) (Figure 2-6). The word vectors of these 8 words are used as input, and the word vector of a specific word is the target word.

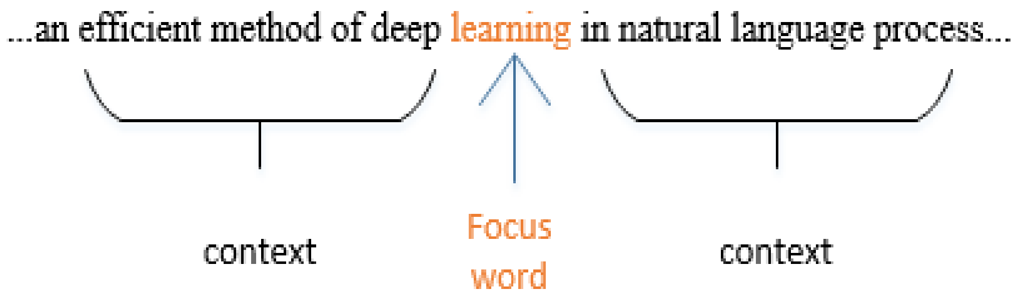

...an efficient method of deep learning in natural language process...
context
Focus word
context

Figure 2-6 target word and context

In the CBOW example in Figure 2-7, the inputs are the eight word vectors and the outputs are the softmax probabilities of all words. The training goal is to train the model on the sample data so that the softmax probability of the target word is maximized. The input layer of the corresponding CBOW neural network model has 8 neurons, and the output layer has Vocab Table-sized neurons. The neurons in the hidden layer can be set arbitrarily. The Deep Neural Network (DNN) is then used with the Back Proportion algorithm to calculate the parameters of the DNN model and the word vector of all words. Thereby, when a new request is made, i.e., once the target word with the highest probability of correspondence to the other eight words is identified, the front Proportion algorithm of the DNN can be executed to identify the word neuron with the maximum probability in the softmax activity function. Figure 1-5 shows the structure of CBOW

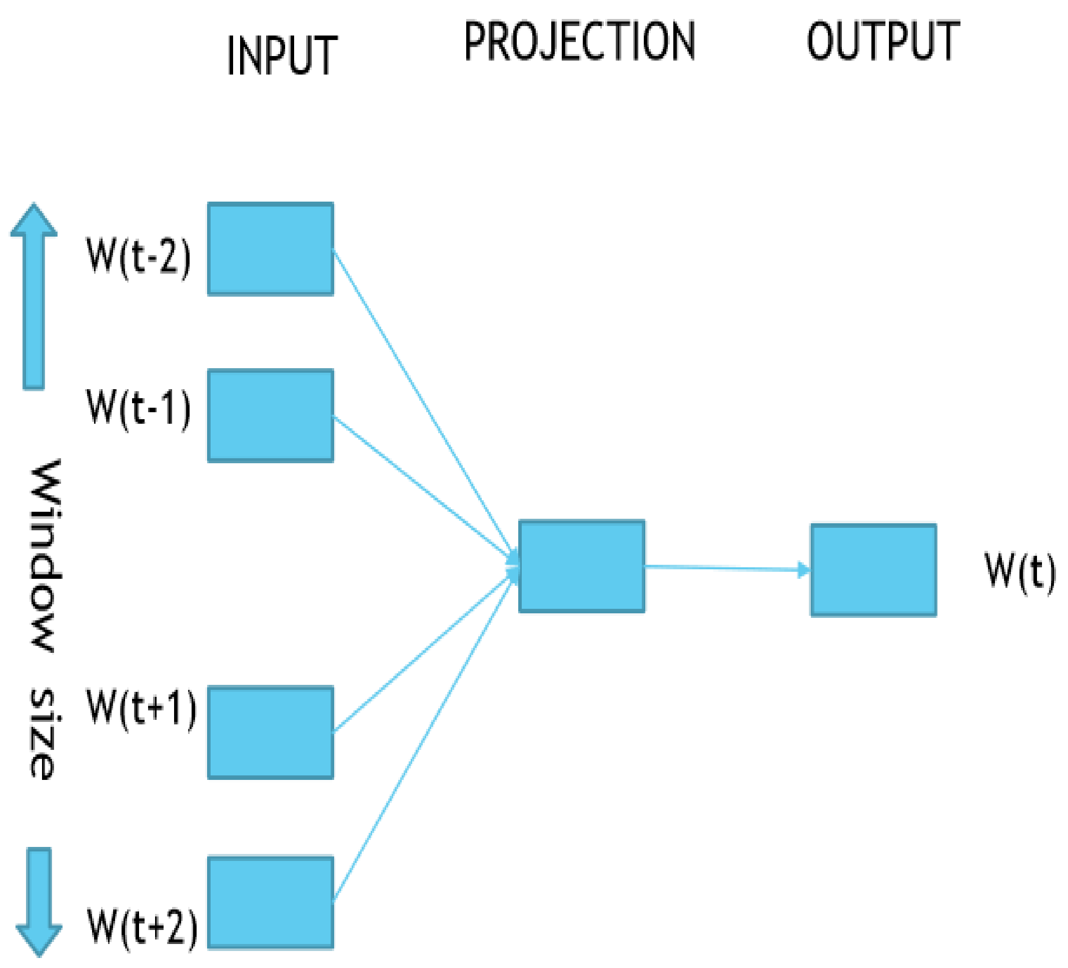


Figure 2-7 structure of CBOW

(2)Skip-Gram

On the other hand, the Skip-Gram model is based on the opposite concept of CBOW, where the input is a word vector of one specific word and the output is a word vector of the word's context (Figure 2-8). In the example shown in Figure 2-6, if the context range is set to 4 and the input is the specific word "learning," the output is the 8 words in the context. In this Skip-Gram example, the input is the word vector of the specific word, and the output is the top 8 words with the highest softmax probability of the word. The input layer of the corresponding Skip-Gram neural network model has one neuron, and the output layer has Vocab Table-sized neurons. The neurons in the hidden layer can be set arbitrarily. The DNN is then used with the Back Proportion algorithm to calculate the parameters of the DNN model and the word vector of all words. Then, when a new request is made, i.e., when 8 words are identified that are likely to be the context of a word, the DNN's front Proportion algorithm can be run to identify the word neuron with the largest probability in the softmax activity function.

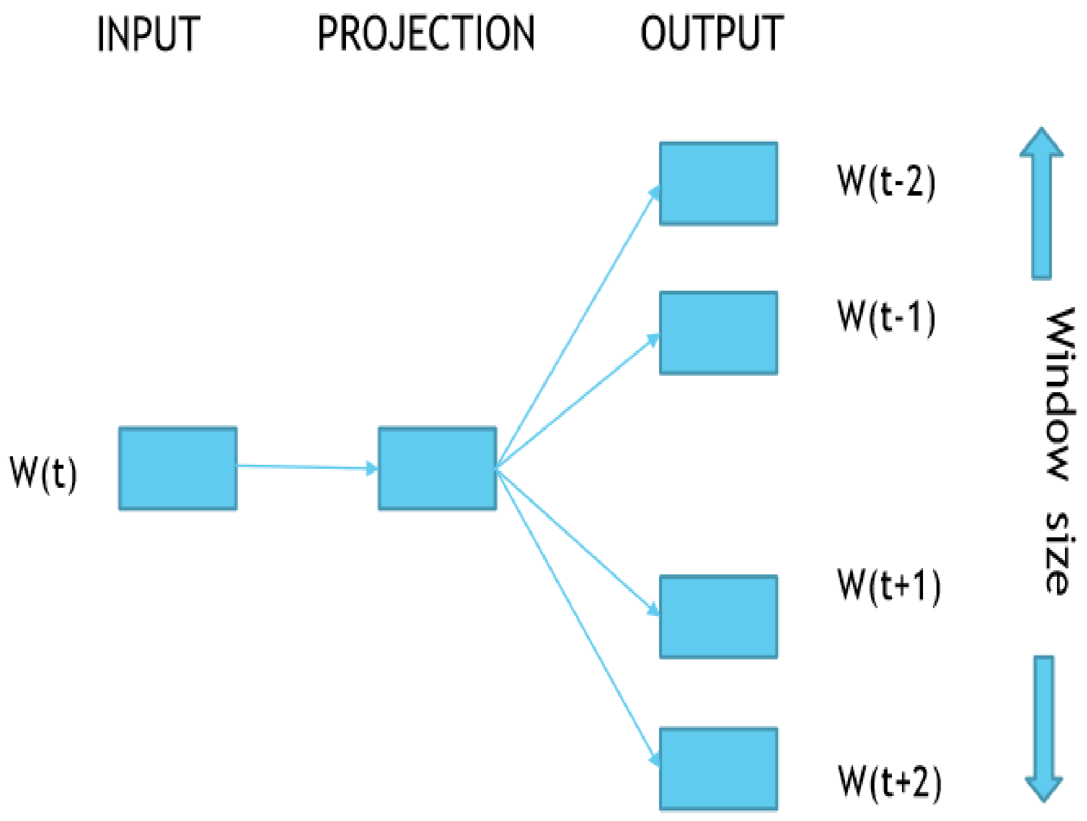


Figure 2-8 structure of Skip-Gram

The above is the process of using CBOW and Skip-Gram in a neural network to train models or to obtain word vectors. However, Word2Vec does not use these DNN models. This is mainly because the time cost of the DNN model processing process is very high: the number of words in a Vocab Table is generally said to be at the million-word level or higher, and the output layer of the DNN is computationally expensive to calculate the output probability of each word in softmax. Therefore, Word2Vec uses a Huffman Tree as a substitute for the neurons in the hidden and output layers. The leaf nodes of the Huffman Tree play the role of neurons in the output layer, the number of leaf nodes is responsible for the size of the Vocab Table, and the intermediate nodes play the role of the hidden layer.

