Predicting the convergence of large sparse matrices in BiCG algorithm based on computer vision

Cui Wei

Master's Program in Informatics
Degree Programs in Comprehensive Human Sciences
Graduate School of Comprehensive Human Sciences
University of Tsukuba
March 2023

Predicting the convergence of large sparse matrices in BiCG algorithm based on computer vision

Name: Cui Wei

BiCG algorithm is an iterative method for solving the system of linear equations, and its convergence is hard to know before computing. A methodology for predicting the convergence of the BiCG algorithm through image recognition was proposed and validated by the previous research, but still needs more improvements.

Firstly, the methodology was validated just by only one type of convolutional neural network. Secondly, to improve the accuracy, the previous research only focused on changing the methods of visualizing the matrices; as the result, the accuracy on this dataset with 875 matrix samples has reached 80% in almost every control group and 5-fold validation groups, but the results of accuracy of proposed methods are so closed and hard to say they are useful for increasing accuracy. What's more, the CNN used in the previous research is a relatively old CNN called LeNet, and the accuracy on positive samples is low as 53.4% on average.

My research purpose is exactly to improve these shortcomings, and I want to improve the previous research on the same dataset by the following three dimensions:

(1)To introduce and adjust more convolutional networks, for checking if the proposed methodology only works on the structure used by the previous research.

(2)the previous research proposed 2 kinds of visualization methods, and 4 different solutions for the pixel, and thought they change accuracy. But I think they don't change the accuracy obviously and need to be validated if they are really useful for increasing accuracy.

(3)On the dataset which calculated convergence(label) by using the residual of 10^{-10}, the previous research could only have a relatively low accuracy of 53.4% on average. I want to improve the low accuracy when the previous research predicts positive(convergent) samples.

To achieve these purposes, I import 3 convolutional neural networks of different structures. They are ShuffleNetV2, SqueezeNet, and ResNet50. After adjusting their parameters, I use them through the same methodology and dataset of the previous research and compare their result and the result gotten by the previous research. And, by validating the proposed methodology through 3 imported CNNs, it turns out that the methodology is still working on 3 imported CNNs, and the methods of the previous research for increasing accuracy indeed don't work obviously by comparison. Besides, the prediction accuracy of positive samples has been improved from 53.4% to 61.9% by ResNet50.

Main Academic Advisor: Hidehiko HASEGAWA
Secondary Academic Advisor: Haitao YU

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 The motivation . . . 2
- 1.3 Thesis overview . . . 4

2 Related Work **5**
- 2.1 Handwritten digit recognition . . . 5
- 2.2 The application of image recognition technology in medical images . . . 6
- 2.3 Using convolutional neural networks to determine the storage format of large sparse matrices . . . 6

3 Background Theory **7**
- 3.1 BiCG algorithm . . . 7
  - 3.1.1 Convolutional Neural Network . . . 9
- 3.2 Convolution layer . . . 11
  - 3.2.1 Pooling layer . . . 12
  - 3.2.2 Fully connected layer . . . 13

4 The Previous Research: Predicting the convergence of BiCG method from grayscale matrix images **16**
  - 4.0.1 Obtain the labels' values of the matrix by BiCG algorithm . . . 17
  - 4.0.2 Visualization of the matrix to obtain an image of the matrix . . . 17
- 4.1 Dataset . . . 19
- 4.2 Training model and validation . . . 19
  - 4.2.1 K-fold cross validation . . . 19
- 4.3 Results . . . 19
- 4.4 Summary of the previous research . . . 20

5 Fundamental knowledge of the improvement method **21**
- 5.1 Several CNN structures of computer vision . . . 21
  - 5.1.1 ShuffleNetV2:A light-weight neural network . . . 21
  - 5.1.2 SqueezeNet: Another light-weight neural network . . . 22
  - 5.1.3 ResNet50 . . . 22

6 Predicting the convergence of large sparse matrices in BiCG algorithm based on computer vision **24**
6.1 Evaluation of two visualization methods through cosine similarity 25
6.2 The comparison of different CNN structures 26
6.3 Comparison of prediction accuracy between models 26
6.4 Performance on positive data prediction 29

7 Conclusion **30**

Acknowledgements **31**

References **32**

List of Figures

3.1 The scheme of Convolutional Neural Network 10
3.2 The convolution 11
3.3 The pooling 12
3.4 Fully connected layer 14

4.1 The process of the previous research 16

6.1 The main process of this research 24
6.2 Similarity between two visualization methods 26

List of Tables

4.1 The number of samples in 2 classes 19
4.2 Sample size of each group 19
4.3 The accuracy of baseline 20
4.4 28×28 pixel's prediction by baseline 20

6.1 The parameter of CNNs used 25
6.2 The accuracy of baseline 26
6.3 5 groups of 4 different models under 28×28 pixel resolution 27
6.4 5 groups of 4 different models under 56×56 pixel resolution 27
6.5 5 groups of 4 different models under 112×112 pixel resolution 28
6.6 5 groups of 4 different models under 224×224 pixel resolution 28
6.7 The number and ratio of TP in prediction, 28×28 pixel resolution, numbers in parentheses are ratios 29

Chapter 1

Introduction

1.1 Background

The problem of solving a large system of linear equations is widespread in current research and industry. For example, when fluid dynamics is applied to engineering problems, there will be occasions when a system of partial differential equations needs to be solved. There is no good mathematical method for solving the analytical solution of the partial differential equations, so the numerical solution of the partial differential equations is often used for solving engineering problems. This is a method to discretize the partial differential equation, and the representative methods are the finite difference method, boundary element method, etc. After discretization, the problem of solving partial differential equations eventually transforms into the problem of solving a large system of linear equations.

The following system of linear equations exists

$$Ax = b \tag{1.1}$$

where A is a large n-dimensional sparse matrix, x is the unknown vector to be solved, and b is the known vector. And how to compute this system of linear equations quickly and inexpensively is a very critical problem.

There are two types of general methods, which are direct solution methods and iterative solution methods. The direct solution methods are represented by the Gaussian elimination method, which is based on the idea of using the primary transformation of the matrix to eliminate the unknown quantities in order to transform A into an echelon matrix, and then solving all the elements in x one by one by substituting b. However, in numerical computation, this method is considered to be time-consuming and memory-intensive.

Instead, the iterative methods are used more often. they begin by setting an initial value of x and iterating through it so that the difference between Ax and b gets closer and closer to a predetermined value measured by residual. These methods are very widely used in scenarios where a computer is used to solve a system of linear equations. A representative method of this type is the BiCG [1] algorithm.

But BiCG also has its problem, especially in the case of solving large systems of linear equations, where a sparse matrix does not converge, time would be consumed very much. So, that is the previous research's motivation: If one can predict in advance whether the

BiCG algorithm can solve a certain system of linear equations (i.e., whether the corresponding sparse matrix converges or not), then one can avoid solving those systems of linear equations that do not converge.

Computer vision is a branch of artificial intelligence that studies how to use computers to understand and process image and video data. It involves a variety of different tasks, such as image classification, target detection, image segmentation, and image generation.

Computer vision has a wide range of applications in many different fields, such as autonomous driving, robotics, image search, medical image analysis, translation, etc. To solve these problems, computer vision researchers use a variety of machine learning algorithms, including convolutional neural networks (CNNs), deep learning, and autoencoders.

This research and the previous research hope to use convolutional neural networks to extract features in sparse matrices to help predict the convergence of the BiCG algorithm.

