Handwritten Math Symbol Recognization

Xueqian Zhang
xzhang2278@wisc.edu

Yuhan Meng
meng46@wisc.edu

Yuchen Zeng
yzeng58@wisc.edu

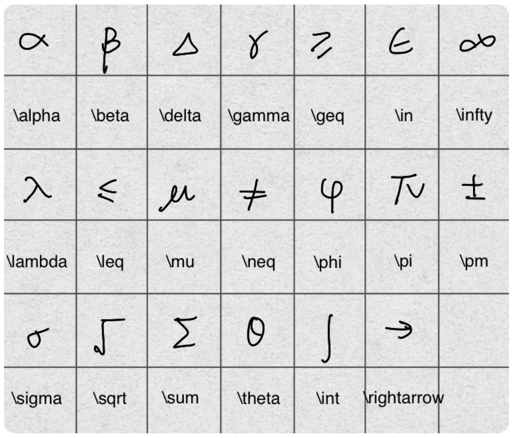


Figure 1. Example of handwritten math symbols

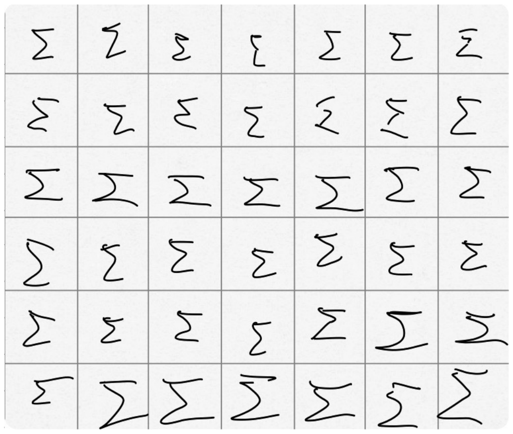


Figure 2. Example of handwritten Sigma

Abstract

Modern world has witnessed giant leap in the development of digital device. E-notes and printed copies are in increasingly more need, leading to the demand for translating hand-written math symbols to latex format for editing and printing convenience. In order to output the latex command automatically so that it can constitute the stepping stone to outputting math formulas, we collect more than 150, 000 images and train various neuron network models based on this large dataset.

To begin with, fundamental data processing such as purifying the images' file names and extracting a subset of images as a sample data source are implemented. This will reduce great time complexity when initially constructing networks. The main models are built based on some well-known models like Alexnet, Resnet34, NiN and VGG16. During the training process, we also tried to introduce some modifications like employing BatchNorm and dropout methods. After comparing the validation accuracies of all the networks, it's obvious to identify the best model is neuron network with 3 convolution and 3 Fully-Connected layers, including BatchNorm. Finally, we do further inference with this perfect model. For instance, we output the top 3 highest probability math symbols prediction of a certain image. Besides, creating new math symbols ourselves, we tested the model's usibility on these future data.

1. Introduction

The groundbreaking attempt to automatically recognize handwritten mathematical expressions dates back to the 1960s. In the past decades, this research has received extensive attention in the research community.

At present, due to the rapid development of technology in recent years, the usage of paper science literature has gradually decreased. Therefore, the use of digital files in the scientific community has greatly increased [1]. Among these documents are scientific literature, more specifically mathematical literature. So a tool is needed for studying the recognition of handwritten math languages in a variety of applications[1]. Every year, relevant conferences publish many related papers. More than 200 papers have been published on different journals and conference proceedings on this particular issue. However, there is still much room for improvement in this area[1]. Besides, for we students, there is an increasing trend and requirement for us to recognize different handwritten mathematical symbols and then translate them into other applicable language such as latex language.

[1]https://www. isical. ac. in/ crohme/index.html

For instance, currently most students need to hand in their assignment in a print version. To reduce the complexity of equation editing with word or latex editors, students might write it on the paper or on the ipad, e-notebook by hand first. Then, with the method of recognizing various handwritten symbols, it could be the pre-steps of classifying and translating different handwritten verison symbols into electronic verison. Moreover, if there is an approach to distinguish different mathematical languages, it will somewhat help math teachers get the right answer whenever they meet those hard-recognized symbols in math exam papers.

