**Project report: classify Russian letters by CNN**

* Junyu Mou
* CS 767
* April 27th, 2022

Research scenario and research question

The dataset are pictures of handwrite Russian letters. The dataset includes 14190 PNG images with a csv file as the description letter. The images have 3 channels as RGB with different colors, different person’s handwritings, and various background. The description file is like a catalog including the Russian letter, the label for the letter, file name and background. In this project I build a CNN model to classify the pictures and predict the handwriting images by the model. There are three essential goals for this project:

1. Build the CNN model to classify the pictures and predict the image’s Russian character
2. Optimize the model to achieve better training and testing accuracy and avoid overfitting
3. Research on the problems met during the training

Basic Requirements

There are a few presupposed conditions for success, the conditions mean the bottom requirement for the project

1. The model will correctly train with 32 x 32 x 3 pictures
2. The accuracy should be at least 60% for both training and testing
3. At least 2 set of designs should be tried

Desired outcome

I will try different designs to find the best model, and there are some targeting performance for the project. It is not mean that the target will be achieved for sure, but the desire outcomes are the final goal for the project.

1. The accuracy for training should be at least 90% with at 70% in testing
2. The difference between training and testing should be less than 10% to prevent overfitting

Describe the data

The data are PNG pictures of handwriting of 33 Russian letters. The dataset includes 14190 PNG format images. There is a .csv file containing information for each picture. The letter in Cyrillic, the label for each letter, name for files and the background which is the label background of the image.

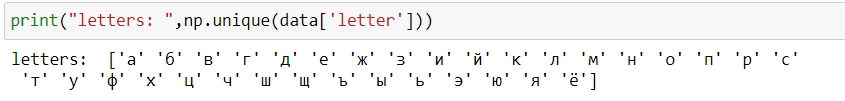
A screenshot of a computer

Description automatically generated with low confidence

Table

Description automatically generated

The letters are:



Data visualization and plots

Let’s look at dataset with the label:

A picture containing text, keyboard

Description automatically generated

I plot some of the letter with more than one samples. In this visualization there are 12 out of 33 letters. Each letter with four examples. For each letter, the handwritings are in different colors, different handwriting style and different background.

Background pattern

Description automatically generated

This is the histogram for data of each label. There are 33 labels representing 33 letters. The graph shows that the dataset is perfectly balanced. In this case no further sampling(over sampling or under sampling) is needed.

Graphical user interface, text, application

Description automatically generated

The distribution of background is not perfectly uniform.

Graphical user interface, application

Description automatically generated with medium confidence

The four examples are the same letter shows in different backgrounds.

Preprocessing the data

1. Open the folder of dataset:



1. Import dataset

A picture containing text

Description automatically generated

In this step, the image name in csv file is used to open each image. For open image the main function here is keras.preprocessing.image.load\_img(). The main function took the image file path and file name to open the image, then the function img\_to\_array() converts image to array. I save all arrays in a list, and finally convert the list to NumPy Array. The label came from the csv file convert into a list.

1. Min-Max normalization and label encode

Text

Description automatically generated with low confidence

Min-Max normalization is one of the most common normalization methods. Even though it doesn’t handle outliers effectively, it is suitable for pictures.[1] When applying backpropagation to optimize the model, unnormalized data could lead to exploding/vanishing gradient problems. [2] In this project I convert class list to binary class matrix. Because the label is not ordinal and not so many categorical features, I will use one-hot encoding here.[3]

1. The dimension for the preprocessed dataset

Graphical user interface, text

Description automatically generated

Train the CNN model

A picture containing text

Description automatically generated

For the initial model, I used a data augmentation layer to randomly flip, randomly rotate and random zoom the images. The model is a sequential model. With Con2D layers, Batch Normalization layers for faster coverage accompanied with ReLU activation functions and Dropout layer as regularization to prevent overfitting. Maxpooling layers are used to prevent overfitting and reduce computational complexity.[4] Then there are two fully connected Dense layers with Batch normalization, ReLU and Dropout. The output layer has 33 neurons for 33 classes with SoftMax activation function. The Adam gradient decent algorithm is used with power scheduling.

Graphical user interface, text, application

Description automatically generated

Chart, line chart

Description automatically generated

The result of the initial model is 62.3% accuracy for training accuracy and 65.5% for testing accuracy. No sign of overfitting. Basically, basic requirements are met: the model can be trained by 32 x 32 x 3 images with at least 60% accuracy. Further, I will try different parameters to optimize the model.

Different designs and results

|  |  |  |  |
| --- | --- | --- | --- |
| Set up | Train accuracy | Test accuracy | Description |
| No batch normalization and drop out | 3.3% | 2.5% | The training is fast, only take 7 steps before stop with EarlyStopping patient = 5. The model failed. |
| No image augmentation | 97.9% | 92.8% | The training is also fast with 3s for each step. It takes 23 steps before stopping. The accuracy for both training and testing are high without sign of overfitting. |
| Sigmoid, SGD | 3.2% | 3.1% | It takes 48 steps before stop. The model failed. |
| Sigmoid, Adam | 95.5% | 79.2% | 38 steps before stop, while the val\_loss is going apart from loss, the model is overfitted. |
| ELU,Adam | 99.0% | 92.4% | 25 steps before stop, no significant sign of overfitting. |
| ReLU,Adam,power scheduling,Glorot Initialization | 97.9% | 90.4% | 17 steps before stop, no sing of overfitting |
| ReLU,Adam,power scheduling,He  Initialization | 97.0% | 91.9% | 17 steps before stop, no sign of overfitting |
| ReLU,Adam,exponential scheduling, He  Initialization | 99.6% | 95.7% | 57 epochs, no sign of overfitting |
| ReLU,Adam,performance scheduling, He  Initialization | 99.9% | 95.5% | 77 epochs, no sign of overfitting |
| ReLU,Adam,exponential scheduling | 99.5% | 94.9% | 36 steps before stop, no sign of over fitting |