1.2 The motivation

This research complements and extends the previous research by Ota [2]. In this paper, I will always use the previous research to name his research.

In the previous research, a method based on image recognition is proposed to predict the convergence of the BiCG algorithm.

The BiCG algorithm is a popular algorithm for solving the system of linear equations. Because it converges quickly, it has been one of the most preferred algorithms in many cases.

When using the BiCG algorithm, the residual is used to measure the convergence of the algorithm. As the algorithm iterates, the residual decreases and eventually reaches a predetermined threshold, indicating that the algorithm has reached convergence. In general, the smaller the residual is set, the higher the precision computed, but the more time-consuming.

To make a classifier that can predict if the BiCG algorithm will be convergent or not on a system of linear equations before calculating it, the previous research collected 982 non-symmetric real sparse matrices from the SuiteSparse Matrix Collection website and used 875 of them to produce the dataset: he first calculated their convergence using the BiCG algorithm and tag their labels by convergent(label to 1) or divergent(label to 0); then visualized these systems of linear equations(but just the sparse matrices of them) into grayscale images, to let them become the fitting inputs of the convolutional neural network. And, used the 5-fold cross-validation to check the accuracy of predicting.

The above is the previous research's method. As the purpose of the previous research, the previous research wants to generate classifiers that could predict the convergence with an accuracy higher as possible. To reach this purpose, the previous research used several methods to generate images of matrices of datasets, such as using two visualization methods to generate matrix images, For calculating the labels(convergence) of datasets, the previous did many trials, as we can see in his control test group as follows:

- Convergence is computed and labeled with the residual of 10^{-6}. In this case, there are 235 matrix images labeled Positive (convergent), and 640 matrix images labeled

Negative(divergent)

- Convergence is computed with and labeled with the residual of 10^{-6}. But in this case, the previous research used a kind of matrix C called pre-conditioner multiplied simultaneously for both sides of the $Ax = b$, calculated new equations like $CAx = Cb$, and labeled their convergence, in this case, there are 289 matrix images labeled to Positive (convergent), and 586 matrix images labeled to Negative(divergent).

- To equalize or balance the number of negative samples and positive samples in the datasets, the previous research extracted a certain percentage of the matrices in the above control group to form the dataset with both Negative and Positive of 235, and the total size of this dataset became 470.

- The convergence was recalculated at residuals of 10^{-10} and labeled, with a total of 176 Positive (convergent) data and 699 divergent.

I am particularly interested in the last control group because the dataset of this group is made by a lower residual. As I mentioned before, a lower residual makes the BiCG algorithm more precise, but, the experiment of this control group is a very small part of previous research, and needs to be extended. What's more, the prediction accuracy is not very good, especially on True positive samples.

Another shortcoming of the previous research is, to improve accuracy, the previous research always used methods like changing the residual, changing the visualization method, or resetting labels in many ways, all of these methods can be seen as the pre-processing methods in the machine learning perspective. Plus, through this machine learning perspective, the previous research used only one structure of the neural network but didn't do more trials in this dimension.

Indeed, there is still a more fundamental question that the previous research did not explicitly address: does this method of using image recognition techniques to predict the convergence of the BiCG algorithm can only work on the CNN used by the previous research? Or it also works on other convolutional neural network structures as well.

Therefore, in this research, to improve the accuracy of the dataset of $residual = 10^{-10}$, and figure out the question I mentioned above, I would import more neural network structures, try to modify them, and evaluate the methods of previous research. Actually, many neural networks structure have been widely used in computer vision fields such as handwritten digit recognition and medical image detection, because these fields are similar to the previous research in the image recognition part, I thought the neural network structure with good performance on these fields would be promising in this subject.

As a result, it turns out that previous research's method is also working on other convolutional neural networks as well. Some of these structures even have higher accuracy between groups during the k-fold (5-fold in this research) cross-validation. And what's more, this computer vision's perspective-based method did improve the accuracy of prediction in the group made by the residual of 10^{-10}

The dataset used in this research was produced and provided by the previous research, and I would like to thank him for his work.

1.3 Thesis overview

In Chapter 2, I summarise the research related to the application of image recognition. The processing of the data in this study follows a similar process to these studies.
In Chapter 3, I introduce the two theories that are most important for this study. The first is the BiCG algorithm, which is both the subject of this study and the tool used to calculate the convergence (labeled values) of the data for this study; the second is the convolutional neural network, which is the primary research method for this study. However different the classifiers used in this study and the prior studies may be, they are both in essence convolutional neural networks.

In Chapter 4, I present the previous research of this study, including its idea, its method, its implementation process, and its results. In addition, the datasets used in this study and the prior research, and the way they were grouped in the k-fold cross-validation also appear in this chapter.

In Chapter 5, I introduce three new convolutional neural networks with different structures that were introduced in this study to validate and improve the methodology of the previous research. They are ShuffleNetV2, SqueezeNet, and ResNet50.

In Chapter 6, I present experimental data comparing the results of the new neural network structures introduced with those of the previous research. The main purpose is to show that the methods proposed in the previous research to improve accuracy are of little help, such as why the two different pictorial methods are not helpful for prediction, and that resolution is far from being a decisive factor. Finally, it is given to what extent the network I introduced and adapted outperforms the previous research in the dataset where the residuals are 10^{-10} and the labels are calculated.

In Chapter 7, I conclude this study and look at future work.

Chapter 2

Related Work

2.1 Handwritten digit recognition

Handwritten digit recognition is a technology that recognizes handwritten digits. This technology is often used to automate the processing of handwritten documents, such as recognizing amounts on checks in a banking system or recognizing handwritten data in spreadsheet software.

A commonly used method for handwritten digit recognition over these years is to use a convolutional neural network. The network is trained on a large number of handwritten digits and is able to learn the features of different digits and is able to accurately recognize the input handwritten digits based on these features. By continuously adjusting the parameters of the model, it can be made to achieve higher accuracy in recognizing new handwritten digits.

In the research of LeCun et al [3]. A method for recognizing handwritten digits using a backpropagation network (i.e., a multilayer perceptron) is described. The paper proposes using each pixel of the training sample as an input to the neural network and training the network using supervised learning methods to map the input to the correct digit labels. The paper also proposes a method called the backpropagation algorithm to adjust the network weights to minimize the training error.

They described the use of convolutional neural networks (CNNs) to recognize handwritten digits. The paper describes methods for converting images into convolutional layers with multiple feature maps and using a maximum pooling layer to reduce the spatial size of the image. The paper also proposes a method to improve recognition accuracy using deep CNNs with multiple convolutional-maximum pooling layers stacked together. This is a very early study of handwritten digit recognition using convolutional neural networks, in which the research method with the MNIST dataset has been followed in the present-day handwritten digit recognition techniques. Moreover, its research method has been widely applied to other image recognition fields.

2.2 The application of image recognition technology in medical images

Image recognition technology has many applications in disease diagnosis. Image recognition technology can be used to recognize medical images, such as CT scans and X-rays. With these images, doctors can diagnose a patient's disease more accurately. For example, with CT scan images, doctors can detect the presence of tumors in the lungs. At the same time, image recognition technology can also be used to screen for diseases, helping doctors to detect potential diseases earlier and take effective treatment measures. Overall, image recognition technology is very useful in disease diagnosis, improving the accuracy and timeliness of diagnosis and helping doctors to better treat diseases.

