Deep Residual Network on Image Classification with PCAM

醫工二 鍾孟辰
b09508011@ntu.edu.tw
National Taiwan University
Taipei, Taiwan

1 Abstract

In this project, I will present a series of experiments using resNet34 for the classification of breast cancer with PCAM. For homework 1, I used resNet34 for training. The AUC (0.823) was significantly lower than the PCAM benchmark (0.942) on PaperWithCode. Besides poor performance, the work also faced overfitting and long training time.

Therefore, for improvement in homework 2, I applied minibatch, transfer learning, and Perlin noise for better performance, and shorter training time. It ended up having little increase in performance (0.823 to 0.847), a slightly improve the problem of overfitting, and a significant decrease in training time.

2 Introduction

Breast cancer is one of the most common cancers around the world. According to the Ministry of Health and Welfare in Taiwan, ten thousand women were diagnosed with breast cancer, and approximately 2,000 died every year. Breast cancer is one of the most thriving threats for women. However, with early diagnosis and well treatment, a high percentage of the patients can fully recover from breast cancer.

Deep learning is now widely used as an efficient tool for medical image classification and segmentation. In the long-term vision, I would like to present a CAD system for breast cancer image classification. While in the short-term action, I would show a classification task with PCAM. With numerous well-performed models, I've chosen resNet34 for it is easy to program and is a state-of-the-art network in image classification. Transfer learning and an augmentation strategy using Perlin noise were both adopted for enhancement. The experiments; therefore, can be categorized into three stages, which are homework 1, applying transfer learning, and adding Perlin noise.

Three major problems need to be solved in the first stage, which are poor performance, overfitting, and a long training process. I've come up with approaches such as active learning, minibatch, transfer learning… and so on. Eventually, I've chosen to implement Transfer learning, minibatch, and

Augmentation with Perlin noise. The reasons are that these methods are easy to implement, and are believed to have good effects on the reports in class.

PCAM is used in this project. While screening mammography is the medical image that is most widely used to diagnose breast cancer, there was no suitable dataset for training. PCAM contains 327,680 color images extracted from histopathologic scans of lymph node sections, with a size bigger than CIFAR10, and smaller than Imagenet. Therefore, it is said to be trainable on a single GPU. Since the GPU resource I could access is limited, the size of PCAM seems to be small enough to be trainable while big enough for training.

This report is constructed as follows. In Section 3, I will introduce the dataset and methods in detail. The results of the experiences would be presented in Section 4 and then a discussion in Section 5. Finally, summarize and conclude the project in Section 6.

3 Materials and Methods

3.1 Subjects

PCAM contains 327,680 color images. The typical size of the histopathologic scans is $96 \times 96\, pixels$. In stages two and three, normalization was required in preprocessing.

The dataset was split into training data, validation data, and testing data, which include 2^{18}, 2^{15}, and 2^{15} examples respectively. In this report, only training data and validation data were used.

3.2 Augmentation

In the first and second stages, the images were randomly flipped and rotated. In addition, the images were transformed into grayscale, and both training and validation data were applied with augmentation. In stage three, the training data was added with Perlin noise, which is generated with the module perlin_noise.

3.3 Batch size

In the first stage, the batch size was chosen to be 400 to reduce training time. In the second and third stages, the batch size was reduced from 400 to 32 for more training steps per epoch.

3.4 Transfer learning

Transfer learning was adopted in stages two and three, which was aim to shorten the training time and enhance the performance. In this project, I used the pre-trained model from the module Torchvision. According to their website, the model was pre-trained in the Imagenet.

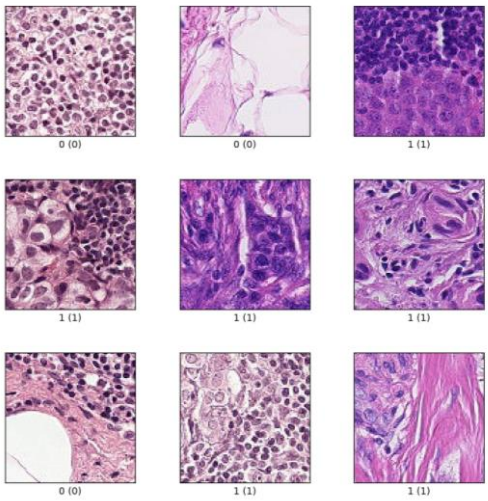

Figure 1. Examples of PCAM images

Methods / Training process	Augmentation on validation data	Normalization	Using minibatch
Previous work	✓		
Transfer learning 1	✓	✓	✓
Transfer learning 2		✓	✓
Perlin noise		✓	✓

Table 1. Summary of experiments

3.5 Data Augmentation with Perlin noise

Data augmentation with Perlin noise was proved to be able to provide good diversity to the images.[1] The utility of the method will also be evaluated in this project. Nevertheless, I would like to remark on the utility of different Perlin noises. Hence, in stage three, I generated three kinds of Perlin noise with a $96 \times 96\, pixels$ size, each of them was set with one, two, or three octaves.

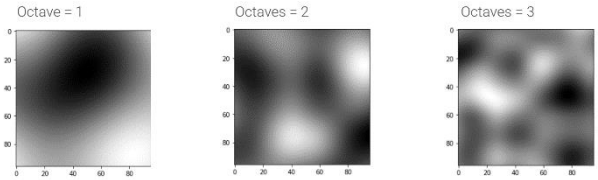

Figure 2. Three different Perlin noises

3.6 Deep Residual Network

I used a resNet34 to classify the images into two classes. The network[2] consists of an encoding path and a decoding path. The encoder is composed of increasing different layers with increasing features, while the decoder performs a global pooling and maps the output to the correct class by using a fully connected layer.