1.1. Motivation

For the purpose of transforming the handwritten notes into electronic format, our project can be a good start. And the reasons are listed as follows. First of all, our dataset is big enough to be used for classifying different handwritten math symbols. In reality, if we want to type those math symbols into latex, it would be very inconvenient. In this case, our project would work on those handwritten math symbol images and recognize them automatically. Additionally, our program would output the latex command in the same time, too. Meanwhile, if we do not know how to type the math symbols, it would be a good way for us to write the symbol first, recognize it by our model and get the corresponding latex command. Secondly, nowadays, more and more students are likely to use electronic devices such as ipads to take notes in classes. In this case, if there is a program can recognize the symbols directly, it would be very easy to transform our handwritten notes into neat latex form. As an application, online identification provides a scientific documentation reference for the users to input math formula. But it would be further application, our purpose at present is just recognizing the math symbols correctly and automatically. Therefore, our project can be very helpful to the application on transforming handwritten notes into electronic format in the future. We believe that it would be pretty meaningful for all kinds of people especially faculty. Overall, success in this area can basically improve the state of the art in visual language understanding.

1.2. Evaluation

Our purpose is to recognize the symbols correctly (accuracy roughly 90%) and output the probability for a handwritten math symbol belongs to different clusters. That is to say, we would first label the symbol with the class which has the highest probability and then compare it with the true test set label to check the accuracy. Additionally, we would give the corresponding latex command for each handwritten math symbol for others to use, which is a little application of our model. Moreover, we would employ the three-way hold-out method. To be more specific, as for our project, we would split our dataset into training, validation and test set. Using validation set to tune for the best hyperparameter and model algorithm in order to optimize the network since the test set could only be used once. Finally, we might verify our model by using test set, if all the training set, validation and test set have the accuracy higher than 90%, then we reach our goal. One thing we should be careful is that we need to avoid over-fitting. Over-fitting the training set so well would lead to the poor performance for the test set. Thus the level of over-fitting is one of the most significant indicator to evaluate the model.

As for evaluating the models, we choose the basic error and accuracy criterion though there are many other principles such as constructing ROC AUC curve, entropy, the accuracy is enough for us.

2. Related Work

An important question in Machine Learning area is how to automatically learn from good features. In this section, we will introduce the previous research on CNN and the recent research on handwritten math symbols recognition based on neuron network.

"Convolutional Networks are trainable multistage architectures composed of multiple stages. "[2] The idea of Convolutional Neuron Networks is based on the cats primary visual cortex which contributed by Hubel and Wiesel in 1962. In 1982, K. Fukushima and S. Miyake first simulated such model on a computer[3]. Y. LeCun innovatively used the back-propagation algorithm to simplify the architecture and applied it to handwritten zip code recognition. Convolutional Networks also applied to hand gesture[4] , text[5] and visions[2]. Convolutional Neuron Networks can make progress on the classification for whole-image and also improve the local tasks with structured output. One of the outstanding advantages of CNN during detecting process is that it can compute efficiently.

Convolution Neural Networks (CNNs) is a powerful technique for classification of handwritten visual inputs. In the last few years, plenty of work have contributed to recognizing handwritten digits and English words [6][7]. MNIST dataset is widely used try on new learning algorithm. [8]

In the field of handwriting recognition, there are two common approaches, one is online and another is offline recognition. [9]. As for online recognition, people write on a digital medium such as a designated application on a tablet. In offline recognition, people often write on a non-digital medium like a piece of paper. [10]Converting typed PDF files into LaTeX representation has already been done by Chang in 2016. [11] An application called Mathpix Snipping Tool would help us transfer the equations images into LaTeX form after screenshotting them. Another online application called Visual Objects Web Equation allows us to write expressions on a pad in their browser and then it will convert the handwritten equations into LaTeX.