2.4.2 BERT

BERT is an improved model based on Transformer model proposed by Google in 2019[31]. It uses the encoder part of Transformer for bidirectional encoding and requires minimal architectural changes for most natural language processing tasks. By using the pre-trained Transformer encoder, BERT is able to represent lexical elements based on their bidirectional context.

Originally applied to sequence-to-sequence learning on text data, the Transformer model has now been generalized to knife a variety of modern deep learning in areas such as language, vision, speech, and reinforcement learning[32].

Chapter 3. Pre-trained ResNet model transfer learning on medical text classification

3.1 Background

With the recent re-entry of AI into science and public awareness, there have been new breakthroughs in AI in the field of healthcare, and the clinical environment is being filled with new AI technologies at an extremely fast rate. Nevertheless, healthcare is one of the most exciting applications of AI. Since the 20th century, researchers have proposed and established multiple systems to support clinical decisions [33]. Since the 1970s, the rule-based approach has achieved many successes and is now seen as a theory that explains electrocardiograms[34], identifies diseases [35], chooses the best therapy , provides scientific logical explanations, and assists doctors in formulating diagnostic hypotheses and challenging patient cases [36]. However, rule-based systems are expensive and unstable because they need to clearly express decision making rules and, like textbooks, require manual modifications. In addition, high-level interactions among various pieces of information compiled by different experts are difficult to code, and the efficiency of the structure is limited by the comprehensiveness of prior medical knowledge [6]. It is difficult to combine deterministic and probabilistic reasoning procedures to narrow down the appropriate psychological background, prioritize medical theories, and prescribe treatment plans; it is also difficult to create a method that combines deterministic and probabilistic reasoning procedures [7].

Compared to general text, medical text has features such as mass terminology and contains structured text data and free-text data. In the medical textual processing field, AI is mainly used to deal with the tasks of text classification, namely entity recognition and relationship extraction[37]. During text classification, because of the information contained in text data, text classifiers based on deep neural networks have fewer layers and their performance is slightly inferior than that of the mature and high-accuracy methods of image classifiers.

3.2 Method

The output of the pre-processing module was fed into the word-embedding mod-ule. The word embeddings used were variants of Word2Vec and BERT. These two word-embedding modes were selected because they represent order-independent and order-dependent word-embedding modes, respectively. Word2vec takes the form of a bag of words, and it is insensitive to the order of the words entered. BERT is an encoder that uses a transformer and assumes that the word order of an input sentence has a meaning.

In the classification module, the output of the word-embedding module was fed into a tuned ResNet network. Although ResNet as a CNN architecture has its origins in computer vision, where images form the input to the network, it is reasonable to use a sequence of word vectors as the input. In a sentence, the relative positions of words convey their meanings. This approach is similar to the role played by the pixels in conveying information in an image. Therefore, in the pre-processing module used in this study, the text was adjusted to a two-dimensional format to make it similar to an im-age. In addition, the image content is usually 3-channel (RGB) or single-channel (binary or grayscale) in nature. This study uses word vectors as the input, and each element of the word vector is used as the value of a channel, meaning that it will be used as the classifier. ResNet is adjusted to ensure that it can accept the channel of the number of word vector dimensions (100 and 256 in this study). And this study also used the AlexNet and VGG11 as classifiers to perform comparisons. Simultaneously, a one-dimensional CNN was prepared to classify sentences without text pre-processing to compare the effects of the pre-processing module.

3.3 Experiment

We used the open medical text dataset in Kaggle, which contains 14,438 clinical texts divided into five categories. This dataset consists of medical abstracts that de-scribe the current condition of a patient. And this dataset was described as a situation in which doctors routinely scan dozens or hundreds of abstracts each day as they complete their duties in a hospital, meaning that they must quickly pick up on the salient information pointing to the patient's malady. This study used this dataset to classify the class of problems described in these abstracts. All of the texts have been tagged. One-fifth of the training set was used as the verification set; its categories and quantities are listed Table 3-1.

To better apply the image-processing algorithm, each record in the dataset was converted into a 25 × 25 (number of words) format. This size format was used because the maximum text length in the dataset was 600; therefore, a size of 25 × 25 = 625 was chosen to accommodate all of the text. If the number of words in the text was insufficient, the # symbol was used.

Table 3-1. Number of reports in each category.

	Training	**Validation**
Digestive system	2530	632
Cardiovascular	1194	299
Neoplasms	1539	385
Nervous system	2440	610
General pathological	3843	961

After using Word2Vec to generate word vectors on the training set without pre-processing, the window size was 100, and the generated word-vector dimension was 100. The word vectors of the filled symbols and unrecognized words are set as zero vectors. The BERT model uses BERT-mini, with parameters L = 4 and H = 256 denoted as the word-embedding method.

ResNet and 1D CNN were used as classifiers to perform text classification. For ResNet, we used ResNet18 with the pre-trained parameters. In addition, each text in the 25 × 25 format was treated as a picture with several channels in the word–vector dimension, that is, the m elements of the word in the jth row and the kth column in text i corresponded to the point p of picture i on channel m (j,k). Simultaneously, the model was fine-tuned such that the input channel was the word–vector dimension and the output size was 5. For the 1D CNN, LeNet was modified into a one-dimensional structure. The input text was not formatted as 25 × 25, instead using a one-dimensional text vector column, and was filled up to a length of 625. The process used is shown in Figure 2. In addition, the Naïve Bayes method of the traditional text classifier was used to perform comparison, using tf-idf as the feature extraction, text without word embedding and length padding as the input.

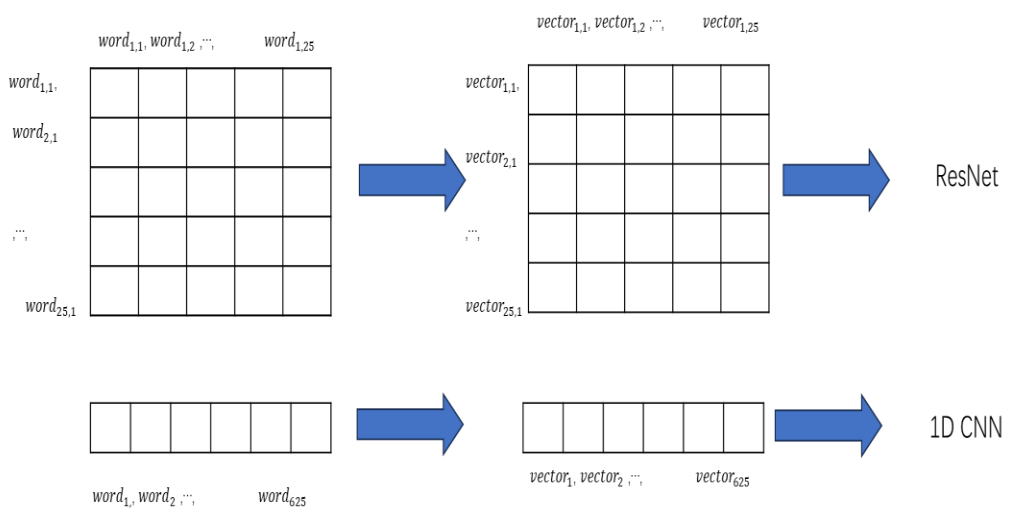

$word_{1,1}, word_{1,2}, \cdots, word_{1,25}$
$word_{1,1}, word_{2,1}, \cdots, word_{25,1}$
$vector_{1,1}, vector_{1,2}, \cdots, vector_{1,25}$
$vector_{1,1}, vector_{2,1}, \cdots, vector_{25,1}$
ResNet
$word_1, word_2, \cdots, word_{625}$
$vector_1, vector_2, \cdots, vector_{625}$
1D CNN

Figure 3-1. Process of text classification.