Altaf's paper [4] presents the application of deep learning in medical image analysis and discusses its strengths, challenges, and directions for development. It also outlines common approaches to deep learning in medical image analysis, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs).

The paper also explores the challenges faced when using deep learning methods for medical image analysis, including data bias, missing labels, and model generalization capabilities. Finally, the paper also looks at the future direction of deep learning in medical image analysis.

In the research of Kallenberg et al. [5], A new approach to the analysis of breast density using unsupervised learning methods is proposed. The method uses a convolutional neural network (CNN) model and uses an unsupervised clustering method for segmentation. The paper also describes how the method can be used to risk score mammograms.

Experiments are conducted to validate the method and compare it with other methods. The experimental results show that the accuracy of breast density segmentation and risk scoring of mammograms using this method is high.

2.3 Using convolutional neural networks to determine the storage format of large sparse matrices

Researchers believe that the multiplication of sparse matrices can be accelerated if a suitable storage format is chosen for the sparse matrix.

So in the research by Cui [6] et al. deep learning techniques were used to automatically select the storage format for sparse matrices. Cui et al decided on the label of the matrix, i.e., the storage format, by comparing their time performance in coefficient matrix multiplication among several formats such as COO, CSR, BSR, and ELL. After that, convolutional neural networks are used to learn the image distribution of the matrix in relation to its label values to obtain the classifier.

Chapter 3

Background Theory

First I will introduce two of the most important theories in previous research: the BiCG algorithm and convolutional neural networks(CNNs).

3.1 BiCG algorithm

In numerical computation, when faced with the problem of solving a system of linear equations, it is preferable to use some iterative solution methods than using Gaussian elimination methods.

I think the advantages of this method are the control-possible precision due to the use of residuals to measure the distance between the solution set and the true solution, and the promise of completing the solution in a finite number of iterations.

The BiCG(biconjugate gradient) algorithm is an iterative method used to solve systems of linear equations. it is a variant of the conjugate gradient method. Like the conjugate gradient method, it is typically used to solve large, sparse systems of linear equations.

The basic idea behind the BiCG algorithm is to find a solution to a system of linear equations by constructing a sequence of approximate solutions, called "iterates", that converge to the exact solution. This is done by using information from the previous iterates to construct a new iterate that is "closer" to the exact solution. The algorithm continues in this manner until the solution is found to be within a specified tolerance.

One of the key features of the BiCG algorithm is that it uses a combination of two different conjugate vectors to construct each new iterate. This allows the algorithm to converge faster than other iterative methods, such as the gradient descent method.

The above process can be described as follows

$$\alpha_k = \frac{r_k^T r_k}{p_k^T q_k}$$
$$x_{k+1} = x_k + \alpha_k p_k$$
$$r_{k+1} = r_k - \alpha_k q_k$$
$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$
$$p_{k+1} = r_{k+1} + \beta_k p_k$$
$$q_{k+1} = A p_{k+1}$$

And, to implement the BiCG algorithm, the algorithm can be written in a few simple steps:

Algorithm 1 The Step of BiCG

1. Initialize the algorithm by selecting a starting point (i.e., an initial guess for the solution) and setting the iteration

2. counter to 0.

3. Compute the residual vector, which is the difference between the right-hand side of the system of equations and the product of the coefficient matrix and the current iterate.

4. Compute the direction vector using the residual vector and the previous direction vector.

5. Update the current iterate by adding a multiple of the direction vector to it.

6. Update the iteration counter and check if the current iterate is close enough to the exact solution. If it is, stop and return the current iterate as the solution. Otherwise, go back to step 2 and continue the iteration.

Some of the key features of the BiCG algorithm include:

- BiCG is an iterative algorithm, which means that it generates a sequence of approximate solutions to a system of linear equations. It does not directly compute the exact solution, but it can often produce solutions that are accurate enough for many applications.
- BiCG is a variant of the Conjugate Gradient (CG) algorithm, which means that it uses the same principle of minimizing the residual error at each iteration. However, BiCG uses a different formula to update the search direction at each iteration, which can make it more effective for certain types of problems.
- BiCG can be used in a variety of applications, including linear regression, image processing, and data analysis.
- BiCG is particularly well-suited for solving large, sparse systems of equations. This is because it uses iterative methods, which can be more efficient than direct methods for solving such systems.

Some of the advantages of the BiCG algorithm include:

- BiCG is an efficient algorithm for solving large, sparse systems of linear equations. This makes it well-suited for applications such as data analysis and image processing, where such systems are common.
- BiCG is relatively easy to implement, which makes it accessible to a wide range of users.
- BiCG can be used in a variety of applications, including linear regression, image processing, and data analysis.

Some of the disadvantages of the BiCG algorithm include:

- BiCG is an iterative algorithm, which means that it does not directly compute the exact solution to a system of linear equations. Instead, it generates a sequence of approximate solutions, which can be less accurate than the exact solution.
- BiCG can be sensitive to the initial starting point, which can affect the convergence of the algorithm. This can make it difficult to use in some situations.
- BiCG may not always converge, or it may converge to a solution that is not accurate enough for the given application. This can be a problem in situations where an accurate solution is required.

3.1.1 Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network that is designed to process data that has a grid-like structure, such as an image. It is called a "convolutional" neural network because it uses a mathematical operation called convolution to filter the input data.

Convolutional neural networks are typically used for image recognition and classification tasks, such as identifying objects in an image or classifying an image as belonging to a certain category. They are also used for other types of data that have a grid-like structure, such as video frames and audio signals.

The basic structure of a CNN consists of an input layer, multiple hidden layers, and an output layer. The hidden layers are made up of a series of convolutional layers, which apply convolutional filters to the input data, and pooling layers, which downsample the data to reduce its dimensionality. The output layer produces the final prediction based on the filtered and downsampled data. As shown in Fig.3.1.

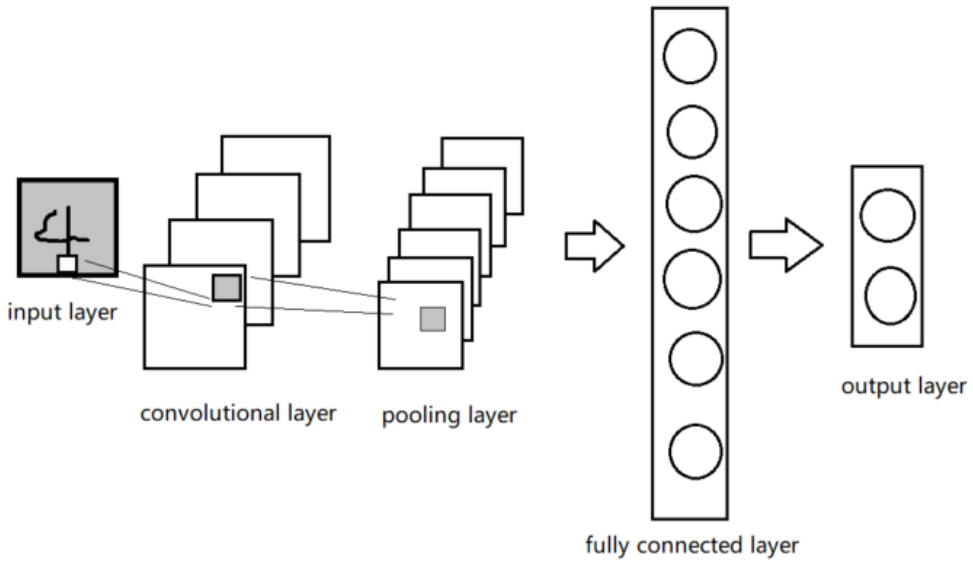

Figure 3.1: The scheme of Convolutional Neural Network

To train a CNN, we first need to provide it with a large dataset of labeled examples. The network uses these examples to learn the patterns and features that are important for making accurate predictions. This is done using a variant of the backpropagation algorithm, which is a widely used method for training neural networks.

