Multi-label image classification with Convolutional Neural Networks

Yang Ge (450005028), Suwan Zhu (480090900), Tianyao Xu (480268059)

Abstract

Deep learning is a technique widely used in various of image classification tasks. In this project Convolutional Neural Networks (CNN) were applied for a multi-label image classification task. The models explored in this project includes ResNet50, DenseNet169, DenseNet201 and VGG16. The influences of various parameters and optimized methods on these models were also discussed in this project.

Keywords: Deep learning, Classification, Convolutional Neural Network, ResNet50,DenseNet169, DenseNet201, VGG16.

1.Introduction

1.1 The aim of the study. To complete a multlabelled image classification task and achieve a higher accuracy, four different Convolutional Neural Networks are applied in this project. Totally, 31925 multi-label images are provides to train a model and after that, 15516 images will be classified. The performance of different models will also be compared.

1.2 The importance of the study. Deep learning is a powerful machine learning method in Artificial Intelligence. Currently, a very important application of deep learning is image classification. It is important for autopilot. E.g. recognizing traffic signal, traffic light, vehicles, pedestrian. The CNN network has been developed many optimized architectures. We want use the latest CNN architecture to test the accuracy with different parameters and explore which one can achieve a better accuracy on our data set.

2. Techniques

2.1. Convolutional Neural Network

VGG16. VGG16 is a Convolutional Neural Network created by Oxford University's VGG group. It's a improvement over AlexNet by the use of several consecutive 3x3 convolution kernels instead of the larger convolution kernels (11x11, 5x5) in AlexNet. The 3x3 convolution kernels also help to maintain image properties. Through the use of the use of stacked small convolution kernels, the number of nonlinear layers becomes larger, which makes network can be deeper and perform better in more complex learning task. Besides, compared with the use of large convolution kernels, VGG16 has less parameters and cost less when trained. Below is a overview of VGG16.

ResNet50. In order to solve the problem of gradient degradation that occurs when training a very deep network, Kaiming He proposed the Resnet structure. Through the use of Resuidal learning, the number of layers in the network can be greatly improved. To reduce the difficulty of optimization, ResNet uses a shortcut to change the identity mapping of the unknown function H(x) that needs to be learned to become a function that approximates F(x)=H(x)-x. Through a reformulation, a problem is decomposed into multiple scale direct residual problems, which can obviously optimize the training process. ResNet also proposes a BottleNeck structure for deeper networks (layers greater than or equal to 50), which reduces the time

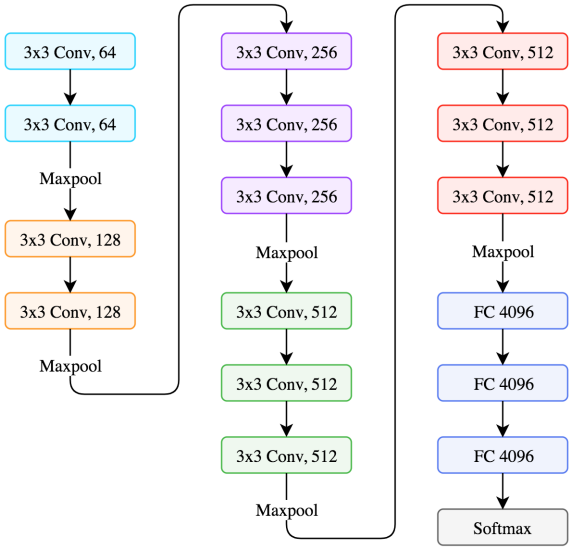

Figure 1. VGG16 Convolutional Neural Network

complexity of the operation. Bellows are views of some ResNet's elements.

Residual Learning:

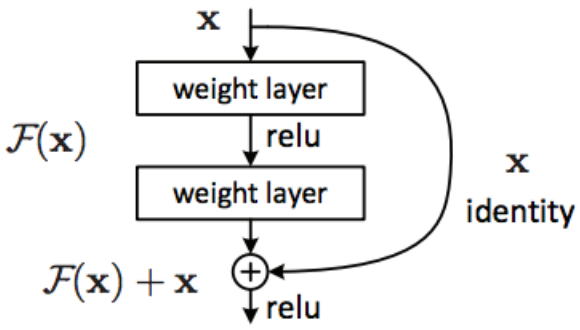

Figure 2. Residual Learning

Residual learning method is completed by constructing a Residual block. As shown in Figure 2, after introducing the residual structure, the identity mapping of the unknown function H(x) that originally needed to learn approximation becomes one that approximates F(x)=H(x)-x function, which makes it easier to optimize.

