Multi-label Classification Using Transfer Learning

Zhuqiang Lu [*] **Pengwei Sui** [*]

Abstract

Multi-label classification is rapidly developing as an important aspect of modern predictive modeling, motivating study of its theoretical aspects and implementation. In this report, we discuss how to perform multi-label classification using modified Keras pre-trained CNN model. Due to the constraint of model size, only ResNet and DenseNet are considered in this report. In addition, we discuss the difference of model structures for multi-label classificaiton and multi-class classification. The best performance in terms of accuracy we obtained is 74.8%, we will further discuss the algorithm to calculate accuracy and analysis the factors that affect the performance.

1. Introduction

Transfer learning has shown great success in the past few years. A well pre-trained model can be used as a part to build a new model to solve similar problems, such as a single label classification model can be used to build a multi-label classifier. However, due to the different structuring, selecting a pre-trained model that fits the problem the best might be challenging.

This assignment has a solid constraint on the size of the model, this helps to select pre-trained model for the assignment. However, the follow-up model structuring and parameters choosing are still required.
In this report, we first discuss the principle of different techniques we used to structure the models as well as the techniques adopted to pre-process data. Then we compare the performance of different models and analyze the potential factors that cause the difference

[*]Equal contribution . Correspondence to: Zhuqiang Lu <zhlu6105@uni.sydney.edu.au>, Pengwei Sui <psui3905@uni.sydney.edu.au>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

1.1. Aim of the study

The primary aim of this study is to prove that we have a solid understanding on how CNN works as well as we can fluently use Tensorflow.Keras library to build a deep learning model. The secondary aim of this study is to build a high performance classifier with a reasonable model size to solve the given problem.

1.2. Importance of Study

Fluently using a library to structure a deep learning model is essential for studying deep learning, as always implementing the whole model is unrealistic and inefficient to experiment the performance of a model. In real world, an image is not necessary to have only one label, instead, there more are multiple. Training a single label classifier might not be enough to solve the problems.

In addition, the experience of designing a multi-label classifier leads a student discover how real-world deep learning models work and understand how the architeture of a model affects the performance. What cannot be ignored is this also drive students to learn what are not covered in the lecture, such as facal loss, hamming loss for multi-label classification.

2. Techniques

In this section, we first present the the Pre-processing method adapted, then we discuss the principle of different modules.

2.1. Data prepossessing and argumentation

Building an effective neural network model requires careful consideration of the network architecture as well as the input data format B (2019). Based on the training dataset we have, in order to make sure that the size of all input training image are uniform, we re-scale each data image into 300x300 RGB form by specifying $IMAGE\_SIZE$ configuration variable of Keras to 300 in prepossessing stage.

Once we ensure that all input image mapped to uniform scale, we perform combination of the affine image transformations and color modification to the training dataset by applying traditional transformation of Keras data argumentation for each training epoch. The affine transformation includes rotation, refection, scaling (zoom in/out) and shearing during training 1. Based on the numbers of training image of 20 class label (31985), which approximate 1600 training examples for each single label. Using data argumentation is the one of the fastest and easiest way to increase the training dataset and significantly improve the learning abilities of our experiment model.

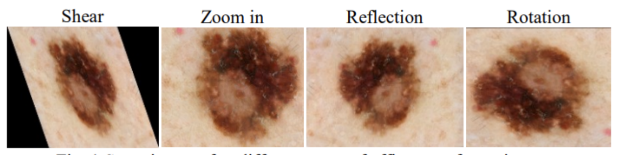


Figure 1. Same image after different types of affine transformations Mikoajczyk & Grochowski (2018)

2.2. Best Performance Model Overview

In this section, we will briefly discuss about the architecture and modification we done of the our best performance model. The detail of a number of model we experiment will be discuss in later section3. The entire structure of the best experimented model can be summarized as follow:

- *Pre-trained DenseNet-169*(Huang et al. (2016)): A deep DenseNet with 4 dense block which exactly follow the same Dense block layout shown in Figure 2 for each. Each Dense block of the model is followed by a composite function(including a rectified linear unit (ReLU) and a 3 3 convolution (Conv)) and a pooling layer 3. In order to deeply improve the information flow between each layer and the dense connectivity, the output of DenseNet-169 will be a feature map and the size of it will according to the dimension of given input image sample.
- *3 Fully-connected layer*: After we get the feature map output by DenseNet-169, there are three fully-connected layer (2048D, 1024D, 512D, 20D) connected after the final pooling layer of DenseNet-169. Between each two layers, Dropout(0.5) is applied to prevent over-fitting during the training process.
- *Output layer*: In output layer, we use Sigmoid activation function instead of Softmax. If we apply Softmax activation function in output layer, the probability of all other labels will decrease when the probability of one label increasing which is not adaptable for muti-label classification.