3.4 Result

Table 3-3 presents the performance of each model using recall, precision, F1, and accuracy as indicators. As the data used in this study were biased in terms of the amount of data in each category, weighted recall, precision, and F1 values were used to perform the evaluation. The 2D form of ResNet18 using pre-trained parameters and retraining on the Kaggle medical text dataset exhibited the best performance, having a weighted F1 value of 90.2% and an accuracy rate of 90.9%. Simultaneously, the weighted F1 values of the traditional text classifier Naïve Bayes model used to perform comparison were 42.2% and 47.8%, respectively. And the models of AlexNet and VGG11 did not have a good performance, as the accuracy rates were lower than that of Naïve Bayes.

Table 3-2. Settings of models.

Models	**Parameters**	**Value**
	Vector size	100
	Model	Skip-gram
Word2vec	Optimizer	Hierarchical softmax
	Window size	100
	Min_count	1
ResNet18	Learn rate	0.1
	Loss	CrossEntropy
	Optimizer	SGD
		Weight decay 5e-4
	Batch size	128
	Learn rate	0.1
	Loss	CrossEntropy
1 D CNN	Optimizer	SGD
		Weight decay 5e-4
	Batch size	128
Naïve Bayes	Feature extraction	Tf-idf

Table 3-3. Performance of each model.

	Recall (Weighted)	**Precision (Weighted)**	**F1 Score (Weighted)**
Naïve Bayes	47.8	62.8	42.2
Bi-LSTM	74.4	77.9	76.2
ResNet (Word2Vec)	90.9	91.1	90.2
1D CNN (Word2Vec)	10.0	1.0	8
AlexNet (Word2Vec)	32.6	10.6	16.0
VGG11 (Word2Vec)	32.6	10.6	16.0
ResNet (BERT)	85.7	87.9	86.8
1D CNN (BERT)	12.0	2.3	8.7

Table 3-4 lists the differences in the effects of ResNet18 (word embedding through word2vec) in different situations. The compared models were as follows: (1) fine-tuning the input and output layers of ResNet, using the parameters pre-trained on ImageNet and re-training the model on the medical text dataset, that is, the model shown in Table 2 in this study; (2) only the input and output layers of ResNet were fine-tuned, and the model was trained using the medical text dataset; (3) the input and output layers of ResNet were fine-tuned, and the parameters were pre-trained in ImageNet. The above three models selected the model with the best performance in the case of Epoch 25. In the case of direct training without using pre-training parameters, the F1 value was only 40.2%, and the correct rate was only 46%, mainly because it had not yet converged at Epoch 25 and the training of the model was insufficient. In the case of only using the pre-training parameters of ImageNet, the performance was extremely poor (F1 7.7%, accuracy 12.5%) because the parameters were trained using ImageNet images, and the input was three channels; this study used data in a 100-channel format as the input, which did not match its parameters.

Table 3-4 Performance of different models on ResNet(Word2Vec)

	Recall (Weighted)	**Precision (Weighted)**	**F1 Score (Weighted)**
With fine-tuning and pre-trained parameters	90.9	91.1	90.2
With fine-tuning but no pre-trained parameters	46.0	42.4	40.2
Only with fine-tuning	12.5	10.8	7.7

Figure 3-2 shows the training and verification accuracies of the ResNet18 model over Epoch 25. The left and right figures show the training and verification processes, respectively. The horizontal axis of the training process represents the number of iterations, and one epoch of the training process contained 91 iterations. The ResNet18 model converged and, finally, found an accuracy of over 90% in training and validation.

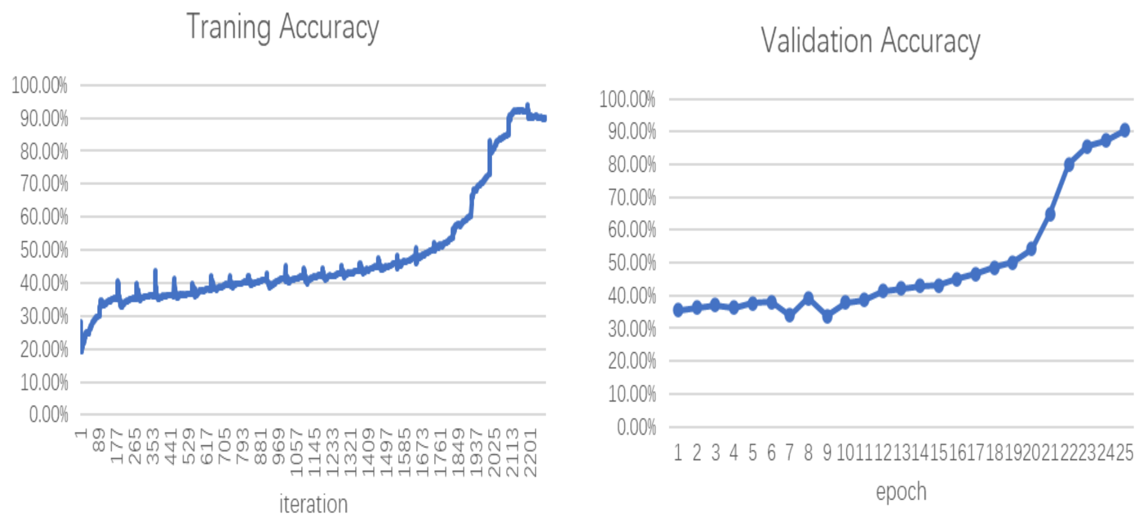


Figure 3-2. Accuracy of ResNet18 in training and validation.

3.5 Discussion

Kilimci et al. confirmed that deep-learning methods perform better than traditional methods in text classification [38]. However, these studies directly applied deep neural networks to the text and trained the features obtained via extracting the tf-idf or word vector of the text as a feature. This approach means that these black box models have low interpretability regarding the conclusions drawn. Some studies directly apply deep learning CNN to text classification and achieve a certain degree of interpretability by mixing the word’s word vector with the clinical identifier CUI corresponding to the word [39]. This study shows that models using deep learning have comparable or even better performances than traditional methods. However, the deep learning model implemented by mixing word vectors was slightly less intuitive for interpreting the results. This study considered converting text information into an image format, which was convenient for attaching additional explanation modules, and the interpretation of the results was more intuitive [40].

In addition, this study explored the transfer learning of ResNet, which is commonly used in image processing and text processing, and compared it to traditional text-processing methods, such as Naïve Bayes and Bi-LSTM. Compared to other deep learning networks, ResNet presented a superior performance, as its structure can solve the problem of gradient explosion. At the same time, it is compared to the results of directly using the 1D CNN in the classifier module to verify the advantages and disadvantages of converting the text format into a two-dimensional form. Previous studies[41] tended to use a 1D CNN to directly process text sequences. In this study, the text is treated as an image format, which makes the performance of the CNN algorithm better. In addition, despite the different image formats, the parameters of pre-training on ImageNet still contribute significantly to the performance of the model, and the ResNet18 model is able to converge within 25 epochs with the use of the pre-training parameters. Also as argued in [42], pre-training on the corresponding dataset plays an important role in improving the model results. And as a limitation, the impact of the choice of word embedding methods was not explored in depth in this study, and the effect of the types of word embedding methods due to the type of word embedding methods and the setting of their hyperparameters on the categorization of medical texts in this study needs to be verified in the future. Additionally, this study did not compare the performance to those of some state-of-the-art models, but only showed the performance of the proposed model and compared it to some classical models. The comparison with state-of-the-art models will be demonstrated in a future work.

Chapter 4. Apply Grad-CAM on text classification for visualization of explainability

4.1 Background

In Chapter 3, "Chapter 3. Pre-trained ResNet model transfer learning on medical text classification", the effectiveness of various types of models for the text classification task is compared, and the feasibility of using the model transfer learning, which is usually used in image processing, for text processing tasks. The effects of pre-training parameters and retraining on the current dataset were also investigated. In this chapter, XAI methods are added to the original model to give it interpretability, and XAI is categorized into pre and post methods with respect to the order of interpretation relative to prediction. The construction of interpretability in the pre-method occurs before the prediction, so that the content of the model, the logic of judgment, and the process can be known at the time of prediction. It is necessary to construct explanatory contents such as knowledge graphs or rule bases before the model is used, and it is also necessary to pay attention to the connection with explanatory modules in the model construction. In contrast, the posterior approach provides explanations after the model predictions are made, so it is usually difficult to show the decision-making process when the model arrives at a result, but only the elements that led the model to that result. In contrast, posterior methods usually do not require additional construction, and most of them are plug-and-play, which makes them less expensive and more generalizable. In post-hoc

XAI, gradient-weighted class activation mapping (Grad-CAM) is a visualization method. This method provides explainability by visualizing the area on which the model focuses when making predictions in the form of a heat map, intuitively showing the basis used by the model to make decisions. Compared to the CAM method, this method can be applied to various convolutional neural networks (CNN) without adjusting the network structure and has better versatility. This approach is now widely used in computer vision-related tasks to intuitively obtain model interpretability. Therefore, this study considers applying this concept to related word-processing tasks to ensure that text-processing tasks can also achieve interpretable visualization.