BottleNeck:

As shown on the right side of the figure3 above, when studying a deeper (layers greater than 50) network ResNet adapt a BottleNeck structure to improve efficiency. We can see that in the Botleneck, the first 1x1 convolutional layer is used to

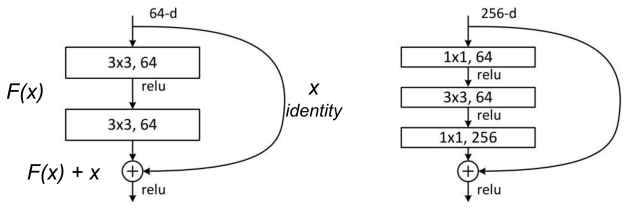

Figure 3. BottleNeck

reduce the latitude (to reduce the computational complexity), while the second 1x1 convolutional layer is used to increase the latitude and keep it to the original Input has the same latitude.

ResNet:

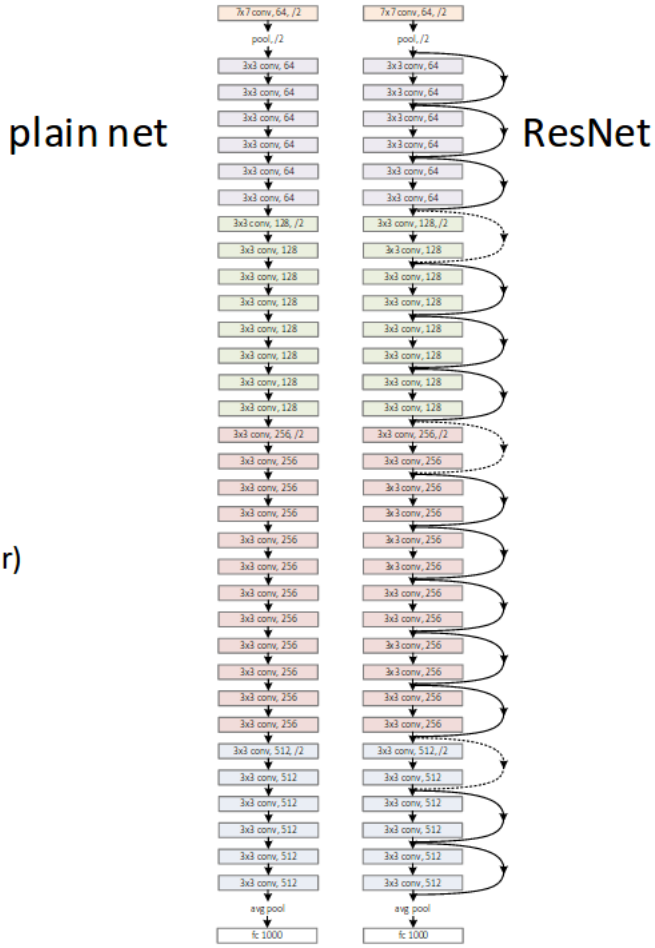

Figure 4. ResNet

The right graph of figure4 is a Resnet network diagram with an identity map. The black solid line represents the identity map at the same latitude (the

number of convolution kernels is the same). The dotted line refers to the identity map between different dimensions (the number of convolution kernels is different).

DenseNet. Different form ResNet and VGG16, it improve the model through feature reuse and bypass settings (Bypass). It has fewer parameters and alleviates the generation of the gradient vanishing problem, whcih makes the model easier to train. Combined with the assumption of information flow and feature reuse, DenseNet was named as CVPR's Best Paper in 2017.

In a traditional convolutional neural network, each layer's input just comes form it's previous layer which means each layer only connected with it's previous and next layer. However, in DenseNet each layer is connected with all of it's previous layers. Bellow is a graph to explain the theory.

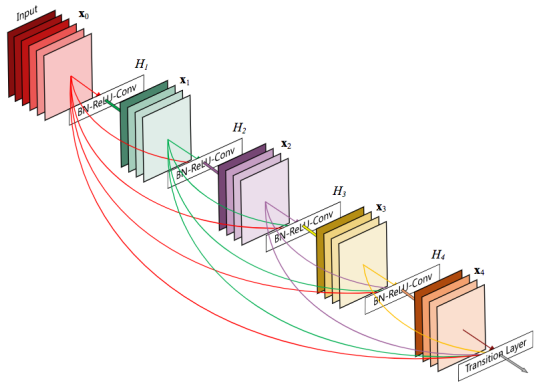

Figure 5. Dense Block

As shown in above figure5, X_0 is the input, therefore, the input of H_1 is X_0, the inputs of H_2 are X_0 and X_1, the input of H_3 are X_0, X_1 and X_2, and so on. Due to the design of this dense block, compared with other Convolutional Neural Networks DenseNet is narrower and has fewer parameters.