Final model

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

A picture containing application

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Discussion about the training

Batch normalization & Dropout

When the batch normalization and Dropout was removed, the model appeared underfitting or overfitting depends on the other parameters. This product questions about when to use batch normalization and Dropout. Batch normalization is the technique used for training deep neural networks especially for very deep ones, it can stabilize the learning process and significantly increase the learning speed by reducing training epochs required. [5] Batch normalization is essential for CNN, because it can make the training in deep model without compromising with training speed. [6] Batch Normalization could also let the model to not to rely on weights initialization so much and restrain overfitting. Without Batch Normalization the model could fail without weights initialization. Dropout is used to prevent overfitting by randomly disabled neurons and their connections. By further training, the Batch normalization cause almost all models to coverage faster and removing Drop out will significantly increase the appearance of overfitting. The effect of removing Batch normalization and Dropout is not the same for all designs. Sometimes the influence of removing them is significant and sometimes it is not.

Why image augmentation decrease accuracy?

When I add an image augmentation layer, several models were worse in training and testing accuracy compared to model without it. Because the data augmentation is used to expand the dataset, I was wondering why the model was worse with it. There are several possible reasons founded.

1. Data augmentation can be a regularization method, it will relieve the overfitting problem and reducing the training accuracy. For the project, one of the models may be because of this reason. That model got 75% and 72% in training and testing accuracy.
2. Some functions of image augmentation, such as randomly flip or randomly zoom, could produce invalid image in the domain. In other words, the image augmentation added data which is not the same pattern as the original dataset.
3. The original dataset is too small, added images from data augmentation attenuate the pattern in the original dataset.

Sign of overfitting in CNN

Overfitting or high variance appears when the training accuracy is greater than testing accuracy. Overfitting indicates the filters in Conv2D layers are too large. This is as same as too many features in regression models or too any layers in DNN. The other reasons for overfitting are: the dataset is too small, the distribution of training and testing set is not identical, or the quality of data is low such as too much noise. To resolve the overfit, we could use regularization by penalizing the model, Batch normalization and Dropout as mentioned earlier, weights constrain by rescaling the network weights, Data Augmentation by expand the dataset, or image data generators to increase the dimension of images.[7] There is a empirical evidence of overfitting: when the training accuracy is extremely high and the difference between training and testing is more than 5% - 10%.

Sparse\_categorical\_crossentropy vs. categorical\_crossentropy

This is a small problem I met when I trained the model. I got error massage talking about the dataset dimension. It turns out sparse\_categorical\_crossentropy could handle two or more labels’ classes and the labels is a one hot representation while categorical\_crossentropy could handle two or more classes’ labels when labels are integers.

Conclusion

In this project, a CNN architecture was built to classify the Russian handwriting letters. The model can correctly recognize the handwriting images with labels and backgrounds. At first, the original model gets more than 60% accuracy which meets the basic requirements. Then 10 different designs are applied including changes in activation function, initializers, optimizers and learning rate schedulers. The final model achieved more than 95% of accuracy without overfitting. Finally, there are research done for Batch normalization, Dropout, data augmentation，overfitting to explore the accuracy decrease and model failure during the training.

Reference

[1] Normalization, Codecademy Team, [www.codecademy.com/article/normalization](http://www.codecademy.com/article/normalization)

[2] Pralhad Gavali, J. Saira Banu,Chapter 6 - Deep Convolutional Neural Network for Image Classification on CUDA Platform, Editor(s): Arun Kumar Sangaiah, Deep Learning and Parallel Computing Environment for Bioengineering Systems, Academic Press, 2019, Pages 99-122, ISBN 9780128167182, <https://doi.org/10.1016/B978-0-12-816718-2.00013-0>.

[3] One-Hot Encoding vs. Label Encoding using Scikit-Learn, Alakh Sethi, Match 6th, 2020, Analytics Vidhya, [www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/](http://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/)

[4] what is Max Pooling? DeepAl, deepai.org/machine-learning-glossary-and-terms/max-pooling

[5] Jason Broenlee, January 16th, 2019, A Gentle Introduction to Batch normalization for Deep Neural Networks, Machine Learning Mastery, machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[6] Prateek Gulati, Sep 23rd, 2019, Batch Normalization is not a norm, Towards Data Science, towardsdatascience.com/batchnormalization-is-not-a-norm-cebfd921dadd

[7] How to Treat Overfitting in Convolutional Neural Networks, September 7th, 2020, Analytics Vidhya, www.analyticsvidhya.com/blog/2020/09/overfitting-in-cnn-show-to-treat-overfitting-in-convolutional-neural-networks/