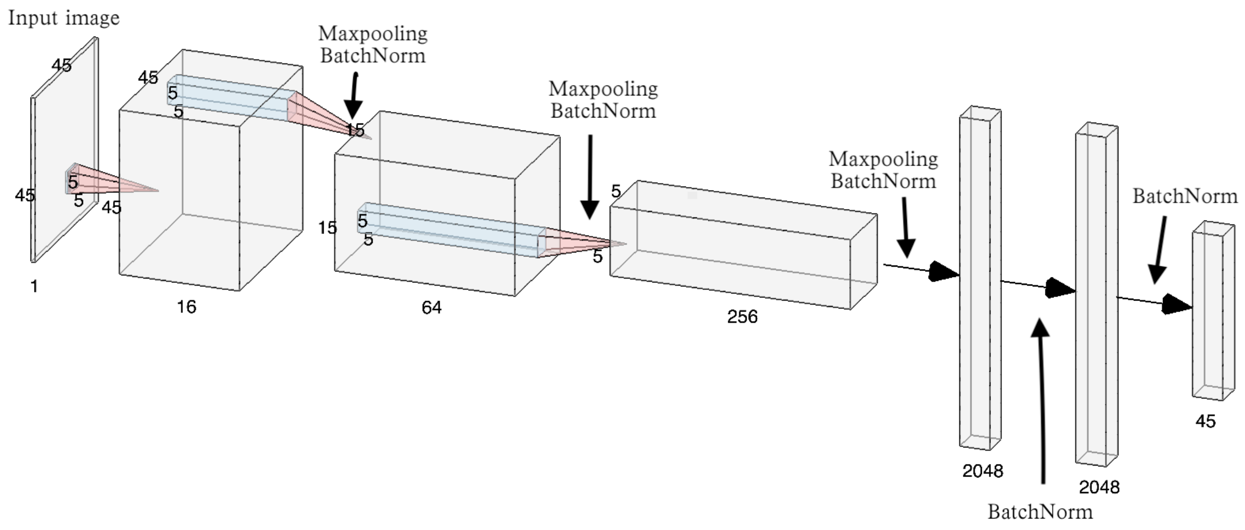


Figure 3. The algorithm graph of the selected model (Model 12).

Based on the researching above, we would not only classify the math operators based on algorithm such as CNN, Alexnet, VGG and Resnet but also make some application. We would like to allow people write math symbols on the ipad or other digital medium and then recognize them and output the LaTeX form.

3. Proposed Method

We tried many method, and some of them are based on some famous models like AlexNet, ResNet, so on and so forth. We would introduce the models we based on, and the changes we did on those models in this section.

3.1. AlexNet

To train the model based on image datasets, there are already oceans of distinguished algorithms and models that have come into being and are implemented by many scholars. Standing on the shoulders of the giants, we also constructed our models based on these existent outstanding models.

According to the logical order from simplicity to complexity, we begin with some basic networks. Since as it known to us all that the multi-layer perceptron model will not perform well due to the high time consuming and low accuracy to image datasets, it should not come as a surprise that we start with combining the Fully-Connected layers and CNN layers, which is the essence of Alexnet model.

AlexNet is a combination of five convolution layers, intervened with max-pooling layers, as well as three Fully-Connected layers at last. The activation function it used is non-saturating ReLU rather than tanh and sigmoid, which gave higher training performance. [12] Because, standardizing the data will always give us an unexpected results, there is a pragmatic method, 'BatchNorm', in deep learning to transform the data so that the inputs are with zero mean and unit variance. The computing algorithm[2] is:
Input: Values of x over a mini-batch: $\boldsymbol{B} = \{x_1, ...x_m\}$.

$$\mu_{\boldsymbol{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$
$$\sigma_{\boldsymbol{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\boldsymbol{B}})^2$$
$$\hat{x_i} = \frac{x_i - \mu\boldsymbol{B}}{\sqrt{\sigma\boldsymbol{B}^2 + \epsilon}}$$
$$y_i = \gamma\hat{x_i} + \beta = BN_{\gamma,\beta}(x_i)$$

Hence, we introduced BatchNorm method to probe whether or not it will improve the performance of our model. Besides, to reduce the over-fitting problem, we also applied the drop out approach. Moreover, to learn about the effect of the usage of max-pooling layers, their existence could also be another component to control. Finally, we constructed some models based on the combination of these components. Figure 3 is an example of these models.

[2]https://kratzert. github. io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html

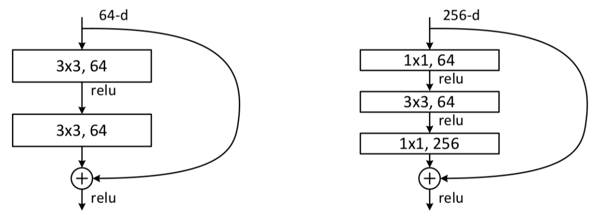


Figure 4. The building Block of Resnet. (Resource: Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep Residual Learning for Image Recognition.)

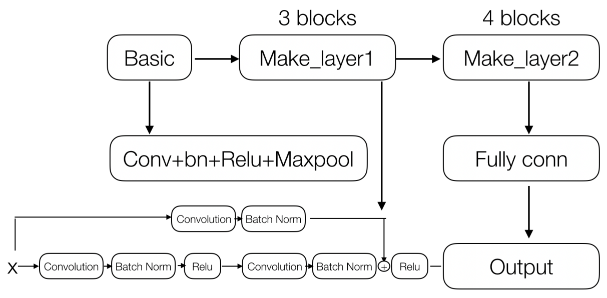


Figure 5. The flow chart of the model (Resnet) we fit based on Resnet.

After employing the fundamental models, what we considered next are some more complicated models.

3.2. ResNet

The words 'resnet34', 'resnet50', 'resnet101' and 'resnet152' might not be unfamiliar to you. Resnet is a model which can skip the connections and learn the residual: $\boldsymbol{R(x)} := \boldsymbol{H(x)} - x$ between output and the previous input layers as well as the identity functions. The original function becomes $\boldsymbol{R(x)} + x$. [13]The very important component in the Resnet is the building block, which can be seen as the gated recurrent units that bear great similarity to elements in RNNs, as shown in Figure 4.