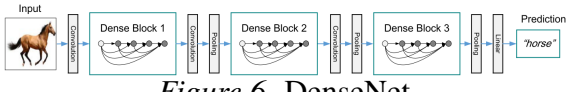

Figure 6. DenseNet

Above figure6 is a DenseNet structure diagram which contains 3 dense blocks. Feature map sizes in each block is unified so that there is no size conflict problem when doing concatenation.In addition, In fornt of 3 by 3 convolution of each of the dense block, there is a 1 by 1 convolution operation which is called the bottleneck layer. The bottleneck layer works to reduce the number of feature maps input, also as the amount of calculation. Besides, it also combines the characteristics of each channel. Finanlly, in order to further compress the parameters, a 1 by 1 convolution operation between each two blocks is added, this is called Translation layer.

2.2 Transfer Learning

Transfer learning is a method to migrate the learned model parameters to a new model to help the new model training. Considering that most of the data or tasks are related, we can learn and transfer the learned model parameters (also known as the knowledge learned by the model) to the new model in a certain way to accelerate and optimize. After that, the learning efficiency of the model does not have to learn from zero like most networks.

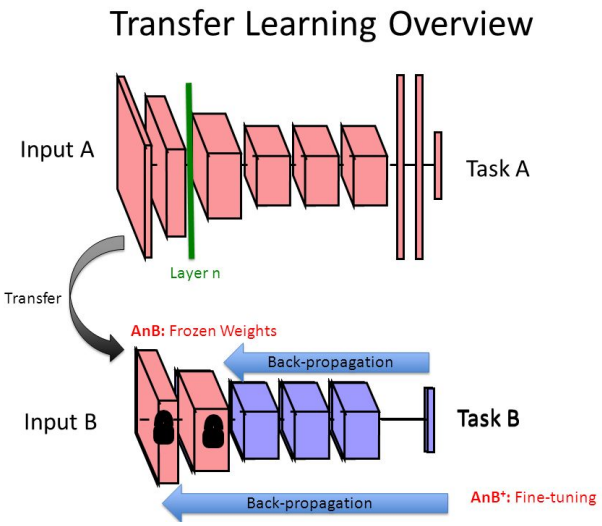

Figure 7. Transfer Learning

In this project, transfer learning is applied to reduce cost and increase the accuracy of the models.

2.3 Data Augmentation

When training a deep network, the number of parameters which need to be learned also increase, which leads to over-fitting problems. Besides, when the data set is not large enough, too many parameters will fit all the characteristics of the data set including the noisy data. To avoid the over-fitting problems to happen, data augmentation is applied. The methods of image data augmentation we used include:

- Rotation
- Zoom
- Shift images horizontally
- Shift images vertically
- Flip images

3. Experiments and Discussions

3.1. Input

Datasets. The first dataset has 31925 images and 20 classes, each image could belong to more than one class. We used 80% of them for training and the rest for testing. The second dataset has 15516 images. We predicted labels on the second dataset and then submitted them.

3.2 Measurement methods

Accuracy Accuracy formula is an important index of performance, which aims to represent the algorithm performance. The formula of accuracy metric is defined as:

$$Accuracy = \frac{number\ of\ correct\ classifications}{total\ number\ of\ test\ examples} \tag{1}$$

As one sample could have multiple labels in this assignment, correct classification implies that predicting at least one correct label and no incorrect label.

3.3 Experiments

In the experiment, we used pre-trained models from ImageNet as base models As the data used in this assignment were multi-labeled, we removed the output layer in the original pre-trained models, replaced with our output layer, which had 20 classes and sigmoid activation function, then added 2 fully connected layers between base model and output layer. We used binary cross entropy or focal loss as our loss function. Then we applied data augmentation, did experiments on DenseNet169, DenseNet201, ResNet50 and VGG16 with batch size 32.

Experiment 1. In this experiment, we froze the internal parameters in pre-trained models. We just trained the parameters in two fully connected layer and output layer.