This study uses a high-accuracy computer vision algorithm model to transfer learning to medical text tasks while simultaneously using the interpretive visualization method Grad-CAM to generate heat maps to ensure that the basis for decision-making can be provided intuitively or via the model. This study makes the black-box neural network model interpretable by attaching the Grad-CAM module to the model and can be visualized through the heatmap. In order to generate the heat map, a CNN method is required, meaning that this study extracts features from the text by performing word embedding and presents the text as an image (the dimensions of the word vectors replace the channels of the RGB image).

4.2 Method

This study used models that are often used in computer vision area to transfer learning to text-processing tasks and used the Grad-CAM method to visualize the explainability of the model. This study uses the Grad-CAM technology to generate class activation maps (thermal phase maps) for a given text and predicted class. Each element of the class activation map corresponds to a token and indicates its importance based on the score of a particular class. Class activation maps provide information regarding the extent to which a particular token present in an input sentence affects ANN predictions. Meanwhile, this study also used SHAP values to explain the contribution of each feature to the inputs of the model prediction.SHAP values are derived from Shapley values in cooperative game theory, which are used to distribute the gains in cooperative games. In machine learning, SHAP provides an interpretable method to help humans understand the contribution of each feature to the model prediction.SHAP assigns a value to each feature that indicates the degree of influence of that feature on the model output, which in this study is the influence of each word in the text. For a particular prediction, the SHAP value indicates the contribution of each feature to that prediction. The interpretability provided by this contribution is localized, i.e., it provides an explanation of the prediction for a single sample.The positivity or negativity of the SHAP value indicates whether the feature has a positive or negative influence on the model output.

The model used in this study is the ResNet18 model from Chapter 3 (word2vec word embeddings that were retrained on a medical text dataset using the pre-training parameters of ImageNet).

4.3 Result

Figure 4-1 shows an example of a heatmap generated for use in Grad-CAM, indicating the regions on which the model focuses when making predictions. The white area indicates that the attention of the model is high. Figure 4-2 shows the text corresponding to the heat map shown in Figure 4-1. The example shows the prediction of the text of the digestive system and highlights the position on which the model focused during the prediction. As shown in the figure, a part of the position that the model focused on is “follicular thyroid cancer treated at the Mayo Clinic, 1946 through 1970.”

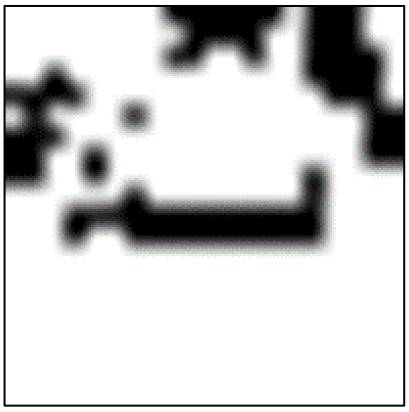
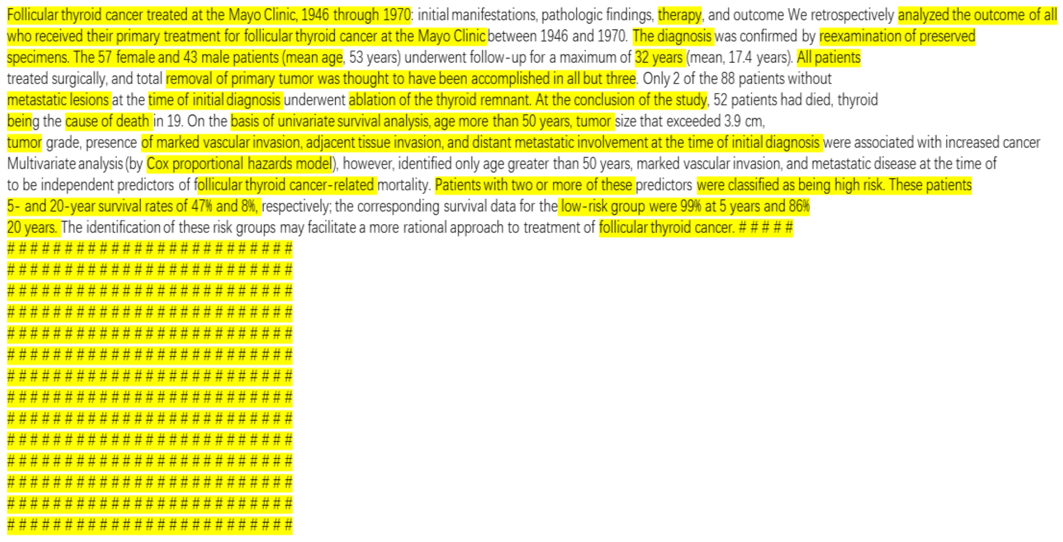


Figure 4-1. Example of a heatmap used to predict text of the digestive system.



Follicular thyroid cancer treated at the Mayo Clinic, 1946 through 1970: initial manifestations, pathologic findings, therapy, and outcome We retrospectively analyzed the outcome of all
who received their primary treatment for follicular thyroid cancer at the Mayo Clinic between 1946 and 1970. The diagnosis was confirmed by reexamination of preserved
specimens. The 57 female and 43 male patients (mean age, 53 years) underwent follow-up for a maximum of 32 years (mean, 17.4 years). All patients
treated surgically, and total removal of primary tumor was thought to have been accomplished in all but three. Only 2 of the 88 patients without
metastatic lesions at the time of initial diagnosis underwent ablation of the thyroid remnant. At the conclusion of the study, 52 patients had died, thyroid
being the cause of death in 19. On the basis of univariate survival analysis, age more than 50 years, tumor size that exceeded 3.9 cm,
tumor grade, presence of marked vascular invasion, adjacent tissue invasion, and distant metastatic involvement at the time of initial diagnosis were associated with increased cancer
Multivariate analysis (by Cox proportional hazards model), however, identified only age greater than 50 years, marked vascular invasion, and metastatic disease at the time of
to be independent predictors of follicular thyroid cancer-related mortality. Patients with two or more of these predictors were classified as being high risk. These patients
5- and 20-year survival rates of 47% and 8%, respectively; the corresponding survival data for the low-risk group were 99% at 5 years and 86%
20 years. The identification of these risk groups may facilitate a more rational approach to treatment of follicular thyroid cancer. # # # # #
# #
# #
# #
# #
# #
# #
# #
# #
# #
# #
# #
# #
# #
# #

Figure 4-2. Highlighting the position of the text combined with the heatmap.

Figure 4-3 demonstrates the SHAP values of the input features (presents each word) for the prediction of an article categorized as “General pathological”. The left side of Figure 4-3 shows the conceptual image formed by the pixels representing the word vectors of the document. Since the word vectors have a dimension of 100, the resulting image will also have 100 channels, and thus the image cannot be interpreted as an image in the usual sense, but only as a picture showing the relative positions of the words in the original text. The right side shows the SHAP value for each word in the text for each output. From output0 to output4, it indicates the SHAP values of each features (presents each word) when the model categorizes the text as "Digestive system", "Cardiovascular", "Neoplasms", "Nervous system", "General pathological". The more red portions, the more features that provide a positive contribution; the more blue portions, the more features that provide a negative contribution. In this prediction result, the correct classification is General pathological, and the graph represented by output4 is the correctly predicted result, which also has more positive features as expected.

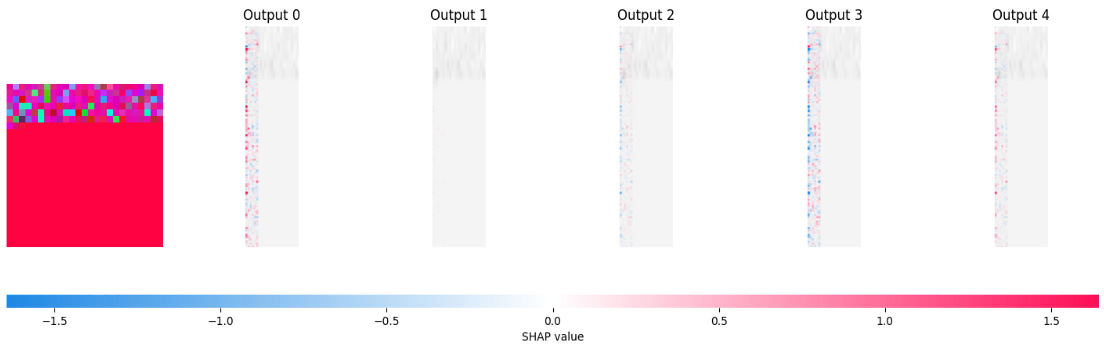


Figure 4-3. The SHAP value of each output

4.4 Discussion

This study applied ResNet to medical text-processing tasks and demonstrated the interpretability of the model using Grad-CAM. In the field of medical texts, Grad-CAM has not been used to generate heat maps to achieve interpretable AI. This study used Grad-CAM to generate heat maps, which visually present the words to which the model pays attention when making predictions and realizes the interpretability of the model. In addition, SHAP value was used to assign interpretability to the model, showing how each feature of the model contributes to a particular prediction outcome. In the study of

[43], SHAP value was used to provide a relative quantitative analysis of model parameter selection. In the study of [44], interpretability was provided for the results of audio-acoustic processing. In this study, SHAP value was used to analyze the overall interpretability of the images.