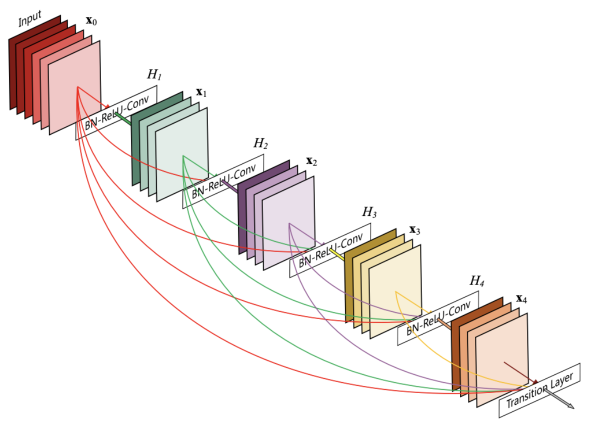


Figure 2. A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input. Huang et al. (2016)

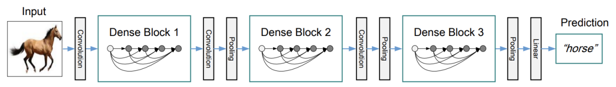


Figure 3. General Architecture of DenseNet-169 Huang et al. (2016)

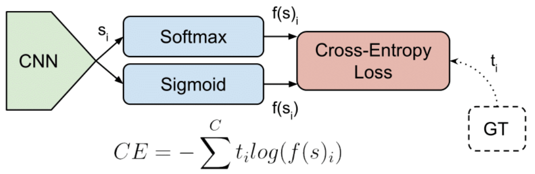
2.3. Evaluation technique

For the evaluation technique we used to calculate the average loss of training example in single epoch, the build-in loss formula *Binary cross-entropy* 4, *Categorical cross-entropy* 1 and *Argmax* are chosen by our model to contribute during training and monitoring. Argmax is that loss function that will select the predicted label with the high score in output layer and compare with the truth label for each training example which given us a "lower bound" reference of the training accuracy of current model during experiment.

$$CE_{categorical} = -\sum_{i}^{C} t_i \log\left(f(s)_i\right) \tag{1}$$

3. Experiments and results

In this section, we mainly discuss the performance difference of different model and we analyze what leads to the performance differences. At first, we study the how the choice of pretrained model affects the overall performance,



$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

Figure 4. Binary cross-entropy Huang et al. (2016)

Table 1. model structure

Model name	pretrained block	Dense block
model 1	ResNet50	(2048, 1024, 512, 20)
model 2	ResNet50	(1024, 512, 20)
model 3	DenseNet169	(2048, 1024, 512, 20)
model 4	DenseNet169	(2048, 1024, 20)
model 5	DenseNet121	(2048, 1024, 512, 20)
model 6	DenseNet121	(2048, 2048, 512, 20)
model 7	DenseNet121	(512, 20)

then we compare how the follow-up structuring of a model affects the accuracy of the model.

3.1. Hardware and Software specification

All models mentioned in this report are trained using Google Colab GPU, RAM: 12GB, Disk: 358GB, GPU: Nvidia K80 Graphic Card.

3.2. Models introduction

Table 1 has shown the structures of different models. The column Dense Block refers to the trainable layers and the number of neurons in each layer. The pretrained block are not trainable due to the limited GPU memory. In addition, pretrained block should not be trained, since we are building a multi-label classifier on image, and the pretrianed blocked are trained as a single-label classifier, considering the high similarity of these two tasks, it it wise to leave the pretrained block untouched.
In this case, the pretrained block is served as a feature maps extractor, and the dense block is a normal deep neural network with sigmoid as final activation function. All fully connected layers are followed by a dropout with drop out rate of 0.3, activation functions are relu if not the final layer. Howeve, there is an exception, model 7 did not use any dropout in order to study whether droupout is needed in this problem.

Table 2. model accuracy

Model name	one-label accu	all label accu
model 1	73.2%	47.3%
model 2	72.7%	47.0%
model 3	74.8%	49.1%
model 4	72.9%	49.8%
model 5	73.5%	48.3%
model 6	73.1%	47.1%
model 7	73.2%	45.1%

Table 3. model size and runtime

Model name	size(MB)	run time(sec)
model 1	185	7221
model 2	162	7324
model 3	130	6934
model 4	99	6789
model 5	87	6899
model 6	92	6590
model 7	66	6203

3.3. Accuracy

From the Table 2, it is surprising that all listed models have a similar accuracy with 10 epochs and 32 batch size. Due to the use of validation set and early stopping according to the value of loss, none of theses has encountered obvious overfitting problem. However, due to the keras built-in accuracy algorithm is based on the number of correct prediction of each example, it doesn't reflect that fact the number of miss prediction
To evaluate the accuracy of the models, we design our own accuracy metric algorithms: single-label accuracy and all-label accuracy. The algorithm of the single label accuracy is calculated as follow: only select the predicted label with the highest probability, if the sample's ground-true label contains it, this the label(output) of this sample is counted as correctly predicted, otherwise, this label is viewed as a wrong prediction. The all label accuracy is calculate by rounding the probability of each predicted label of a sample data, then compare to the ground-true label. Miss prediction or wrong prediction of a single label will make the whole prediction of this sample image be counted as wrong.