Epoch	Accuracy	run time(s)
10	0.725	4452
15	0.732	6674
20	0.734	8789

Table 1
Freeze parameters. DenseNet169 with binary cross entropy loss

Epoch	Accuracy	run time(s)
10	0.735	4392
15	0.731	6553
20	0.738	8684

Table 2
Freeze parameters. DenseNet169 with focal loss

Epoch	Accuracy	run time(s)
10	0.726	4860
15	0.730	7296
20	0.741	9731

Table 3
Freeze parameters. DenseNet201 with binary cross entropy loss

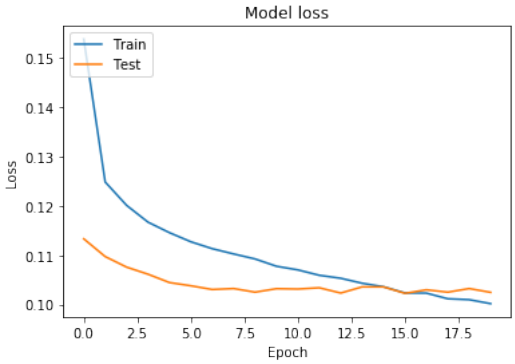

Figure 8. Freeze. Loss of DenseNet201 with binary cross entropy loss

Epoch	Accuracy	run time(s)
10	0.727	4723
15	0.738	6954
20	0.736	9452

Table 4

Freeze parameters. DenseNet201 with focal loss

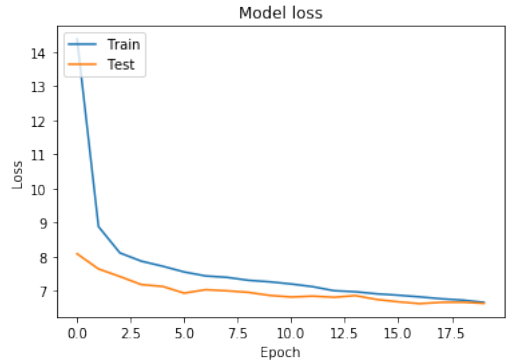

Figure 9. Freeze. Loss of DenseNet201 with focal loss

Epoch	Accuracy	run time(s)
10	0.688	5443
15	0.699	8155
20	0.694	10744

Table 5

Freeze parameters. ResNet50 with binary cross entropy loss

Experiment 2. In this experiment, we did not freeze the internal parameters in pre-trained models. We used the parameters pre-trained from ImageNet as starting point, trained all parameters in our models.

Epoch	Accuracy	run time(s)
10	0.578	5430
15	0.570	8345
20	0.583	10850

Table 6

Freeze parameters. VGG16 with binary cross entropy loss

Epoch	Accuracy	run time(s)
20	0.410	14025
30	0.539	22253

Table 7

Not freeze. DenseNet169 with binary cross entropy loss

Epoch	Accuracy	run time(s)
20	0.381	15334
30	0.363	22877

Table 8

Not freeze. DenseNet201 with binary cross entropy loss

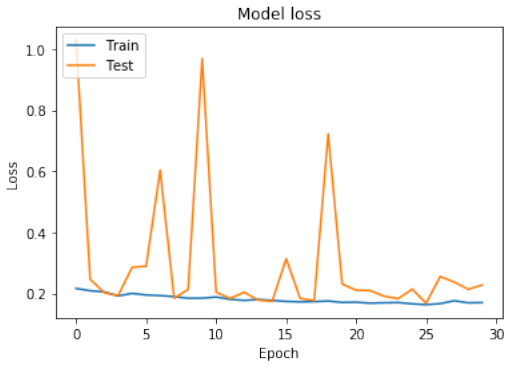

Figure 10. Not freeze. Loss of DenseNet201 with binary cross entropy loss

3.4 Experiment Analysis

We used and compared several different techniques in the experiments, which were VGG16,

Epoch	Accuracy	run time(s)
20	0.417	16274
30	0.448	24568

Table 9
Not freeze. ResNet50 with binary cross entropy loss

ResNet, DenseNet, transfer learning, binary cross entropy loss and focal loss. DenseNet169 and DenseNet201 had similar results, DenseNet201 had a little bit higher result also the best result among all experiments, which was 74.1%.
Binary cross entropy loss and focal loss showed similar results in the experiments, as they were very similar. We could get better result if we can find best parameters in focal loss.
Not freezing parameters generated lower results than freezing did, it also cost more time because all internal parameters were trained, and was very difficult to converge.