Thanks to this technique, we could construct models based on Resnet, although our model might have lower complexity than that of the Resnet34.

The structure of our model is shown in Figure 5. Simpler than Resnet34, our Resnet model only includes 2 great layers. The first contains 3 building blocks and the other contains 4 blocks. After implementing the basic layers with convolution, BathNorm, Relu and maxpooling, the model will go across 7 blocks and then enter the Fully-Connected layer and finally the output layer.

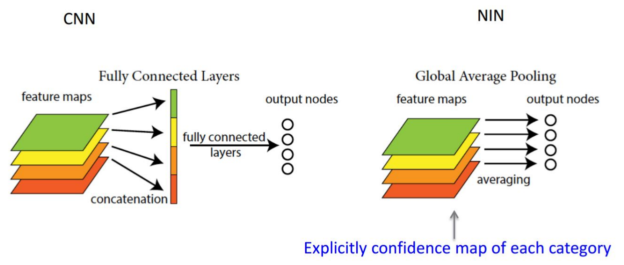


Figure 6. The comparision of CNN (left) and NiN (right). (Resource: http://vsooda. github. io/2017/03/09/global-pooling/.)

3.3. NiN

"The conventional convolutional layer uses linear filters followed by a nonlinear activation function to scan the input. The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN; they are then fed into the next layer. " [14]. In traditional CNN architectures, the feature maps of the last convolution layer are vectorized and the model will go across Fully-Connected layers and then obtain the probability of each class after activation function such as sigmoid and softmax. However, this approach might be abstract since we cannot get visualizion at the last Fully-Connected layers. Meanwhile, the large amount parameters will increase the tendency to over-fitting. To prevent over-fitting, NiN appied by using global average pooling instead of feature maps in the classification layer[3]. The comparison between CNN and NiN model is shown in Figure 6.

During the implemention of our NiN, we modified the NiN model proposed by Lin Min, Qiang Chen in "arXiv", via using 3 convolutional layers with maxpooling method instead of micro-network, whereas the last 2 convolutional layers with the global average pooling method for better generalization and less over-fitting.

3.4. VGG

Similar to Resnet, VGG has come up with increasingly more versions these days, such as VGG-11, VGG-11 (LRN), VGG-13, VGG-16 (Conv1), VGG-16 and VGG-19, shown in Figure 7. The reason why scholars focus on this model is that many other models are constructed according to the basic structure of VGGNet (Figure 7).

In this article we are extremely focused on VGG16, which is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". Because extracting features are general, we pre-trained the model and then modified the last two layers based on VGG16.