Once the network has been trained, we can use it to make predictions on new, unseen data. The network applies the filters and pooling operations learned during training to the new data, and produces a prediction based on the resulting processed data. In summary, a convolutional neural network is a type of artificial neural network that is designed to process grid-like data, such as images. It uses convolutional and pooling layers to learn patterns and features in the data and can be trained to make predictions on new, unseen data.

In the following subsection, I will introduce the structure of a convolutional neural network in the sequence of levels

3.2 Convolution layer

The role of the convolutional layer is to perform convolutional operations on the input data and extract meaningful features by learning parameters.

Specifically, the convolution layer divides the input data into several small blocks, each of which is convolved with a matrix of weights. This weight matrix is called a convolution kernel (also called a filter). Each convolution kernel has a number of weights, which are parameters learned during the training process.

In the convolution operation, the input data of each chunk is dotted with the weights of the corresponding convolution kernel, and then all the dotted results are added up to get a new value, which is the convolution result of this chunk.

The output of the convolution layer is a multi-channel two-dimensional matrix. Each channel corresponds to a convolution kernel, and each convolution kernel corresponds to an output channel. In this way, the convolutional layer can extract many different features at the same time.

The above process can be found in Figure 3.2.

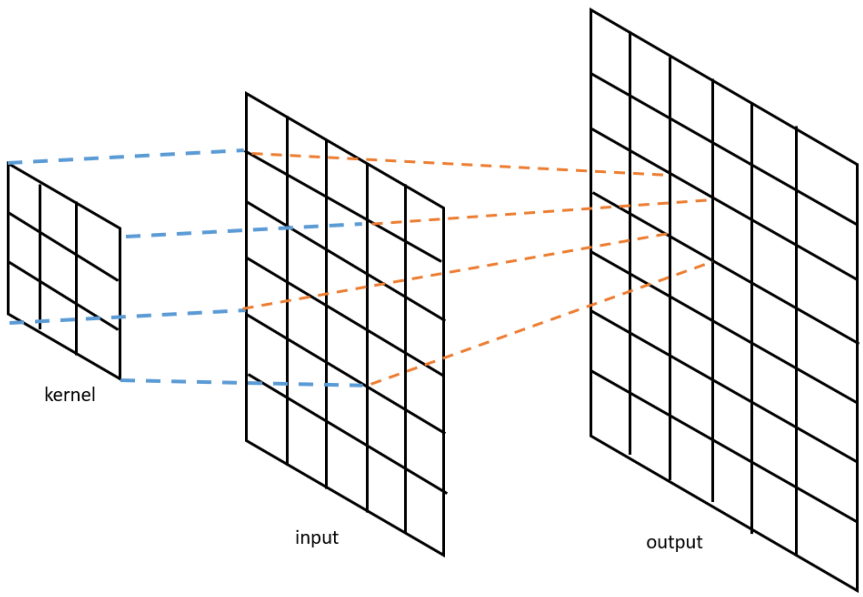

Figure 3.2: The convolution

The main parameters of the convolutional layer are
The shape of the input data: i.e., the length, width, and number of channels of the input data.

A number of convolution kernels: i.e., the number of output channels of the convolution layer.

The size of the convolution kernel: i.e., the length and width of the convolution kernel.

Step size: i.e., the distance that the convolution kernel moves each time on the input data.

Padding: i.e., the number of pixels to be padded at the edges of the input data.

In practice, convolutional layers usually follow a nonlinear activation function, such as a ReLU layer, after the convolutional layers to enhance the representation of the model. Maximum pooling layers can also be inserted between convolutional layers to reduce the number of model parameters and improve the generalization capability.
During the training process, the weight parameters of the convolutional layers are learned automatically by the backpropagation algorithm. The basic idea of the backpropagation algorithm is that, for a given training sample, the output of the model is calculated by forward propagation, then the difference between the output and the true label is calculated, and then the parameters of the model are adjusted in the direction of the difference so that the model can predict the output more accurately in the next forward propagation.

3.2.1 Pooling layer

Pooling layer is a common neural network layer that is mainly used to reduce the size of the input data and extract the important features of the input data at the same time.

The input of the pooling layer is a multi-channel two-dimensional matrix. It divides the input data into several chunks, each of which can be either a two-dimensional matrix or a one-dimensional vector. Then, the pooling layer performs a pooling operation on each chunk to obtain a smaller output.

Common pooling operations include maximum pooling and average pooling. Maximum pooling is to take the maximum value in a small block as the output, while average pooling is to take the average of all elements in a small block as the output.

Their differences in average pooling and maximum pooling can be seen in the example of Figure 3.3.

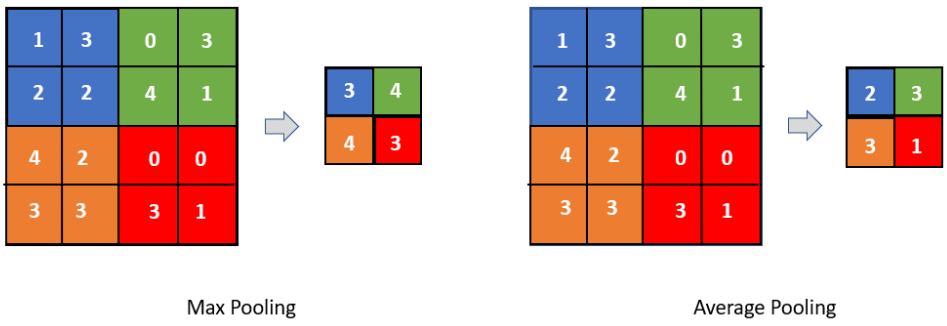

Figure 3.3: The pooling

The main parameters of the pooling layer are:

- The shape of the input data: the length, width, and number of channels of the input data.
- Size of the pooling operation: the length and width of the chunks required for the pooling operation.
- Step size: the distance that the pooling operation moves each time on the input data.
- Padding: the number of pixels to be padded at the edges of the input data.

The output of the pooling layer is a multi-channel two-dimensional matrix, each channel corresponding to one input channel. The pooling layer has no learning parameters and its output is only related to the input.

The pooling layer is usually used following the convolutional layer, which reduces the number of model parameters, reduces the risk of overfitting, and improves generalization ability. Pooling layers can also improve the computational efficiency of the model because reducing the size of the input data means that fewer elements need to be computed as well.

However, pooling layers also have some drawbacks. Since a pooling layer discards some of the information of the input data, it may affect the representation capability of the model. In addition, pooling layers also lead to a reduction in the spatial resolution of the input data, which may affect the model's ability to extract fine-grained features.

3.2.2 Fully connected layer

The fully connected layer (also known as the fully corresponding connected layer or the densely connected layer) is used in many deep learning models.

The input of the fully connected layer is a one-dimensional vector, and its output is also a one-dimensional vector. Between the input layer and output layer, there are one or a dozen of hidden layers, as shown in Fig 3.4.

The main role of the fully connected layer is to perform a linear transformation of the input vector and extract meaningful features by learning parameters.