4.Person reflection

What we learned. In this assignment, we used three different Convolutional Neural Network structures, which were VGG, ResNet and DenseNet. All those network structures are powerful in image classification problem. We then learned several techniques in deep learning, which were transfer learning, data augmentation. Transfer learning helped us to quickly train our models in a not very large dataset. Data augmentation can be used to generate new data and deal with overfitting problem.
Shortcomings. We were lacking of hardware. We used Google Colab, which just provide limited memory and GPU power, in this assignment. It was very difficult for us to do experiments and tests. Experiment on not freezing parameters could generate better results, but it was too time-consuming, we did not train much epochs on it.
There were many hyper-parameters, it is also difficult to find the best combination.

5. Conclusion

In conclusion, we used and compared three different Convolutional Neural Network structures, which were VGG, ResNet and DenseNet, applied transfer learning and data augmentation in this multi-label image classification assignment. The highest accuracy result we got was around 74% by using DenseNet201 and freezing its internal parameters. Not freezing parameters generated lower results than freezing did, it also cost more time because all internal parameters were trained, and was very difficult to converge. Finally, we choose the model "DenseNet201" as our final CNN model.

References

[1] Bishop, C. M. (2006). Pattern recognition and machine learning. New York: Springer

[2] Cheng, W., Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. Machine Learning, 76(2-3), 211-225. doi:10.1007/s10994-009-5127-5

[3] G Huang, Z Liu, L Van Der Maaten. (2017). Densely Connected Convolutional Networks.

[4] K He, X Zhang, S Ren, J Sun .(2016). Deep Residual Learning for Image Recognition.

[5] K Simonyan, A Zisserman.(2014). Very Deep Convolutional Networks For Large-Scale Image Recognition.

Appendix

1.Hardware and Software.

- Software and hardware: Google Colab
- Interpreter: Python 3
- Libraries: tensorflow 1.13.1, numpy, matplotlib, pandas, tarfile

2.Data Set.

- train.tar.gz
- train.txt
- test.tar.gz

3.Code Instruction. Code was written in Python3.

To run our code. :

1. Use Google Colab to run train.ipynb and predict.ipynb Use google drive, create a directory called "comp5329assignment2", upload train.tar.gz, train.txt,test.tar.gz, two ipynb file and my_model.h5 to google drive, and use Google Colab to open the files and run. Make sure using GPU mode.

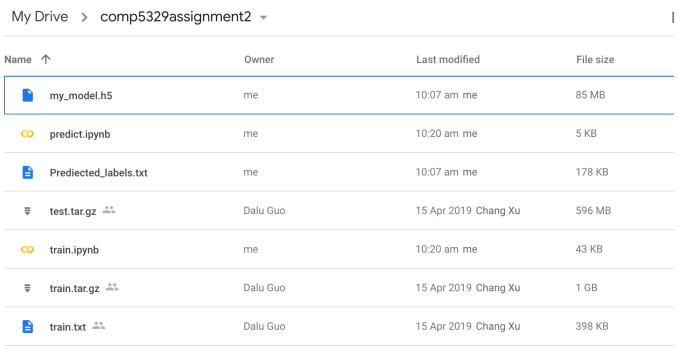

Figure 11. Google Colab directory and files

Figure 12. Go to the link

Figure 13. Copy the code from the link, press "enter", then run the rest cells

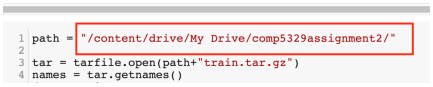

Figure 14. local method

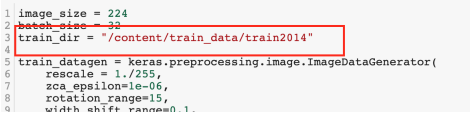

Figure 15. local method

2. Change the "path" to local path, e.g. "path = "../Input/"". We did not test this, it should work. Also need to change tar.extract path, and train_dir and test_dir.

Figure 16. local method

To load the model. :

from tensorflow.keras.models import load_model
model =load_model("../Output/my_model.h5")

To load Prediected_labels.txt. :

import pandas as pd
predicted_test_data = pd.read_csv('../Output/Prediected_labels.txt', sep="\t", *header = None*)

4.contribution of group members. The code were written in team work as each person in charge of particular parts of code, also as the work of data collection and analysis. After all coding work finished, we wrote the report together through using Overleaf. All in a word, each group member did the same contribution.