Compared to other XAI methods, Grad-CAM can visualize the explainability of a model and be easily added to various ANN methods to visualize the model interpretation. Other interpretable methods for text, such as knowledge graphs, ontology databases, and other interpretable methods used to construct rules, incur considerable labor costs in construction and updating. As a post hoc XAI method, Grad-CAM has a higher degree of freedom, and its structure can be adapted to various ANN methods. Even if the model was modified, it could be processed using the same procedure. In contrast, Grad-CAM can only provide explainability but not interpretability, that is, it can only determine the basis of the results given by the model, and it cannot know the reasoning process influencing the results given by the model. The limitation of this study is that the explainability provided is relatively qualitative; the physicians are able to be informed of the basis for the results given by the AI model, however, it is still up to the physicians to make them own judgment as to whether or not that basis is reasonable and whether or not the results should be adopted.

5. Conclusion

This study explores natural language processing of medical text. In Chapter 3, the problem of medical text categorization is learned using a pre-trained image processing model transfer that treats medical text as an image format for processing. The word vector of each word in the medical text is treated as a pixel point in the image, to which various types of CNN models are applied and the model effects are compared. Traditional natural language processing models are also used to apply directly to the medical text, and the modeling effects are compared. In addition, the effects of 1D CNNs, which are commonly used for text categorization, were also compared. The best results (weighted Recall, weighted F1 score) were obtained by training a ResNet model with pre-training parameters on imaged text used in this study. This model architecture was then compared between Model A, which used the pre-trained model parameters and was retrained using image-formatted text, with Model B, which did not use the pre-trained model parameters but was retrained using image-formatted text, and Model C, which used only the pre-trained model parameters. The result is that Model A has a significant effect advantage. This study verifies the feasibility of converting text format to image format representation and transfer learning using high-precision image processing models.

In Chapter 4, Grad-CAM, which uses a visual interpretable method, is appended to the model to achieve interpretability of the model and to be able to intuitively see what the model is basing its conclusions on. This is more intuitive than the usual textual interpretable method, and is more convenient to use because it does not require too much modification of the model and is less labor-intensive.

In this study, the feasibility of processing medical text in the form of images is explored through Chapters 3 and 4, comparing the effects with traditional text processing methods, and the interpretability of the model and the visualization of the interpretability are achieved through the addition of the Grad-CAM module.

Reference

[1] S. E. Dilsizian and E. L. Siegel, “Artificial Intelligence in Medicine and Cardiac Imaging: Harnessing Big Data and Advanced Computing to Provide Personalized Medical Diagnosis and Treatment,” *Curr. Cardiol. Rep.*, vol. 16, no. 1, p. 441, Dec. 2013, doi: 10.1007/s11886-013-0441-8.

[2] V. L. Patel *et al.*, “The coming of age of artificial intelligence in medicine,” *Artif. Intell. Med.*, vol. 46, no. 1, pp. 5–17, May 2009, doi: 10.1016/j.artmed.2008.07.017.

[3] S. Jha and E. J. Topol, “Adapting to Artificial Intelligence: Radiologists and Pathologists as Information Specialists,” *JAMA*, vol. 316, no. 22, pp. 2353–2354, Dec. 2016, doi: 10.1001/jama.2016.17438.

[4] E. Strickland, “IBM Watson, heal thyself: How IBM overpromised and underdelivered on AI health care,” *IEEE Spectr.*, vol. 56, no. 4, pp. 24–31, Apr. 2019, doi: 10.1109/MSPEC.2019.8678513.

[5] D. B. Neill, “Using Artificial Intelligence to Improve Hospital Inpatient Care,” *IEEE Intell. Syst.*, vol. 28, no. 2, pp. 92–95, Mar. 2013, doi: 10.1109/MIS.2013.51.

[6] E. S. Berner *et al.*, “Performance of Four Computer-Based Diagnostic Systems,” *N. Engl. J. Med.*, vol. 330, no. 25, pp. 1792–1796, Jun. 1994, doi: 10.1056/NEJM199406233302506.

[7] P. Szolovits and S. G. Pauker, “Categorical and probabilistic reasoning in medical diagnosis,” *Artif. Intell.*, vol. 11, no. 1–2, pp. 115–144, Aug. 1978, doi: 10.1016/0004-3702(78)90014-0.

[8] F. Jiang *et al.*, “Artificial intelligence in healthcare: past, present and future,” *Stroke Vasc. Neurol.*, vol. 2, no. 4, pp. 230–243, Dec. 2017, doi: 10.1136/svn-2017-000101.

[9] A. Esteva *et al.*, “A guide to deep learning in healthcare,” *Nat. Med.*, vol. 25, no. 1, pp. 24–29, Jan. 2019, doi: 10.1038/s41591-018-0316-z.

[10] J. Krause *et al.*, “Grader Variability and the Importance of Reference Standards for Evaluating Machine Learning Models for Diabetic Retinopathy,” *Ophthalmology*, vol. 125, no. 8, pp. 1264–1272, Aug. 2018, doi: 10.1016/j.ophtha.2018.01.034.

[11] A. Holzinger, B. Malle, A. Saranti, and B. Pfeifer, “Towards multi-modal causability with Graph Neural Networks enabling information fusion for explainable AI,” *Inf. Fusion*, vol. 71, pp. 28–37, Jul. 2021, doi: 10.1016/j.inffus.2021.01.008.

[12] A. Eck *et al.*, “Interpretation of microbiota-based diagnostics by explaining individual classifier decisions,” *BMC Bioinformatics*, vol. 18, no. 1, p. 441, Dec. 2017, doi: 10.1186/s12859-017-1843-1.

[13] A. Barredo Arrieta *et al.*, “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020, doi: 10.1016/j.inffus.2019.12.012.

[14] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700. in Lecture Notes in Computer Science, vol. 11700. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-28954-6.

[15] A. Adadi and M. Berrada, “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),” *IEEE Access*, vol. 6, pp. 52138–52160, 2018, doi: 10.1109/ACCESS.2018.2870052.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[17] C. Dong, C. C. Loy, K. He, and X. Tang, “Image Super-Resolution Using Deep Convolutional Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016, doi: 10.1109/TPAMI.2015.2439281.

[18] B. C. Kwon *et al.*, “RetainVis: Visual Analytics with Interpretable and Interactive Recurrent Neural Networks on Electronic Medical Records,” *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 299–309, Jan. 2019, doi: 10.1109/TVCG.2018.2865027.

[19] J. Zhang, K. Kowsari, J. H. Harrison, J. M. Lobo, and L. E. Barnes, “Patient2Vec: A Personalized Interpretable Deep Representation of the Longitudinal Electronic Health Record,” *IEEE Access*, vol. 6, pp. 65333–65346, 2018, doi: 10.1109/ACCESS.2018.2875677.

[20] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, “RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2016. Accessed: Nov. 25, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/hash/231141b34c82aa95e48810a9d1b33a79-Abstract.html

[21] D. A. Kaji *et al.*, “An attention based deep learning model of clinical events in the intensive care unit,” *PLOS ONE*, vol. 14, no. 2, p. e0211057, Feb. 2019, doi: 10.1371/journal.pone.0211057.

[22] B. Shickel, T. J. Loftus, L. Adhikari, T. Ozrazgat-Baslanti, A. Bihorac, and P. Rashidi, “DeepSOFA: A Continuous Acuity Score for Critically Ill Patients using Clinically Interpretable Deep Learning,” *Sci. Rep.*, vol. 9, no. 1, p. 1879, Feb. 2019, doi: 10.1038/s41598-019-38491-0.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA: ACM, Aug. 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.

[24] T. Panch, H. Mattie, and L. A. Celi, “The ‘inconvenient truth’ about AI in healthcare,” *Npj Digit. Med.*, vol. 2, no. 1, p. 77, Aug. 2019, doi: 10.1038/s41746-019-0155-4.

[25] M. Porumb, S. Stranges, A. Pescapè, and L. Pecchia, “Precision Medicine and Artificial Intelligence: A Pilot Study on Deep Learning for Hypoglycemic Events Detection based on ECG,” *Sci. Rep.*, vol. 10, no. 1, p. 170, Jan. 2020, doi: 10.1038/s41598-019-56927-5.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied

to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[28] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2921–2929. doi: 10.1109/CVPR.2016.319.

[29] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization".