Specifically, the input vector of the fully connected layer is dotted with a weight matrix and then a bias vector is added to obtain a new vector, which is the output of the fully connected layer. Both this weight matrix and the bias vector are parameters learned during the training process.

The output of the fully connected layer is usually passed through a nonlinear activation function, such as a ReLU function, a sigmoid function, or a softmax function, to enhance

the representation of the model.

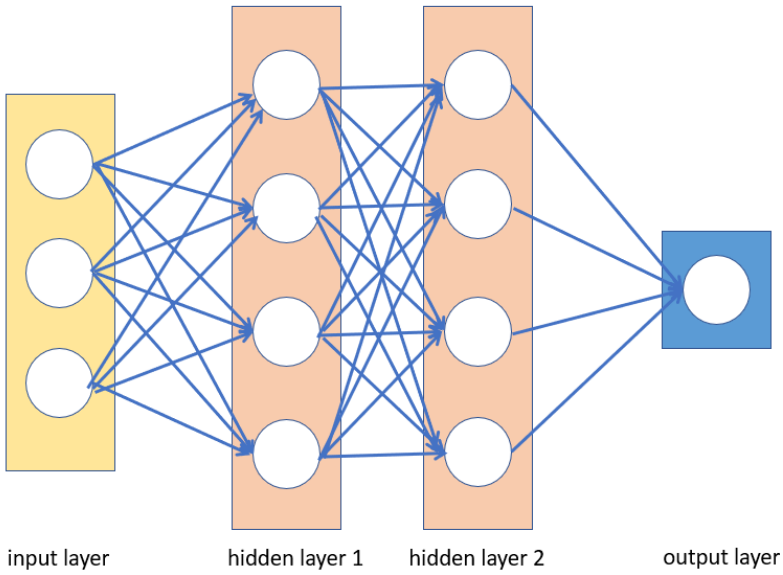

Figure 3.4: Fully connected layer

The main parameters of the fully connected layer are

- The dimension of the input vector: i.e., the number of elements of the input vector.
- The dimension of the output vector: the number of elements of the output vector.

During the training process, the weight matrix and bias vectors of the fully connected layer are learned automatically by the backpropagation algorithm. The basic idea of the backpropagation algorithm is that for a given training sample, the output of the model is calculated by forward propagation, then the difference between the output and the true label is calculated, and then the parameters of the model are adjusted in the direction of the difference so that the model can predict the output more accurately in the next forward propagation.

Fully-connected layers are used in many deep learning models, including multilayer perceptrons, convolutional neural networks, recurrent neural networks, etc. Fully connected layers can be used to extract multiple features of the input data and automatically learn these features during the training process.

Since the number of weight parameters of the fully connected layer is proportional to the dimensionality of the input vector, when the dimensionality of the input vector is large,

the number of parameters of the fully connected layer will also be large, which leads to an increase in the training time and memory consumption of the model. In addition, fully-connected layers do not handle spatial structure information well, so when processing data with spatial structure information such as images, it is usually necessary to use a convolutional layer to extract features.

Chapter 4

The Previous Research: Predicting the convergence of BiCG method from grayscale matrix images

After proposing the idea of obtaining classifiers that can predict the convergence of sparse matrices by training neural networks, the previous research implements its idea through the process shown in Fig 4.1.

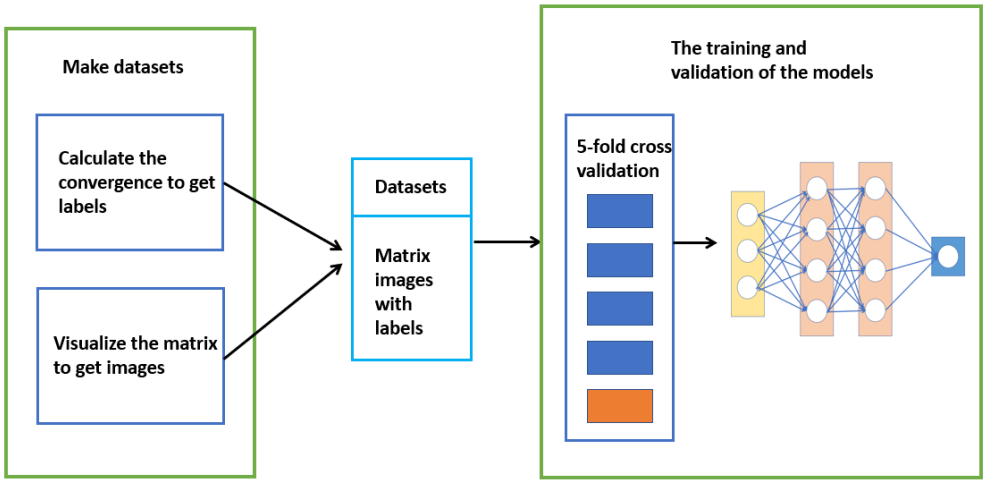

Figure 4.1: The process of the previous research

In the following, I will describe each step of the process shown above:

The dataset used in this research was downloaded by the previous research from the SuitesSparse Matrix Collection website.

The Suitesparse Matrix Collection is an online database that collects a large sample of sparse matrices. These sparse matrices come from a variety of resources, including linear algebra, computer graphics, signal processing, optimization, and machine learning.

The site provides a large sample of sparse matrices that can be used by researchers and developers. The site provides extensive documentation and instructions to help users understand the properties and uses of matrices.

This section will describe the previous research on how to make datasets and also how to transform sparse matrices into labeled ones that can be fed into convolutional neural networks.

4.0.1 Obtain the labels' values of the matrix by BiCG algorithm

In this step, the convergence of $b = Ax$ needs to be solved using the BiCG algorithm. The initial value of x and residual r_0 is set as follows.

$$X_0 = [0, 0, 0, 0, \ldots\ldots, 0]^T \tag{4.1}$$

The residuals in the previous study are 10^{-6} and 10^{-10}, and my research only focuses on the case of 10^{-10}, which is the case as follows

$$||r|||_2 <= 10^{-6}||b|||_2 \tag{4.2}$$

4.0.2 Visualization of the matrix to obtain an image of the matrix

In this section, we will talk about how to convert matrix data into grayscale images. The previous research used two different methods, one is called $SuiteSparse$ and the other is called *Sigmoid*. The idea of both methods is to first take the absolute values of the matrix elements, then use a method of smoothing similar to Gaussian blur to unify the size of the matrix, and finally use different normalization methods to transform the values of the elements to between 0 and 255.

Their algorithms are as follows.