[3]http://teleported. in/posts/network-in-network/

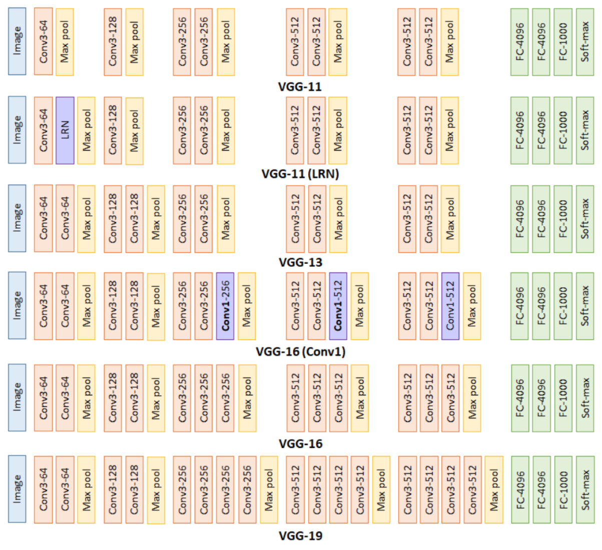


Image Conv3-64 Max pool Conv3-128 Max pool Conv3-256 Conv3-256 Max pool Conv3-512 Conv3-512 Max pool Conv3-512 Conv3-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-11
Image Conv3-64 LRN Max pool Conv3-128 Max pool Conv3-256 Conv3-256 Max pool Conv3-512 Conv3-512 Max pool Conv3-512 Conv3-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-11 (LRN)
Image Conv3-64 Conv3-64 Max pool Conv3-128 Conv3-128 Max pool Conv3-256 Conv3-256 Max pool Conv3-512 Conv3-512 Max pool Conv3-512 Conv3-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-13
Image Conv3-64 Conv3-64 Max pool Conv3-128 Conv3-128 Max pool Conv3-256 Conv3-256 Conv1-256 Max pool Conv3-512 Conv3-512 Conv1-512 Max pool Conv3-512 Conv3-512 Conv1-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-16 (Conv1)
Image Conv3-64 Conv3-64 Max pool Conv3-128 Conv3-128 Max pool Conv3-256 Conv3-256 Conv3-256 Max pool Conv3-512 Conv3-512 Conv3-512 Max pool Conv3-512 Conv3-512 Conv3-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-16
Image Conv3-64 Conv3-64 Max pool Conv3-128 Conv3-128 Max pool Conv3-256 Conv3-256 Conv3-256 Conv3-256 Max pool Conv3-512 Conv3-512 Conv3-512 Conv3-512 Max pool Conv3-512 Conv3-512 Conv3-512 Conv3-512 Max pool FC-4096 FC-4096 FC-1000 Soft-max
VGG-19

Figure 7. Different structure of several different verisons of VGG models. (Resource: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11)

The input to conv1 layer is of size 224 x 224 RGB image. [15] So in order to pretrained our model with VGG16, we need to first transformed our image into RGB mode. VGG16 is a collection of convolutional layers, where most of the filters are only in a small size: 3x3, so that they can capture the notion of all the sides. After the stack of convolutional layers, the last three are Fully-Connected layers with the same configuration. The first two 4096 channels and the last one 1000 channels for 1000 classes. [15]

After tranforming our image dataset into RGB mode and pretraining the first part of our model except the last layer using VGG16, we changed the last output layer into 2 Fully-Connected layers. The structure of our model is shown in Figure8.

3.5. Model Flow

We constructed 8 networks based on all that models mentioned above. The structure of each (except the Resnet) is shown as Table 1. The structure of Resnet is more complicated than those seven, so we would introduce it later.

Now we focus on the seven models first. The first model, based on AlexNet, includes 3 subnetworks with three components, BatchNorm, maxpooling and drop-out to be existent or not. Obviously, it seems no structure difference among model 11, model 12 (Figure 3) and model 13. (Table 2 on the last page). To compare these three models, we can know how much improvement we did by adding batch normalization and dropout. As for model 2 and model 3, they are similar with the first three(three convolutional layers + three Fully-Connected layers), but have different value of parameters like kernel and channel sizes. When it comes to NiN, VGG16 and Resnet, actually they are also modified versions as introduced before.

4. Experiments

4.1. Software

1.Python: training model, predicting.
2. R: making graphs.
3. NN SVG: making the CNN structure graphs.

4.2. Hardware

Laptops.

4.3. Dataset

The dataset we chose is from kaggle database webset[4]. The data are chosen from this website because there contains large amount of well-known data mainly for machine learning and deep learning. Our dataset comes from the competitions CROHME 2011, 2012 and 2013 called Handwritten math symbols dataset [5]. It consists of jpg files of plenty of characters. The size of our images is 45×45. Since detecting the letters are not our main purpose, we delete the images of letters. After removing these images, we got our whole dataset and the total number of remaining images is 153, 355. The math symbols are:

1.Basic Greek alphabet symbols $\alpha, \beta, \gamma, \mu, \sigma, \phi, \theta$;
2. All math operators, set operators:$+$, $-$, $\times$, $=$, $()$, $[]$, $\{\}$;
3. Basic pre-defined math functions like: log, lim, cos, sin, tan;
4. Math symbols like:δ, $\exists$, $\sum$;
5. So on and so forth.

The original data set we download is really mass hence we renamed our images with class and order number. (eg. alpha_11.jpg means this image is α and it is the 11^{th} image of alpha category in the folder). What's more, we created a sample image folder which includes part of the images in the whole dataset. We selected them randomly and also according to the proportion of each symbol group. In practice, we would like to use the sample dataset first to reduce time complexity. Once the model performs well on the sample dataset we will apply it to the whole dataset.

Before training the model, we used three-way hold-out method to split our dataset into three parts: training, validation and test set. We also apply it to the sample dataset.

Excluding the whole dataset we got online, we create some images by ourselves and friends which would be used to check the performance of our model on future dataset.