The single label accuracy algorithm cannot reflect the overall performance of the model, however, it ensures maximum marks that can be earned, therefore, the single label accuracy metric is chosen as the overall model accuracy metric.

3.4. Model Size

The size of model is also one of the main concern of this assignment. Due to ResNet is a large CNN model by itself, the final models with ResNet as pretrained block usually have much larger size. In contrast, DenseNet is design for small size, therefore, those models with DenseNet as pretrained block are generally smaller. However, considering the total trainable parameters, even with DenseNet as pretrained block, the sizes of some models are still go over 100MB.

The size of a model does not simply affect the time it requires train the model, but more importantly, the time needed to predict. To maintain the relatively high performance and reasonable predict run time, a model should not be smaller than 100MB as the assignment specification suggests. Table 3 shows the size of different models

3.5. Extensive Analysis

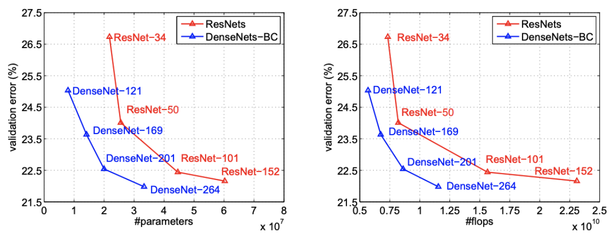


Figure 5. DenseNet vs. ResNet Huang et al. (2016)

This section is made based on the accuracy of a model (see table 2 and table 3). What out of expect is the accuracy is independent of the size of a model, a large model can do similarly but not significantly better than a smaller model. This could be caused by the high efficiency and good performance of DenseNet(see 5). It is fair to assume that the feature maps of DenseNet and ResNet are similar in terms of information they contains. Then a feature map is fed to the dense blocks which can be viewed as a normal multi-label classifier deep neural network. Due to the similarity of feature maps that DenseNet and ResNet can return, the size of them would not affect the performance of a model.

At the point, it is fair to say the performance of these models are solely affected by their own structures. Since they all use the same optimization algorithm(RMSProp), same dropout rate(0.5), the best accuracy is achieved by the one which has the deepest network, as expected.

However, the even the best accuracy is still lower than 75%, the undesired performance partly comes from the structure as it can potentially be improved. The main reason of this performance is insufficient amount of data. There are in total 20 labels, and the number of training image is around 31000. On average, there are less than 1600 training example for each label, even with data augmentation, the performance of the models are still unsatisfying. In addition, due to the meaning of each label is unknown, it is time consuming and hard to collect more data for training.

4. Discussion

According to section 3, we found that when apply transfer learning, since the body of pre-trained model are not trainable, the quality of the feature map the pre-trained model returns can directly affect the overall performance of the model. However, the trainable part (also known as dense block in this context), can be structured to fit the need.

For the report, the overall goal is to build a multi-label classifier. Due to the limit of keras library, in this report, we did not use some more advantage loss function such as hamming loss and facal loss. If more training data is given or collected, a better performance can be achieved.

5. Conclusions

The multi-label classifier can be further improved in multiple ways, such as using a better structured dense block, a finer tuned parameters, some other loss functions or pre-trained models if having more training data is not possible. This experiment has shown how the pretrained model can affect performance of a model that uses it in transfer learning.

References

B, N. Image data pre-processing for neural networks. *Becoming Human: Artificial Intelligence Magazine*, 3(3), 2019.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Mikoajczyk, A. and Grochowski, M. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary*

PhD Workshop (IIPhDW), pp. 117–122, May 2018. doi: 10.1109/IIPHDW.2018.8388338.

APPENDIX

- `cd Algorithm` then `python3 predict.py`
- You can change some default parameters for the prediction such as the number of samples you want to predict
- Make sure you don't change the `LABELS` in `CONFIG.py` unless you know what you are doing
  - due to my lazyness, I did not specify the class-label dictionary during training, the given dictionary is generated by the library
  - If you retrained the model, you can use the following code to get the new library if you forgot to specify: `your_generator.class_indices()`
- Only the predict code is given, which can be found in `predict.py`
  - OK, I did include the `train.py`, but it is generated by notebook
- Also, if you want to use some other trained model with the `predict.py`, make sure you specify the custom objects correctly.
- Enjoy