Algorithm 2 SuiteSparse

1. Take the non-zero elements in A to their absolute values, i.e., make them positive.
2. Divide the dimension n of the matrix by the dimension d of the image to be generated, and the number obtained after the cut is denoted as s
3. Divide the matrix by the square of $s \times s$, take the maximum value of each block to represent this matrix, and reconstruct a new matrix $A^{'}$ in this way.
4. Take the common logarithm of the non-zero elements of $A^{'}$
5. Find the standard deviation σ of the median Me of the non-zero elements for which the common logarithm is found.
6. Transform the elements $a^{'}_{i,j}$ in $A^{'}$ by the following equation:

$$grayscale = \begin{cases} 255 & Me + \sigma < \log_{10}(a^{'}_{i,j}) \\ 128 + [127\frac{log_{10}(a^{'}_{i,j}) - Me}{\sigma}] & Me - \sigma \leq log_{10}(a^{'}_{i,j}) \leq Me + \sigma \\ 0 & Me - \sigma > log_{10}(a^{'}_{i,j}) \end{cases} \tag{4.3}$$

7. If the matrix $[\frac{n}{s}]$ is not an integer, the image needs to be expanded and the lost quality is corrected with bicubic

Algorithm 3 Sigmoid

1. Take the non-zero elements in A to their absolute values, i.e., make them positive.
2. Divide the dimension n of the matrix by the dimension d of the image to be generated, and the number obtained after the cut is denoted as s
3. Divide the matrix by the square of $s \times s$, take the maximum value of each block to represent this matrix, and reconstruct a new matrix $A^{'}$ in this way.
4. Take the common logarithm of the non-zero elements of $A^{'}$
5. Find the mean μ, and the standard deviation σ for the non-zero elements of the over common logarithm
6. $u = \log_{10} nonzero(A^{'})$, and use the following formula for normalization

$$u_i = \frac{u_i - \mu}{\sigma} \tag{4.4}$$

7. The normalized values in 6 are normalized to between 0 and 1 using the sigmoid function, and then multiplied by 255 (and round) to obtain the grayscale value
8. If the matrix $[\frac{n}{s}]$ is not an integer, the image needs to be expanded and the lost quality is corrected with bicubic

4.1 Dataset

This research wanted to examine the control group of the previous research at a residual of 10^{-10} using multiple computer vision. This is a dataset with a slightly disproportionate ratio of negative to positive samples. There are 176 negative data and 699 positive data, as shown in the following Table 4.1.

Table 4.1: The number of samples in 2 classes

positive samples	negative samples	account
176(20%)	699(80.0%)	875

To control the previous research, they were grouped exactly the same as in the previous research. The ratios are shown in Table 4.2.

Table 4.2: Sample size of each group

	group1	group2	group3	group4	group5
negative(divergent)	140	140	140	140	139
positive(convergent)	35	35	35	35	36

4.2 Training model and validation

Due to the small data set, k-fold cross-validation was used to test the model in the previous research. This method is also used in this research

4.2.1 K-fold cross validation

K-fold cross-validation is a method used to evaluate the performance of a model. It does this by dividing the dataset into K different subsets, using one subset at a time as the test set and the remaining K-1 subsets as the training set for training and evaluation, so that the evaluation results of K models can be obtained. The final model performance evaluation result is the average of the K evaluation results.

4.3 Results

In the dataset produced with a residual of 10^{-10}, which is of most interest to my study, the results obtained by the previous research are shown in the following table 4.3. and table 4.4.

Although the accuracy in the table reaches 85%, the accuracy of the prediction in the positive sample is relatively low. It is only the low percentage of the positive samples occupying the whole data set that causes the overall higher precision. The predicted data for the labeled samples are as follows, and the visualization method used for the dataset used in this table is SuiteSparse.

Table 4.3: The accuracy of baseline

size	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
28×28	84.0	86.2	86.2	86.8	81.7	85.0
56×56	87.4	86.8	86.2	86.8	81.7	85.7
112×112	86.2	86.8	88.0	88.0	81.7	86.1
224×224	86.8	82.8	86.8	86.8	81.7	85.0

Table 4.4: 28×28 pixel's prediction by baseline

	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)	
TN	129(92.1)	129(92.1)	135(96.4)	127(90.7)	130(92.8)	650(93.5)
FN	17	13	19	10	23	82
FP	11	11	5	13	9	49
TP	18(51.4)	22(62.8)	16(45.7)	25(71.4)	13(36.1)	94(53.4)
Total	175	175)	175	175	175	875

4.4 Summary of the previous research

In previous research, the authors proposed a method to predict the convergence of the BiCG algorithm using convolutional neural networks. After computing the system of linear equations with two residuals of 10^{-6} and 10^{-10}, labels of convergence or non-convergence are given for the corresponding sparse matrices.

Afterward, two visualization methods, SuiteSparse and Sigmoid, were used to transform the matrix datasets into grayscale images which are suitable for loading into the convolutional neural network. The previous research used methods such as changing the visualization method, scaling, and changing the residual accuracy to try to improve the accuracy of the prediction. However, from the machine learning perspective, these are all pre-processing methods for the data and the authors did not try to use more neural network structures for improvement.

In addition, the prediction accuracy for the positive samples was only 53.4 % on average in the dataset generated by increasing the residual accuracy to 10^{-10}.

These shortcomings are what my research hopes to improve.

In the next chapter, I will introduce the new structure of the CNNs I used.

Chapter 5

Fundamental knowledge of the improvement method

The new methods introduced in this research are mainly new CNNs of computer vision. Considering that the image resolutions of pixels in the previous research are 28×28, 56×56, 112×112, 224×224, and most of the CNN are applicable to images with resolutions above 224×224, the lightweight networks ShuffleNetV2, SqueezeNet; and ResNet50 which is applicable to small size images.

5.1 Several CNN structures of computer vision

ShuffleNet V2 is a lightweight convolutional neural network structure. ShuffleNet V2 uses an operation called "shuffle" to reduce the interaction between channels, thus reducing the computational complexity.

SqueezeNet is also a lightweight convolutional neural network structure that can be used especially for mobile devices. SqueezeNet uses a module called "fire" to extract features and reduces the number of parameters in the model by reducing the size of the convolutional kernel.

ResNet50 is a deep convolutional neural network structure for computer vision applications. ResNet50 solves the gradient disappearance problem in deep neural networks by using residual connections to train deeper networks. ResNet50 has achieved high accuracy on the ImageNet dataset and is widely used in computer vision.

5.1.1 ShuffleNetV2:A light-weight neural network

ShuffleNet V2 was presented at the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).Ma [7] proposed the ShuffleNet V2 structure and compared the impact of different design parameters on the network performance. The authors also conducted experiments using a variety of datasets and demonstrated that ShuffleNet V2 provides high accuracy in computer vision applications while maintaining low computational complexity and memory footprint.

It uses "shuffle", "pointwise group convolution" and "channel shuffle" operations to reduce computational complexity and memory consumption, and "bottleneck design" to improve network performance. It is suitable for computer vision applications, especially for mobile devices, and has achieved good results in many applications.

"Shuffle" is an operation used to reduce the interaction between channels, which reduces the computational complexity. This operation is achieved by dividing the channels into groups and then swapping them within the groups. For example, if a channel is divided into two groups, the channel in the first group can be swapped with the channel in the second group. This reduces the interaction between channels and thus reduces the computational complexity.

The "pointwise group convolution" divides the channels into multiple groups and convolves each group with a convolution kernel. For example, if the channels are divided into two groups, two convolution kernels can be used for convolution. This reduces the number of parameters and thus the memory footprint of the model.

The "channel shuffle" is implemented by randomly disrupting the order of channels between each layer. This increases the generalization ability of the model.

In addition, ShuffleNet V2 also uses a design called "bottleneck design" to improve the performance of the network. This design reduces the number of parameters by using a small convolutional kernel between each layer and reduces the output size by using a convolutional kernel with a step of 2, thus reducing the computational complexity.

5.1.2 SqueezeNet: Another light-weight neural network

SqueezeNet was proposed by FN Iandola and other researchers [8] in the paper "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size" published in 2016.

It is a small convolutional neural network whose structure is designed to minimize the number of parameters and computational effort of the model while maintaining excellent classification performance. It is commonly used for image classification tasks and works well on many resource-constrained devices.