[4]https://www. kaggle.com/datasets website[2]
[5]https://www. kaggle.com/xainano/handwrittenmathsymbols

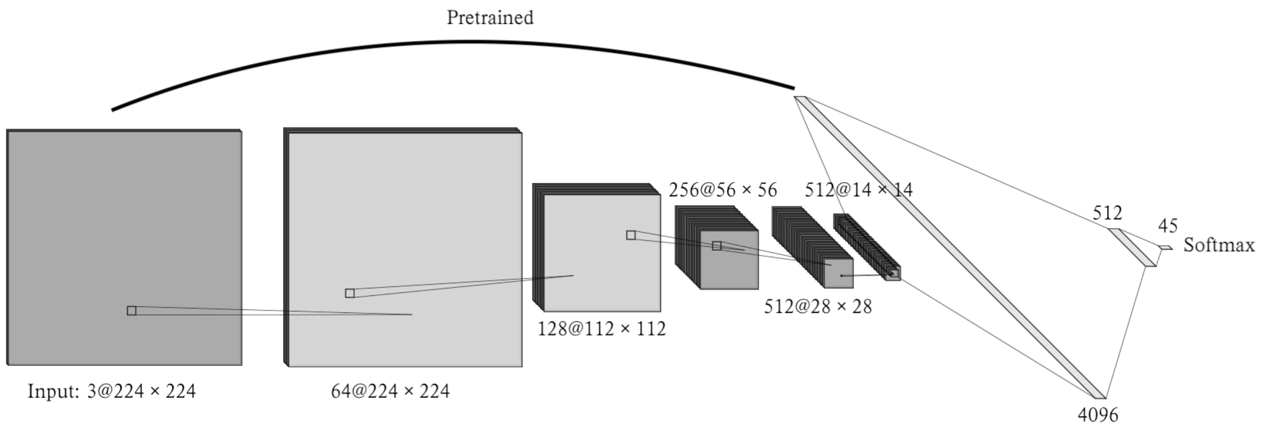


Figure 8. Our model which based on VGG16. The layers except the last two layers are pre-trained and the last two Fully-Connected layers are changed.

ConvNet Configuration						
Model 11	Model 12	Model 13	Model 2	Model 3	NiN	VGG16
Input(45×45 Gray Scale Images)						
conv5-16			conv3-32	conv4-64	conv3-32	conv3-64, conv3-64
maxpool(5, 3, 1)		-	-	-	maxpool(3, 1, 1)	maxpool(2, 2, 0)
conv5-64			conv3-256	conv4-128	conv4-32	conv3-128, conv3-128
maxpool(5, 3, 1)		-	-	maxpool(2, 2, 0)	maxpool(3, 2, 1)	maxpool(2, 2, 0)
conv5-256			conv4-256	conv4-256	conv4-256	conv3-256, conv3-256, conv3-256
maxpool(5, 5, 0)		-	-	maxpool(2, 2, 0)	maxpool(3, 2, 1)	maxpool(2, 2, 0)
FC-2048			FC-4086	FC-2048	conv2-45	conv3-512, conv3-512, conv3-512
FC-2048			FC-4086	FC-2048	avgpool(3, 2, 1)	maxpool(2, 2, 0)
FC-45					-	conv3-512, conv3-512, conv3-512
-						maxpool(2, 2, 0)
-						FC-4096
-						FC-512
-						FC-45
softmax						

Table 1. ConvNet Configuration of our models except Resnet whose structure is more complex. The a, b, c in "maxpool(a, b, c)" indicate the kernel size, stride and padding. The k, r in "convk-r" indicate the kernel size and the number of ourput channel respectively.

4.4. Data Processing

The raw data (153,355 gray scale jpeg images) we downloaded from kaggle have already been classified into different folders. First, we renamed the images to let the names match their classes. Meanwhile we extracted them into the same folder. Second, we used python to write a csv file contained all the name of images and their labels. Third, we took one-tenth of the full data to be the sample data. Additionally, for both full data and sample data, we took the training data set, validation data set and test data set according to the ratio of 7:2:1.We wrote every data set such as the training data set in sample data into csv files for us to load later in python. To speed up and simplify the process of model training, we used sample data set to train our models first, and then used full data if the model works well on sample data. This approach saved us a lot of time because it helped us avoid wasting too much time on inappropriate models.

4.5. Model Selection

After data processing, we constructed the networks based on the proposed methods which we listed in the section 3. The accuracy of all the models are graphed in Figure 9. As the figure shows, the model 13 shows the highest training accuracy and the model 12 shows the highest validation accuracy. The lowest accuracy appears in VGG16 whose accuracy is under 90%, much lower than other seven.

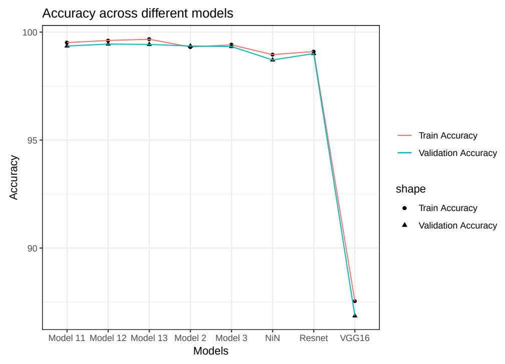


Figure 9. Accuracy across different models.