3.7 Experiments

In the first stage, I directly train the model with conventional augmentation. In stages two and three, I adopted Transfer learning, minibatch, and Augmentation with Perlin noise, anticipating better results. To evaluate the utility of Perlin noise and Transfer learning, I performed two experiments. First, I adopted conventional data augmentation and a pre-trained model. However, during this experiment, I discovered that I shouldn't have adopted augmentation on validation data. Therefore, in this experiment two models were trained. One was trained with augmented validation data, while the other did not. As for the second experiment, I adopted data augmentation with Perlin noise. In this experiment, three kinds of Perlin noise with different octaves were generated. Table 1, is a summary of the six conditions mentioned above.

[1] A Perlin Noise-Based Augmentation Strategy for Deep Learning with Small Data Samples of HRCT Images

[2] Deep Residual Learning for Image Recognition

	Stage 1	Stage 2 First experiment	Stage 2 Second experiment	Stage 3
AUC	0.823	0.808	0.836	0.847
Overfitting	Yes	Yes	Slightly	No
Training duration	2~3 days	2~3 hours	~2 hours	~2 hours

Table 2. Performances of the three stages.

Octaves	**1**	**2**	**3**
AUC	0.802	0.847	0.727

Table 3. Performances of augmentation with different Perlin noises.

4 Result

The comparison of the three stages is summarized in Table 2.

4.1 The first stage

In the first stage, the self-built model was trained with conventional augmentation. The AUC of the R.O.C curve was 0.823 with a stopping point at 20 epochs. Compared to the result on PaperWithCode, in which AUC is 0.942, the model was not well-trained. Moreover, the training accuracy haven't converged before the validation loss began to raise. Overfitting happened in this training process. During the training process, an epoch took nearly an hour to train on Colab, 9 hours for every 5 epochs on Kaggle. The whole training process took 2~3 days, which would be an obstacle for further experience. In the coming stages, I aim to overcome the poor performance, overfitting, and long training process.

4.2 Stage two

In this stage, transfer learning was adopted into the model. From Table 2, the model showed a lower AUC with the augmented validation data and a higher AUC with proper validation data. Under the same condition, in which validation data is with augmentation, the transfer learning model didn't achieve a better AUC as expected, and overfitting still exists. However, the training duration has a significant decline.

4.3 Stage three

In stage three, data augmentation with Perlin noise was adopted. In this stage, I have achieved the best AUC (0.847) among all three stages. Furthermore, by comparing the results of the second experiment in stage 2 and stage3, we can say that the overfitting was overcome.

The AUCs for each augmentation with Perlin noises are shown in Table 3. Before we

start to look at the table, I would like to discuss some errors that were made in this experiment. During the training process of octave set with 1, the preprocessing and the augmentation were redone. What's the matter is that the Perlin noise was regenerated, too. As shown in Table 3, we know that different Perlin noises with different sets of octaves make different AUCs. Yet, how would the different Perlin noises with the same set of octaves affect the AUCs is unknown. As a consequence, the hypothesis I made in the next paragraph might exist a significant difference.

From Table 3, the model accomplishes the highest performance adopted augmentation with Perlin noise generated with the set of two octaves (0.847). Interestingly, the models that adopted augmentation with the other two Perlin noises end in significantly low AUCs, which is even lower than the model with conventional augmentation (0.836).

5 Discussion

Compared with stage one, stages two and three showed a better AUC and a remarkable loss in training duration. Moreover, the overfitting was overcome with the adoption of Perlin noise in augmentation. Notwithstanding, some errors made during the experiments might cause a less convincing result. The most severe error is that the stopping point was not set. As the result, the model wasn't stopped at the best timing. Thereby, the highest AUCs might not be recorded the outcomes exist an unknown deviation.

5.1 Perlin noise

Compared with Perlin noise generated with one or three octaves, augmentation adopted Perlin noise generated with two octaves has a notable high AUC. This perfectly shows the limitation of Perlin noises that they sometime might not be able to represent the target pattern.

5.2 Future improvement

Even though I have brought about higher AUCs in stages two and three, there remains a enormous gap between the models on PaperWithCode and mine. For future improvement, I plan to apply an early stop to prevent overfitting. Moreover, I believe that a deeper model could bring me a better AUC. Last, I might consider adding steerable filter in augmentation for the leading models on PaperWithCode use it in their work.

6 Conclusion

I have built a model for classification on PCAM. During the semester, I've learned numerous methods that might finer the train. Despite that, my programming ability is limited. In consequence, I've chosen the methods I am capable of, which are, minibatch, transfer learning, and augmentation with Perlin noise. These methods play a role in increasing AUCs. Nevertheless, some critical errors lead to inconvincible results. Even so, I've learned valuable lessons and skills. These errors would be stepping stones in my future works.

And maybe someday, after years of trials and errors, I would be able to build a model that could fulfill my purpose, and perhaps, it could bring benefits to human health care.

7 Reference

1. S. Keshav How to Read a Paper http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/07/paper-reading.pdf
2. Li Shen Deep Learning to Improve Breast Cancer Detection on Screening Mammography https://www.nature.com/articles/s41598-019-48995-4
3. Kaiming He, Xiangyu Zhan,g Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385
4. PCAM dataset https://github.com/basveeling/pcam
5. Hyun-Jin Bae A Perlin Noise-Based Augmentation Strategy for Deep Learning with Small Data Samples of HRCT Images https://www.nature.com/articles/s41598-018-36047-2