The network structure consists of a series of convolutional and pooling layers that contain a number of components called "Fire" blocks. Each Fire block consists of a "Squeeze" layer and two "Expand" layers. layer uses a set of 1x1 convolutional kernels and a set of 3x3 convolutional kernels to increase the number of output channels.

In addition, it uses a technique called "deep separable convolution", which uses fewer parameters for convolution and has higher computational efficiency. In addition, SqueezeNet uses skip connections (also known as short-circuit connections) to avoid losing information in the network.

5.1.3 ResNet50

Resnet50 is a deep neural network that derives its name "ResNet50" from the fact that it has 50 convolutional layers by He [9].

The structure of ResNet50 consists of a number of residual blocks, each of which contains a set of convolutional and batch normalization layers. The input of each residual block is convolved and batch normalized, and then the input itself is added as the output. This type of connection is called a cross-layer connection or residual connection.

The purpose of using cross-layer connectivity is to solve the problem of gradient disappearance in deep neural networks. In deep neural networks, as the number of layers increases, the gradient usually becomes smaller and smaller, leading to poorer training results. Using cross-layer connectivity allows the network to learn the residual function instead of the original function, which helps solve the degradation problem in deep neural networks and allows the network to converge faster.

ResNet50 also uses a batch normalization layer to control the bias and variance of the activation function in the network. The batch normalization layer calculates the mean and variance on each batch of training data and then normalizes the input data by subtracting the mean and dividing it by the variance. Using a batch normalization layer can help prevent overfitting and allow the network to generalize better to new data.

Chapter 6

Predicting the convergence of large sparse matrices in BiCG algorithm based on computer vision

In the experiments, I will first measure the similarity between the two visualization methods, SuiteSparse and Sigmoid, using cosine similarity to explain the question posed by the previous research -Why is the final classification accuracy of the two visualization methods similar?

Second, I will measure the accuracy of all the neural network structures used, taking the four resolutions' situations respectively to show that the classification accuracy does not differ significantly in this range of 28 to 224 resolutions.

Finally, I will give the performance of each neural network structure for the predictions of Positive Samples at 28×28 resolution. to show whether the neural network structure I used in my research is an improvement over the previous research.

Since the experiments are mainly focused on training the neural network structures, I will give my flow in this process as Fig 6.1.

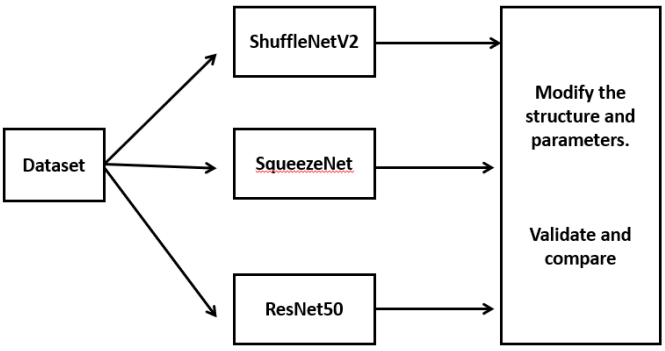

Figure 6.1: The main process of this research

The experimental procedure of this research is shown above, the datasets were loaded into ShuffleNetV2, SqueezeNet, and ResNet50 respectively for 5-fold cross-validation.

In order to better compare the accuracy of each group, I used the dataset with groupings that were consistent with the previous research

The following Table 6.1. is the parameter of the neural network I used.

Table 6.1: The parameter of CNNs used

CNNs	ShuffleNet V2	SqueezeNet	ResNet50
Loss function	Cross Entropy Loss	Cross Entropy Loss	Cross Entropy Loss
Optimizer	Adam	Adam	Adam
Learning rate	1×10^{-3}	1×10^{-4}	5×10^{-5}

6.1 Evaluation of two visualization methods through cosine similarity

The authors of the previous research concluded that there was not much difference in the accuracy of the predictions obtained by the two proposed methods of visualizing the matrix.

So, in this research, I first tested the cosine similarity of the images generated by the two visualization methods. The idea of the cosine similarity is to expand the matrix into a one-dimensional vector and to calculate it by the following equation.

$$similarity = cos(\theta) = \frac{A \cdot B}{||A||||B||} \tag{6.1}$$

If the cosine similarity is closer to 1, the more similar the two matrices are. In this experiment, all the images generated by the two visualization methods were expanded in turn and the cosine similarity was calculated. As shown in Fig 6.2. :

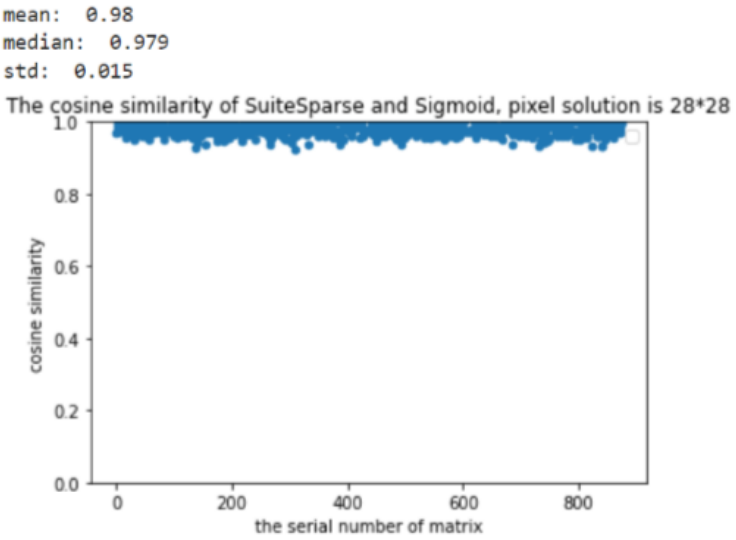

Figure 6.2: Similarity between two visualization methods

With such a high cosine similarity, the authors' view seems to be confirmed in some ways.

Therefore, in the following sections, I would discard the Sigmoid visualization method and use only the images generated by the SuiteSparse method for the experiments.

6.2 The comparison of different CNN structures

In this section, the prediction results of four different computer vision CNN structures, including Baseline's neural network, will be shown at four resolutions of matrix images. The validation method uses a 5-fold cross-validation, and the evaluation method uses accuracy, which is formulated as follows:

$$Accuracy = \frac{TN + TP}{TN + FN + FP + TP} \tag{6.2}$$

where TN means True Negative, TP means True Positive, FP means False Positive, FN means False Negative

6.3 Comparison of prediction accuracy between models

First I will give the results of the baseline experiment, as the Table 6.2.

Table 6.2: The accuracy of baseline

size	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
28×28	84.0	86.2	86.2	86.8	81.7	85.0
56×56	87.4	86.8	86.2	86.8	81.7	85.7
112×112	86.2	86.8	88.0	88.0	81.7	86.1
224×224	86.8	82.8	86.8	86.8	81.7	85.0

The table above showed the accuracy rate of the Baseline for a residual of 10^{-10}.

According to this table, I have guessed that the most significant difference is not caused by the resolution of the image, but there is a difference across groups and the variation of this difference is obvious and similar. So I conducted experiments using more computer vision models. And, I will give a comparison of the accuracy between the different models through 4-pixel solutions respectively in Table 6.3., 6.4., 6.5., 6.6.