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2013. Accessed: Nov. 25, 2023. [Online]. Available: https://papers.nips.cc/paper_files/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/1810.04805

[32] A. Vaswani *et al.*, "Attention Is All You Need." arXiv, Aug. 01, 2023. Accessed: Nov. 24, 2023. [Online]. Available: http://arxiv.org/abs/1706.03762

[33] P. E. Beeler, D. W. Bates, and B. L. Hug, "Clinical decision support systems," *Swiss Med. Wkly.*, vol. 144, no. 5152, Art. no. 5152, Dec. 2014, doi: 10.4414/smw.2014.14073.

[34] M. Kundu, M. Nasipuri, and D. Kumar Basu, "Knowledge-based ECG interpretation: a critical review," *Pattern Recognit.*, vol. 33, no. 3, pp. 351–373, Mar. 2000, doi: 10.1016/S0031-3203(99)00065-5.

[35] F. T. De Dombal, D. J. Leaper, J. R. Staniland, A. P. McCann, and J. C. Horrocks, "Computer-aided Diagnosis of Acute Abdominal Pain," *BMJ*, vol. 2, no. 5804, pp. 9–13, Apr. 1972, doi: 10.1136/bmj.2.5804.9.

[36] R. A. Miller, M. A. McNeil, S. M. Challinor, F. E. Masarie, and J. D. Myers, "The INTERNIST-1/QUICK MEDICAL REFERENCE Project—Status Report," *West. J. Med.*, vol. 145, no. 6, pp. 816–822, Dec. 1986.

[37] S. Wu *et al.*, "Deep learning in clinical natural language processing: a methodical review," *J. Am. Med. Inform. Assoc. JAMIA*, vol. 27, no. 3, pp. 457–470, Dec. 2019, doi: 10.1093/jamia/ocz200.

[38] Z. H. Kilimci and S. Akyokus, "Deep Learning- and Word Embedding-Based Heterogeneous Classifier Ensembles for Text Classification," *Complexity*, vol. 2018, pp. 1–10, Oct. 2018, doi: 10.1155/2018/7130146.

[39] P. Wang, X. Kong, W. Guo, and X. Zhang, "Exclusive Feature Constrained Class Activation Mapping for Better Visual Explanation," *IEEE Access*, vol. 9, pp.

61417–61428, 2021, doi: 10.1109/ACCESS.2021.3073465.

[40] L. Yao, C. Mao, and Y. Luo, “Clinical text classification with rule-based features and knowledge-guided convolutional neural networks,” *BMC Med. Inform. Decis. Mak.*, vol. 19, no. S3, p. 71, Apr. 2019, doi: 10.1186/s12911-019-0781-4.

[41] Ł. Górski and S. Ramakrishna, “Towards Grad-CAM Based Explainability in a Legal Text Processing Pipeline”.

[42] Li, J., Lin, Y., Zhao, P. et al. Automatic text classification of actionable radiology reports of tinnitus patients using bidirectional encoder representations from transformer (BERT) and in-domain pre-training (IDPT). BMC Med Inform Decis Mak 22, 200 (2022).

[43] Kumar, N.; Sharma, M.; Singh, V.P.; Madan, C.; Mehandia, S. An empirical study of handcrafted and dense feature extraction techniques for lung and colon cancer classification from histopathological images. Biomed. Signal Process. Control 2022, 75, 103596.

[44] A. Raza Syed and M. I. Mandel, "Estimating Shapley Values of Training Utterances for Automatic Speech Recognition Models," ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 2023, pp. 1-5, doi: 10.1109/ICASSP49357.2023.10097237.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Professor Ogasawara. I have benefited tremendously from his critical thinking and insightful viewpoint. Through his patient instruction, I finally focused on the object studied in this thesis, and obtained valuable advice on aspects ranging from frame work constructing and data analysis.

Then, I pleased to acknowledge the Professor Uesugi and Professor Takashima for their comments.

Besides, I am grateful for the members of the lab of Ogasawara for their comments and wise advice.

Moreover, I owe my thanks to my family and friends, who have always supported me with their generous encouragements and praises.

Appendix

A-1 Trainer

```
import os
import sys
import argparse
import numpy as np
import torch
from torch import nn
from torchvision import transforms,utils
from torchvision import datasets
import torchvision.models as models
from torch.utils import data
path_add=r'D:/transfer-test/quicktools'
sys.path.append(path_add)
import textprocess

#import data
data_path = 'DATA_PATH'
#train_img_set,val_img_set = textprocess.get_1Dwv_set(data_path)
train_img_set = torch.load('train_tensor_list')
val_img_set = torch.load('val_tensor_list')

#net = nn.Sequential(
#       nn.Conv1d(100,200,kernel_size=5,padding=2),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=2),
#       nn.Conv1d(200,128,kernel_size=5,padding=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=4,stride=2),
#       nn.Conv1d(128,64,kernel_size=5,padding=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=1),
#       nn.Conv1d(64,16,kernel_size=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=1),
#       nn.Flatten(),
#       nn.Linear(2400,120),nn.Sigmoid(),
#       nn.Linear(120,84),nn.Sigmoid(),
#       nn.Linear(84,5)
#)
#####################################

net = models.alexnet()
net.features[0] = nn.Conv2d(100,64,kernel_size=(7,7),stride=(2,2),padding=(2,2))
net.features[1] = nn.ReLU6()
net.features[2] = nn.MaxPool2d(kernel_size=3,stride=2,padding=0,ceil_mode=False)
net.features[3] = nn.Conv2d(64,192,kernel_size=(3,3),stride=(1,1),padding=(1,1))
net.features[5] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
net.features[12] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
net.avgpool = nn.AdaptiveAvgPool2d(output_size=(3,3))
net.classifier[1] = nn.Linear(in_features=256*3*3,out_features=4096,bias=True)
net.classifier[6] = nn.Linear(in_features=4096,out_features=5,bias=True)
net.classifier[2] = nn.ReLU6()
net.classifier[5] = nn.ReLU6()
#####################################
#net = models.vgg11(pretrained = True)
#net.features[0] = nn.Conv2d(100,64,kernel_size=(3,3),stride=(1,1),padding=(1,1))
#net.features[1] = nn.ReLU6()
#net.features[2] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
#net.features[5] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
#net.features[10] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
#net.features[15] = nn.MaxPool2d(kernel_size=3,stride=2,padding=0,ceil_mode=False)
```

```
#net.features[20] = nn.MaxPool2d(kernel_size=2,stride=2,padding=0,ceil_mode=False)


#net.avgpool = nn.AdaptiveAvgPool2d(output_size=(5,5))
#net.classifier[0] = nn.Linear(in_features=12800,out_features=4096,bias=True)
#net.classifier[1] = nn.ReLU6()
#net.classifier[6] = nn.Linear(in_features=4096,out_features=5,bias=True)
#net.classifier[4] = nn.ReLU6()
##############################


#net = models.resnet18(pretrained = False)


#print(net)
#net.conv1=nn.Conv2d(100,64,kernel_size=(7,7),stride=(2,2),padding=(3,3),bias=False)
#in_channel = net.fc.in_features
#net.fc = nn.Linear(in_channel,5)
print(net)


def init_weights(m):
    if type(m)==nn.Linear:
        nn.init.normal_(m.weight,0.01)


net.apply(init_weights)


my_trans = transforms.Compose([
        transforms.ToTensor()
        ])


#hyperparameter
EPOCH = 25
pre_epoch = 0
```

```
BATCH_SIZE = 128
LR = 0.1
device= torch.device('cpu')


classes = ('1','2','3','4','5')


train_iter = data.DataLoader(train_img_set,batch_size=BATCH_SIZE,shuffle=False)
test_iter = data.DataLoader(val_img_set,batch_size=BATCH_SIZE,shuffle=False)


loss = nn.CrossEntropyLoss()
trainer = torch.optim.SGD(net.parameters(),lr=LR,momentum = 0.9,weight_decay = 5e-4)


def train_classify(EPOCH,pre_epoch,trainloader,testloader,net,optimizer,criterion):
    parser = argparse.ArgumentParser(description='Pytorch Training')
    parser.add_argument('--outf',default='./model/',help = 'folder to output images and model
checkpoints')
    args=parser.parse_args()
    if not os.path.exists(args.outf):
        os.makedirs(args.outf)
    best_acc =85 #
    print("Start Training, ResNet-18!")
    with open('acc.txt','w') as f:
        with open('log.txt','w') as f2:
            for epoch in range(pre_epoch,EPOCH):
                print('\nEpoch: %d'%(epoch+1))
                net.train()
                sum_loss = 0.0
                correct = 0.0
                total = 0.0
                for i,d in enumerate(trainloader,0):
```

```
        length = len(trainloader)
        inputs,labels = d
        inputs,labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        print(inputs.shape)
        outputs = net (inputs)
        loss = criterion(outputs,labels)
        loss.backward()
        optimizer.step()
        sum_loss += loss.item()
        _,predicted = torch.max(outputs.data,1)
        total += labels.size(0)
        correct += predicted.eq(labels.data).cpu().sum()
        print('[epoch:%d,iter:%d] Loss: %.03f | Acc:%.3f%%'
            %(epoch+1,(i+1+epoch*length),sum_loss / (i+1),100.*correct/total))
        f2.write('%03d %05d | Loss: %.03f | Acc:%.3f%%'
            %(epoch+1,(i+1+epoch*length),sum_loss / (i+1),100.*correct/total))
        f2.write('\n')
        f2.flush()

    print('Waiting Test!')
    with torch.no_grad():
        correct=0
        total=0
        for d in testloader:
            net.eval()
            images,labels = d
            images,labels = images.to(device),labels.to(device)
            outputs = net(images)
            _,predicted = torch.max(outputs.data,1)
```

```
                total += labels.size(0)
                correct += (predicted == labels).sum()
            print('test accuracy is ： %.3f%%'%(100 * correct/total))
            acc = 100.*correct/total
            print('Saving model ......')
            torch.save(net.state_dict(),'%s/net_%03d.pth'%(args.outf,epoch +1))
            f.write('EPOCH = %03d,Accuracy= %.3f%%'%(epoch+1,acc))
            f.write('\n')
            f.flush()
            if acc > best_acc:
                f3 = open("best_acc.txt",'w')
                f3.write('EPOCH = %d,best_acc =%.3f%%'%(epoch +1,acc))
                f3.close()
                best_acc =acc
        print("Training Finished, TotalEPOCH=%d"%EPOCH)
train_classify(EPOCH,pre_epoch,train_iter,test_iter,net,trainer,criterion = loss)
os.wait()
```