However, the time it take us to fit the model is the longest even if we downloaded the pre-trained model. To be more specific, it almost took us 10 times as the time other models took. In this case, VGG16 is the worst model no matter in which aspect. Finally, we select the second one of the first model due to higher validation accuracy and lower over-fitting as the final model we would use to do the prediction.

4.6. Prediction

After selecting the model which performs best among 8 models, since the test accuracy is high enough for us to apply it into reality, we use it to do the prediction. The prediction is the most meaningful part in our project. Actually, it is the final goal of doing all these troublesome data processing, model selection and so on.

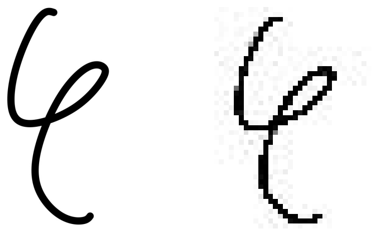
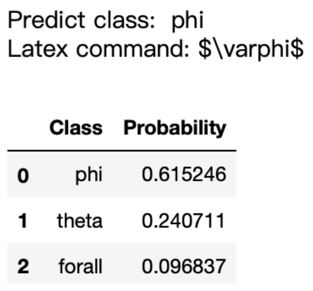


Figure 10. Transcode the raw image (left) into the formatted image (right).

Now we can write any math symbols included in our dataset, and save it as an image. However, the format of the new image is totally different with the images in our dataset. Thus we need do some transcoding on the input images. After the input RGB big image has been transformed into Gray 45×45 jpeg image which fit the format of images in our dataset (Figure 10), we use our model to do the classification. To make the output informative, we not only output the predict class, but also output the top 3 classes which have the highest probability. To make it more useful, we can output the latex command together with the predict



```
Predict class:  phi
Latex command: $\varphi$
```

	Class	Probability
0	phi	0.615246
1	theta	0.240711
2	forall	0.096837

Figure 11. Example prediction output table format.

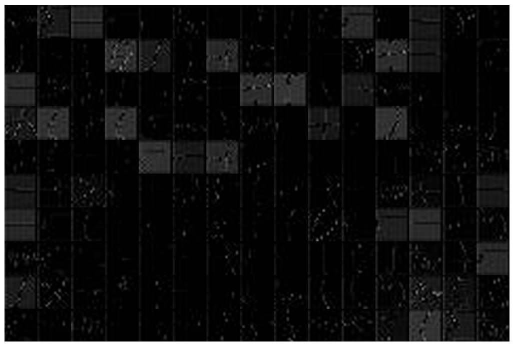


Figure 12. Some feature maps from the first convoluntional layer.

class, which would be of great use in education. The format of our output table is shown as Figure 11.

5. Results and Discussion

In conclusion, we have performed eight different models, some of them are CNN whose hyper-parameters are adjusted by ourselves and some of them are constructed based on some well-known models such as Resnet34, NiN and VGG16. The results are shown in Table 2. Thanks to the handwritten math symbols all are written on whiteboard, we do not have to distinguish different symbols in different background. Therefore, seven out of eight models achieve extremely high accuracy without serious over-fitting. As for the best model which shows the highest validation accuracy, the test accuracy of it is 99. 55%. Thus it is reasonable for us to conclude that the best model we selected is of good fit. From the table 2 we can also analyze the effect of BatchNorm and dropout have on our models. The visualization of the first layer of our model is shown as Figure 12. Because the original images are gray scale images, the feature map is pretty dark.

In our test on predicting the classes of handwritten math symbols by model 12, our model performs not bad as long as we can write the math symbols neatly without big confusion. However, if we try some confusing handwriting on purpose, our model sometimes may give some ridiculous results. We three have performed some tests on our final

model	BatchNorm	Dropout	Training Acuuracy	Validation Accuracy
Model 11	False	False	99. 514%	99. 356%
Model 12	True	False	99. 613%	**99.446**%
Model 13	True	True	**99.669**%	99. 43%
Model 2	True	True	99. 311%	99. 356%
Model 3	True	False	99. 41%	99. 33%
Resnet	True	False	99. 101%	99. 006%
NiN	False	False	98. 958%	98. 71%
VGG16	False	False	87. 535%	86. 86%

Table 2. The comparision of training accuracy and validation accuracy of different models.

model. We wrote 30 math symbols and tried our best to make it looks confusing. The result is six out of thirty are labelled wrongly. Although sometimes it cannot recognize some indistinguishable handwriting, we still think it is a applicable model. This is because in most of the cases, mistake we made in doing prediction is pretty reasonable (the handwriting is indeed very confusing).