Table 6.3: 5 groups of 4 different models under 28×28 pixel resolution

model name	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
baseline	84.0	86.2	86.2	86.8	81.7	85.0
ShuffleNetV2	82.3	78.9	79.4	84.0	76.0	80.1
SqueezeNet	77.7	82.9	74.9	80.6	80.6	79.3
ResNet50	84.6	85.1	86.9	88.6	78.3	84.7

Table 6.4: 5 groups of 4 different models under 56×56 pixel resolution

model name	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
baseline	87.4	86.8	86.2	86.8	81.7	85.7
ShuffleNetV2	81.7	77.7	79.4	82.3	74.3	79.1
SqueezeNet	73.7	84.6	72.6	81.7	81.1	78.7
ResNet50	85.7	85.1	85.1	87.1	79.2	84.4

Table 6.5: 5 groups of 4 different models under 112×112 pixel resolution

model name	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
baseline	86.2	86.8	88.0	88.0	81.7	86.1
ShuffleNetV2	79.4	77.1	76.7	86.3	76.7	79.2
SqueezeNet	75.4	82.9	76.0	80.0	77.7	78.4
ResNet50	88.6	84.0	84.6	90.9	79.4	85.5

Table 6.6: 5 groups of 4 different models under 224×224 pixel resolution

model name	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
baseline	86.8	82.8	86.8	86.8	81.7	85.0
ShuffleNetV2	82.3	81.1	79.4	83.4	72.6	79.8
SqueezeNet	74.3	80.0	76.0	81.7	82.3	78.9
ResNet50	91.4	81.7	83.4	83.4	80	84.0

In the above tables, I have used the same datasets to perform experiments on three vision models, ShuffleNetV2, SqueezeNet, and ResNet50. Further confirming my conjecture, the accuracy does not cause significant difference at the resolutions of $28 \times 28, 56 \times 56, 112 \times 112, 224 \times 224$, and all the computer vision models basically conform to this pattern. And, the variation in accuracy between groups is indeed similar across all computer vision models, i.e., if a model performs well or poorly on a particular group, it performs similarly at all resolutions.

One more notable thing: Although the accuracy of these models all looks to be in the neighborhood of about 80 to 85, the actual TP to TN ratio difference is very large. We will illustrate this in the table below. Since each resolution exhibits similar performance, the following experiments use only the case at a resolution of 28×28 as an illustration.

6.4 Performance on positive data prediction

As mentioned in the previous section, although all models have 80% 85% accuracy in prediction, the high accuracy for True Negative prediction can mask the low accuracy of True Positive prediction in the final accuracy calculation due to the imbalance between Negative and Positive data, which is also a problem in the previous research.

In this research, the introduced ResNet50 model possesses a better performance than the Baseline in terms of the proportion of True Positive. The following table shows the comparison of the 4 models in terms of True Positive prediction accuracy in the case of 28×28 images in the dataset.

I gave out the result in Table 6.7. as follows

Table 6.7: The number and ratio of TP in prediction, 28×28 pixel resolution, numbers in parentheses are ratios

model name	group1(%)	group2(%)	group3(%)	group4(%)	group5(%)	Average(%)
baseline	18(51.4)	22(62.8)	16(45.7)	25(71.4)	13(36.1)	94(53.4)
ShuffleNetV2	24(68.6)	17(48.6)	12(34.3)	19(54.3)	8(22.2)	80(45.5)
SqueezeNet	5(14.3)	6(17.1)	21(60.0)	23(65.7)	20(55.6)	75(42.6)
ResNet50	22(62.9)	20(57.1)	23(65.7)	26(74.2)	18(50.0)	109(61.9)
Total	35	35	35	35	36	176

As noted in the table above, ShuffleNetV2 and SqueezeNet outperformed Baseline's neural network in predictions for Positive data only in some groups. On average, the performance is lower than that of the Baseline.

In contrast, ResNet50 not only outperforms Baseline on average, but also shows much better prediction accuracy than Baseline in some groups where Baseline has very low prediction success (e.g., groups 1, 3, and 5). This is not only proof that ResNet50 may be a more suitable neural network than Baseline in this research, but it can also prove that trying new computer vision models is a feasible and valuable research method in the study of this problem.

Considering that the dataset of this research is images of sparse matrices, I think the study of the type and proportion of sparse matrices in these groups is also a future work worth exploring.

Chapter 7

Conclusion

This research imported 3 new convolutional neural networks to evaluate the methodology proposed by the previous research and showed the proposed methods for increasing the accuracy are not so working. Finally, it turns out that the deep neural network ResNet50 can improve the shortcoming of low accuracy when predicting positive(convergent) samples. Three new different CNNs structures, ShuffleNetV2, SqueezeNet, and ResNet50, were used in this research and validated by using the same datasets from the previous research. By comparison, it turns out that differences between groups were more obvious than differences between 4-pixel resolutions.

Secondly, the datasets produced with residuals of 10^{-10} which we cared about most, had a lower correct prediction rate in the classifier produced in the previous research, with an average of only 53.4%. Among them, the prediction rates were 51.4% for group 1, 45.7% for group 3, and only 36.1% for group 5, respectively.

And, the neural network of ResNet50 introduced in this study not only improved the average prediction accuracy from 53.4% to 61.9% but also improved the accuracy of 11.5%, 20% and 13.9% in group 1, group 3 and group 5, respectively.

It showed that in some ways, ResNet50 may be a more suitable neural network than the previous research's structure for this problem, and also proved that trying new computer vision networks or models is a feasible and valuable research method for increasing the accuracy in this problem. The other neural network structures introduced in this study, although not as good as Baseline's accuracy, also outperform the Baseline in some specific groups. What's more, it implies that the methodology of previous research can also work on other CNNs as well.

In the follow-up study, I will optimize the parameters of the neural networks to obtain better prediction accuracy. In addition, since different computer vision models have different prediction performances in different groups, I have a guess: this is essentially due to the different visual models' different abilities to perceive sparse matrices with different patterns. I will try to prove this guess. And if it's real, a scheme or algorithm that integrates the use of visual models to allow the most suitable neural network structure to make predictions on the most suitable images can be proposed, then it is possible to expect further improvements in the accuracy of classification and prediction.

Acknowledgements

I am grateful to my advisors Hasegawa and Yu for all the help they gave me during the study, both in terms of coaching on the progress of the experiments and in terms of moral encouragement, which has been of great help in this study.
I would also like to thank the author of the previous research, Ota, who left detailed experimental information and data, which provided an excellent basis for the development of this project.

References

[1] Owe Axelsson. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Linear algebra and its applications*, 29:1–16, 1980.

[2] Ryo Ota and Hidehiko Hasegawa. Predicting the convergence of bicg method from grayscale matrix images. *JSIAM Letters*, 12:45–48, 2020.

[3] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[4] Fouzia Altaf, Syed MS Islam, Naveed Akhtar, and Naeem Khalid Janjua. Going deep in medical image analysis: concepts, methods, challenges, and future directions. *IEEE Access*, 7:99540–99572, 2019.

[5] Michiel Kallenberg, Kersten Petersen, Mads Nielsen, Andrew Y Ng, Pengfei Diao, Christian Igel, Celine M Vachon, Katharina Holland, Rikke Rass Winkel, Nico Karssemeijer, et al. Unsupervised deep learning applied to breast density segmentation and mammographic risk scoring. *IEEE transactions on medical imaging*, 35(5):1322–1331, 2016.

[6] Hang Cui, Shoichi Hirasawa, Hiroyuki Takizawa, and Hiroaki Kobayashi. A code selection mechanism using deep learning. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, pages 385–392, 2016.

[7] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[8] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

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