A-2 Predict

```
import sys
import torchvision.models as models
from torch.utils import data
from torch import nn
import torch
from torch import Tensor
import torch.nn.functional as F
import numpy as np

import cv2
```

```
from sklearn.metrics import f1_score,recall_score,precision_score


path_add=r'D:/transfer-test/quicktools'
sys.path.append(path_add)
import textprocess
def predict(net,testloader):
    with torch.no_grad():
        correct=0
        total=0
        y_predict,y_real = [],[]
        for d in testloader:
            net.eval()
            images,labels = d
            images,labels = images.to(device),labels.to(device)
            outputs = net(images)
            _,predicted = torch.max(outputs.data,1)
            total += labels.size(0)
            y_predict.extend(predicted.tolist())
            y_real.extend(labels.tolist())
            correct += (predicted == labels).sum()


        print('predict result is：  {predict}\n, real category is  ：
{real}\n'.format(predict=y_predict,real=y_real))
        average = 'weighted'
        print('\nweighted precision:',precision_score(y_real,y_predict,average=average),
        '\nweighted recall      :',recall_score(y_real,y_predict,average=average),
        '\nweighted f1_score :',f1_score(y_real,y_predict,average=average))
        print('test accuracy is  ：  %.3f%%'%(100 * correct/total))
#hyperparameter
BATCH_SIZE = 16
```

```
device = torch.device('cpu')

#import data
data_path = 'D:/transfer-test/kaggle_data'
#train_img_set,val_img_set = textprocess.get_wvi_set(data_path,get_train=False)
val_img_set = torch.load('val_tensor_list')

#change the format to dataloader
test_iter = data.DataLoader(val_img_set,batch_size=BATCH_SIZE,shuffle=False)

#import model
net=models.resnet18()
paras = torch.load('./model_res18/net_025.pth')
#net = models.alexnet()
#paras = torch.load('./model_alex/net_024.pth')

#finetuning
net.conv1=nn.Conv2d(100,64,kernel_size=(7,7),stride=(2,2),padding=(3,3),bias=False)
in_channel = net.fc.in_features
net.fc = nn.Linear(in_channel,5)

net.load_state_dict(paras)

#net.features[0] = nn.Conv2d(100,64,kernel_size=(7,7),stride=(2,2),padding=(2,2))
#net.features[1] = nn.ReLU6()
#net.features[2] = nn.MaxPool2d(kernel_size=3,stride=2,padding=0,ceil_mode=False)
#net.features[3] = nn.Conv2d(64,192,kernel_size=(3,3),stride=(1,1),padding=(1,1))
#net.features[5] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
#net.features[12] = nn.MaxPool2d(kernel_size=2,stride=1,padding=0,ceil_mode=False)
#net.avgpool = nn.AdaptiveAvgPool2d(output_size=(3,3))
```

```
#net.classifier[1] = nn.Linear(in_features=256*3*3,out_features=4096,bias=True)
#net.classifier[6] = nn.Linear(in_features=4096,out_features=5,bias=True)
#net.classifier[2] = nn.ReLU6()
#net.classifier[5] = nn.ReLU6()

#net.load_state_dict(paras)
print(net)
#net = nn.Sequential(
#       nn.Conv1d(100,200,kernel_size=5,padding=2),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=2),
#       nn.Conv1d(200,128,kernel_size=5,padding=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=4,stride=2),
#       nn.Conv1d(128,64,kernel_size=5,padding=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=1),
#       nn.Conv1d(64,16,kernel_size=1),nn.Sigmoid(),
#       nn.AvgPool1d(kernel_size=2,stride=1),
#       nn.Flatten(),
#       nn.Linear(2400,120),nn.Sigmoid(),
#       nn.Linear(120,84),nn.Sigmoid(),
#       nn.Linear(84,5)
#)

#predict
predict(net,test_iter)
```