6. Conclusions and Limitations

6.1. Conclusions

- Regularization techniques such as Dropout and Batch Normalization lead performance and efficiency improvements. From model 11 to model 12 and model 13, we add regularization step by step and it is clear shown that the training accuracy and validation accuracy indeed increase and all of them give an excellent performance. Since our models don't have over-fitting, adding dropout method does not lead to a better accuracy instead it kinds of reduce the capacity of our models.

- CNN all perform greatly while for complex models like Resnet, NiN and VGG 16, the accuracy of them are not as good as CNN models. The reason that lead to this phenomenon might be that the background of our images is white which reduce the noise. What's more the complex models also cost us a lot. Hence simple structure is more suitable for our dataset.

- The performance of our best model on future dataset is quite good. When we apply our model on the future dataset, six of them produce misclassification. Since the purpose of our application is to label indistinguishable symbols we all created confusing handwritten math symbols. If we apply our network on regular math symbol images the accuracy can be much higher than 80% .

6.2. Limitations

- Our model cannot recognize a complete math formula. As we described before, our model can only recognize a single symbol. If we want to get stronger application like transforming handwritten math formulas into latex command directly, we still have a long way to go: CNN is not enough, we may need RNN to process the math formulas because there may exist some subscript related to the former input. Actually we may need more tools out of classes, too.

- It is not real-time. After we write the math symbols, we cannot get the predicted class instantly but have to crop it and save it into our computer. We believe we can solve this problem if there is longer time for us to spend on our project.

- We only use accuracy to judge the performance and select the models. Actually there are many other auxiliary approaches such as graphing ROC AUC curve and confusion matrix, constructing statistical confidence intervals and etc. Because computing the accuracy is the most fundamental criterion to select appropriate models, and the confusion matrix is so large for our model(45 classes), we only use this evaluation method.

7. Acknowledgements

This project is encouraged by Sebastian Raschka at the University of Wisconsin-Madison, Madison, Wisconsin. Thanks the Stat 479 staff for putting together such an incredible class. We all learned a lot.

8. Contributions

Each of us three made a great contribution to the project and all of us take part in all the parts. To be more specific, as for the code, first, we decided the main idea together; then, Yuchen Zeng and Xueqian Zhang did the data processing while Yuhan Meng checking the result; next, we fit the model together and each team member tried about five models; finally, Yuchen Zeng did the prediction part. As for the presentation, we worked together: Xueqian Zhang and Yuhan Meng made most of the slides. As for the report, we work together again.

Overall, it is hard to say who did a certain part. We are working together and helping with each other all the time.

References

[1] A. . S. Fitriawan, H., "Neural networks for lampung characters handwritten recognition," *Computer and Communication Engineering (ICCCE), 485-488.*, 2016.

[2] Y. Lecun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*, 2010.

[3] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition, vol. 15, no. 6, pp. 455469*, 1982.

[4] S. Nowlan and J. Platt, "A convolutional neural network hand tracker," *San Mateo, CA: Morgan Kaufmann, pp. 901908*, 1995.

[5] M. Delakis and C. Garcia, "Text detection with convolutional neural networks," *International Conference on Computer Vision Theory and Applications*, 2008.

[6] S. S. D.-S. Lee, "Handprinted digit recognition: a comparison of algorithms.," *Proc. 3rd Int. Workshop on Frontiers of Handwriting Recognition, pp. 153164.*, 1993.

[7] R. Plamondon and S. Srihari., "On-line and off-line handwriting recognition: A comprehensive survey.," *IEEE Transactions on Pattern Analysis and machine intelligence, 22(1).*, 2000.

[8] T. Bluche, "Mathematical formula recognition using machine learning techniques.," *APA*, 2010.

[9] C. Lu and K. Mohan., "Recognition of online handwritten mathematical expressions using convolutional neural networks.," 2015.

[10] R. Plamondon and S. N. Srihari., "Online and off-line handwriting recognition: a comprehensive survey.," *IEEE Transactions on pattern analysis and machine intelligence 22.1 (2000): 63-84*, 2000.

[11] S. G. Chang, Joseph and A. Zhang., "Painfree latex with optical character recognition and machine learning," 2016.

[12] S. I. S. H. G. E. Krizhevsky, Alex, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM. 60 (6): 8490.*, 2016.

[13] K. H. X. Z. S. R. J. Sun, "Deep residual learning for image recognition," *Microsoft Research,770-778*, 2015.

[14] S. Y. Min Lin, Qiang Chen, "Network in network," *arXiv:1312.4400*, 2014.

[15] K. H. J. S. Xiangyu Zhang, Jianhua Zou, "Accelerating very deep convolutional networks for classification and detection," *arXiv:1505.06798*, 2015.