A-3 Text Processing

```
import os
import sys
import argparse
import numpy as np
import torch
from torch import nn
from torchvision import transforms,utils
from torchvision import datasets
import torchvision.models as models
from torch.utils import data
from IPython import display
from pprint import pprint
from gensim.models import word2vec
def txt_loader(input):
    item = open(input,'r',encoding='utf-8').read()

    return item

def txt_split(loader_in):

    txt_list=[]
    for txt in loader_in.dataset:
        txt = list(txt)
        txt[0] = txt[0].rstrip()
        txt[0] = txt[0].replace('\n','')
        txt[0] = txt[0].split(' ')
        txt_list.append(txt)

    return txt_list

def get_tag(loader_in):
        tag_list=[]
        for txt in loader_in.dataset:
            txt = list(txt)
            tag_list.append(txt[1])

        return tag_list

def get_wvi_set(data_path,get_train=True,get_val=True):

        my_data = datasets.DatasetFolder(data_path,loader=txt_loader,extensions="txt",transform=None)
        n_val = int(len(my_data)/5)
        n_train = len(my_data)-n_val
        print(n_val,n_train)
        train_set,val_set = data.random_split(my_data,[n_train,n_val])
        train_loader = data.DataLoader(train_set,batch_size=2,shuffle=True)
        val_loader = data.DataLoader(val_set,batch_size=2,shuffle=True)

        val_txt_list = txt_split(val_loader)
        train_txt_list = txt_split(train_loader)
        val_tag_list = get_tag(val_loader)
        train_tag_list =get_tag(train_loader)

        model=word2vec.Word2Vec.load('C:/Users/zhj/wvmodel_vkg.model')

        wv=[]
        zero_vector = []
        vector_image_train=[]
        vector_image_val=[]
        wv_d = model.wv.vector_size
        for i in range(0,wv_d):
```

```
    zero_vector.append(0)

b=np.empty([25,25,100])
train_data=list()
test_data=list()
train_img_set=[]
val_img_set=[]

if get_train==True:
        for train_txt in train_txt_list:
                for word in train_txt[0]:

                        if word == '#':
                                wv.append(zero_vector)
                        else:
                                c=[]
                                for i in range(0,wv_d):
                                        c.append(model.wv[word][i])
                                #print(c)
                                wv.append(c)

                tmp = [wv.copy(),train_txt[1]]
                vector_image_train.append(tmp)
                #print(tmp)
                wv.clear()

        print(len(vector_image_train))
        for text in vector_image_train:
                b = np.array(text[0]).reshape(25,25,100)
                #print(b)
```

```
            train_data.append(b.copy())
            #test_data = test_data.reshape(25,25,100)
            b.fill(0)
        #test_data = test_data[0].reshape(25,25)
        #test_data=test_data.reshape(25,25)
        train_data=np.array(train_data)
        train_data=torch.tensor(train_data,dtype=torch.float32)

        train_data = train_data.permute(0,3,1,2)

        print(train_data[0])
        print(train_data.shape)
        print(len(train_data))

        for i,train_img in enumerate(train_data):
            tmp = [train_img,torch.tensor(train_tag_list[i])]
            train_img_set.append(tmp)
            #print(len(train_img_set))
            #print(train_img.shape)



    if get_val == True:
        for val_txt in val_txt_list:
            for word in val_txt[0]:

                if word == '#':
                    wv.append(zero_vector)
                else:
                    c=[]
                    for i in range(0,wv_d):
```

```
                c.append(model.wv[word][i])
            #print(c)
            wv.append(c)

    tmp = [wv.copy(),val_txt[1]]
    vector_image_val.append(tmp)
    #print(tmp)
    wv.clear()

print(len(vector_image_val))
for text in vector_image_val:
    b = np.array(text[0]).reshape(25,25,100)
    #print(b)
    test_data.append(b.copy())
    #test_data = test_data.reshape(25,25,100)
    b.fill(0)
#test_data = test_data[0].reshape(25,25)
#test_data=test_data.reshape(25,25)
test_data=np.array(test_data)
test_data=torch.tensor(test_data,dtype=torch.float32)

test_data = test_data.permute(0,3,1,2)

print(test_data.shape)
print(len(test_data))

for i,val_img in enumerate(test_data):
    tmp = [val_img,torch.tensor(val_tag_list[i])]
    val_img_set.append(tmp)
    return train_img_set,val_img_set
def get_1Dwv_set(data_path,get_train=True,get_val=True):
    my_data = datasets.DatasetFolder(data_path,loader=txt_loader,extensions="txt",transform=None)
    n_val = int(len(my_data)/5)
    n_train = len(my_data)-n_val
    print(n_val,n_train)
    train_set,val_set = data.random_split(my_data,[n_train,n_val])
    train_loader = data.DataLoader(train_set,batch_size=2,shuffle=True)
    val_loader = data.DataLoader(val_set,batch_size=2,shuffle=True)

    val_txt_list = txt_split(val_loader)
    train_txt_list = txt_split(train_loader)
    val_tag_list = get_tag(val_loader)
    train_tag_list =get_tag(train_loader)
    model=word2vec.Word2Vec.load('wvmodel_vkg.model')
    wv=[]
    zero_vector = []
    vector_image_train=[]
    vector_image_val=[]
    wv_d = model.wv.vector_size
    for i in range(0,wv_d):
        zero_vector.append(0)

    b=np.empty([625,100])
    train_data=list()
    test_data=list()
    train_img_set=[]
    val_img_set=[]

    if get_train==True:
```

```
for train_txt in train_txt_list:
    for word in train_txt[0]:

        if word == '#':
            wv.append(zero_vector)
        else:
            c=[]
            for i in range(0,wv_d):
                c.append(model.wv[word][i])
            #print(c)
            wv.append(c)

    tmp = [wv.copy(),train_txt[1]]
    vector_image_train.append(tmp)
    #print(tmp)
    wv.clear()

print(len(vector_image_train))

for text in vector_image_train:
    b = np.array(text[0]).reshape(625,100)
    #print(b)
    train_data.append(b.copy())
    #test_data = test_data.reshape(25,25,100)
    b.fill(0)
#test_data = test_data[0].reshape(25,25)
#test_data=test_data.reshape(25,25)
train_data=np.array(train_data)
train_data=torch.tensor(train_data,dtype=torch.float32)
```

```
        train_data = train_data.permute(0,2,1)
        print(train_data[0])
        print(train_data.shape)
        print(len(train_data))
        for i,train_img in enumerate(train_data):
                tmp = [train_img,torch.tensor(train_tag_list[i])]
                train_img_set.append(tmp)
if get_val == True:
        for val_txt in val_txt_list:
                for word in val_txt[0]:


                        if word == '#':
                                wv.append(zero_vector)
                        else:
                                c=[]
                                for i in range(0,wv_d):
                                        c.append(model.wv[word][i])
                                #print(c)
                                wv.append(c)


                tmp = [wv.copy(),val_txt[1]]
                vector_image_val.append(tmp)
                #print(tmp)
                wv.clear()


        print(len(vector_image_val))
        for text in vector_image_val:
                b = np.array(text[0]).reshape(625,100)
                #print(b)
                test_data.append(b.copy())
                #test_data = test_data.reshape(25,25,100)
                b.fill(0)
        #test_data = test_data[0].reshape(25,25)
        #test_data=test_data.reshape(25,25)
        test_data=np.array(test_data)
        test_data=torch.tensor(test_data,dtype=torch.float32)
        test_data = test_data.permute(0,2,1)
        print(test_data.shape)
        print(len(test_data))

        for i,val_img in enumerate(test_data):
                tmp = [val_img,torch.tensor(val_tag_list[i])]
                val_img_set.append(tmp)

return train_img_set,val_img_set
```

A-4 Naïve Bayes

```
#coding: utf-8
import torch
from torchvision import transforms,utils
from torchvision import datasets
from torch.utils import data
import matplotlib.pyplot as plt
from pprint import pprint
from gensim.models import word2vec
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
```

```
def txt_loader(input):
    item = open(input,'r',encoding='utf-8').read()

    return item

def txt_split(loader_in):

    txt_list=[]
    for txt in loader_in.dataset:
        txt = list(txt)
        txt[0] = txt[0].rstrip()
        txt[0] = txt[0].replace('\n','')
        #txt[0] = txt[0].split(' ')
        txt_list.append(txt[0])

    return txt_list

def get_tag(loader_in):

    tag_list=[]
    for txt in loader_in.dataset:
        txt = list(txt)
        tag_list.append(txt[1])

    return tag_list

def naivebayes():
    my_data =
datasets.DatasetFolder('DATA_PATH',loader=txt_loader,extensions="txt",transform=None)
    n_val = int(len(my_data)/5)
    n_train = len(my_data)-n_val
    print(n_val,n_train)
    train_set,val_set = data.random_split(my_data,[n_train,n_val])
    train_loader = data.DataLoader(train_set,batch_size=2,shuffle=True)
    val_loader = data.DataLoader(val_set,batch_size=2,shuffle=True)



    x_train=txt_split(train_loader)
    x_val = txt_split(val_loader)
    y_train=get_tag(train_loader)
    y_val = get_tag(val_loader)

    tf=TfidfVectorizer()
    x_train=tf.fit_transform(x_train)
    x_val=tf.transform(x_val)

    mlt=MultinomialNB(alpha=1.0)
    mlt.fit(x_train,y_train)
    y_predict = mlt.predict(x_val)
    print(y_predict)
    score= mlt.score(x_val,y_val)
    print("accuracy is ： ",score)
if __name__ == '__main__':
    naivebayes()
```

業績リスト

1. 著書
なし

2. 学会誌又は学術雑誌への論文掲載

I. 論文発表
1) Grad-CAM-Based Explainable Artificial Intelligence Related to Medical Text Processing. Zhang Hongjian.; Ogasawara Katsuhiko. Bioengineering 2023, 10, 1070.
2) Extraction and Quantification of Words Representing Degrees of Diseases: Combining the Fuzzy C-Means Method and Gaussian Membership, Han F, Zhang Z, Zhang H, Nakaya J, Kudo K, Ogasawara K. JMIR Form Res 2022;6(11):e38677
3) Examining the Effect of the Ratio of Biomedical Domain to General Domain Data in Corpus in Biomedical Literature Mining, Zhang, Ziheng, Feng Han, Hongjian Zhang, Tomohiro Aoki, Katsuhiko Ogasawara. *Applied Sciences* 12, no. 1: 154,2021
4) Developing a RadLex-Based Named Entity Recognition Tool for Mining Textual Radiology Reports: Development and Performance Evaluation Study, Tsuji S, Wen A, Takahashi N, Zhang H, Ogasawara K, Jiang G. J Med Internet Res 2021;23(10):e25378
5) Cost-Effectiveness of a Continuous Glucose Monitoring Mobile App for Patients With Type 2 Diabetes Mellitus: Analysis Simulation , Tsuji S, Ishikawa T, Morii Y, Zhang H, Suzuki T, Tanikawa T, Nakaya J, Ogasawara K, J Med Internet Res 2020;22(9):e16053

II. 口頭発表
1) Grad-CAM-Based Explainable Artificial Intelligence Related to Medical Text Processing. Zhang Hongjian, Ogasawara Katsuhiko, 2023FHS, Sapporo,October ,2023.
2) Grad-CAM による説明可能な人工知能による医療文書処理. 張 洪健, 小笠原 克彦, , 医療情報学会第 21 回北海道支部会学術大会,札幌市,2023 年 9 月.
3) 画像識別によるテキスト分類の転移学習-BERT による医療テキストの画像変換-,張 洪健, 小笠原 克彦 ,医療情報学会第 20 回北海道支部会学術大会,札幌市,2023 年 2 月.
4) 医療テキストを画像に変換し、ResNet を適用する転移学習. 張 洪健, 小笠原 克彦, 第 42 回医療情報連合大会,札幌市,2022 年 11 月.
5) 放射線医学領域の標準用語集を拡張する機械学習のパラメータの検討. 張 洪健, 辻 真太朗,Andrew Wen, 蒋 国謙, 曹 瀛丹, 小笠原 克彦,第 39 回医療情報連合大会,千葉市,2019 年 11 月

3. 総説・解説
なし

4. 学会賞・学術賞の受賞
なし

5. その他
なし