# COMP5329 assignment 2


| Yujun Liu | Yueying Sun | Shaoqi Xun |
| --- | --- | --- |
| 480422097 | 470169265 | 480529282 |

# Contents

**1 Introduction** **3**

**2 Datasets Information** **3**
2.1 Preprocessing . . . . . 4

**3 Techniques** **4**
3.1 Convolutional Layer . . . . . 5
3.2 Pooling Layer . . . . . 5
3.3 ReLU Layer . . . . . 5
3.4 Fully Connected Layers . . . . . 5
3.4.1 Dropout . . . . . 5
3.5 Adam optimizer . . . . . 6
3.6 Nadam Optimizer . . . . . 6
3.7 Residual learning . . . . . 7

**4 Experiment and Results** **7**
4.1 Self-constructed model . . . . . 7
4.2 Pre-trained Model . . . . . 9

**5 Conclusions and Discussion** **11**

**A How to run the codes** **14**

**B Predicted outputs for test data** **14**

**C predicted label model** **14**

**D Hardware and Software Specifications** **15**

**Abstract**

*In this task, our goal is to use Convolutional Neural Network and Residual learning to perform a multi-class classification in given images. The reason to use Convolutional Neural Network is simple. It is extensively used nowadays for classification images and the performance can be significant increased compared to other methods such as MLP or KNN. To be specific, we conducted a comprehensive experiment on training a given datasets which contains 31924 images and 20 labels in order to get top accuracy. For the implementation process we used pre-trained model and our self-constructed model to obtain the accuracy as high as possible.*

# 1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques to optimizing accuracy [9]. With the huge development of computer performance, larger datasets such as LabelMe [16], which consists of hundreds of thousands of fully segmented images, and ImageNet [3], which consists of over 15 million labeled high-resolution images in over 22,000 categories, can be classified and recognized by model.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. Nowadays, Convolutional Neural Networks (CNNs) have been established as a powerful class of models for image recognition problems [7]. Images have become ubiquitous on the internet, which has encouraged the development of algorithms that can classify their properties. For example, classify their label. This can also apply to search and summarization. Recently, Convolutional Neural Networks (CNNs) [10] have been demonstrated as an effective class of models for image classifying and it can give state-of-the-art results [2]. Therefore, it is important to understand how to classify images using convolutional neural network.

In this task, we propose a convolutional neural network to finish a multi-class classification problem. Meanwhile, the shift, rotation and scaling of data augmentation are also utilized in this project. The rest of the report is organized as follows. Section 2 introduces datasets we are going to use. Section 3 illustrate techniques of CNN architecture. Section 4 describes the experiment and the results, followed by discussions and conclusions in section 6.

# 2 Datasets Information

The dataset consists 31924 real-world images which were taken by camera. Those images contain 20 labels. Notice that it may contains multiple labels for a single image. The following table illustrate the details of the first images.

| | |
|---|---|
| Dimensions | 320*214 |
| Width | 320 pixels |
| Height | 214 pixels |
| Horizontal resolution | 96 dpi |
| vertical resolution | 96 dpi |
| Bit depth | 24 |

### 2.1 Preprocessing

Since the input datasets are images, we need to do some preprocessing on it:

1. Randomly rotate images in the range (deg 0 to 180)
2. Randomly shift images horizontally
3. Randomly shift images vertically
4. Random shear (The range can be changed in code file)
5. Random zoom (The range can be changed in code file)
6. Random channel shifts (The range can be changed in code file)
7. Filling points outside the input boundaries
8. Randomly flip images horizontally and vertically
9. Rescaling
10. Change image size

In machine learning framework, to avoiding the scale which can cause a knock-on effect on learning accuracy, we always want variables to have zero mean and equal variance. Typically, for this task, since pixel values fall in the range of 0 to 255 in every image, we want to scale those values to a range of 0 to 1 before feeding to the neural network model. Therefore, we use preprocessing function to scale images to have zero mean.

## 3 Techniques

CNN is a kind of multilayer neural network with a supervised learning structure. It contains two basic components: an automatic feature extractor and a trainable classifier [13]. Although there are many different CNN architectures, the basic components in all designs are very similar whereas the feature extractor contains feature map and retrieves discriminating features from the raw images via two operations: convolutional filtering and down sampling. The basic CNNs components consist of convolutional layers, pooling and fully connected layers, which will be discussed as follow:

## 3.1 Convolutional Layer

Convolutional layers are the core building block of a CNN architecture. They focus on the exploitation of local substructure within the input (e.g., images) because pixels that are closer together in the image (e.g., adjacent pixels) tend to have stronger correlation than pixels that are farther apart within the same image [12]. The convolutional filtering kernels on feature maps have the size of n by n pixels and neurons that belong to the same feature map which shares the same weights in that layer. Thus, in CNN architectures are captured by constraining each neuron to depend only on a local receptive field.

## 3.2 Pooling Layer

It is common to periodically apply a pooling layer between successive convolutional layers in CNN architectures [1]. Down sampling operations are often performed top reduce the dimension of spatial resolution of the feature map as well as the number of parameters and computation in the network [11]. The pooling layer operates independently on every depth slice of the input and resizes it spatially. A common form is a pooling layer with filters of size 2x2 applied with a stride of 2 in both width and height, discarding 75% of the activations. Every max operation is over 4 numbers so depth dimension remains unchanged after pooling operations. It is worth noting that pooling sizes with larger receptive fields can be too destructive [1].

## 3.3 ReLU Layer

ReLU is not a separate component of the convolutional neural networks' process. It effectively removes negative values from an activation map by setting them to zero [11]. The purposeof applying ReLU is to increase the non-linearity in our images. The function is defined as follow:

$$f(s) = max(0, s) \tag{1}$$

## 3.4 Fully Connected Layers

The output from the convolutional layers represents high-level features in the data. The output could be flattened and connected to the output layer, adding a fully connected layer is usually a sufficient way of learning non-linear combinations of these features. Essentially, the fully connected layer is learning a (possibly non-linear) function in that space.

### 3.4.1 Dropout

Dropout is a regularization technique, which aims to reduce the complexity of the model with the goal to prevent overfitting. In CNN, the dropout drops connections of neurons from the dense layer to prevent overfitting. Specifically, it is a regularization where it randomly sets some of the dimensions of the input

vector to be zero with probability keep_prob. The remaining dimensions which are not set to zero are scaled by $\frac{1}{keep\_prob}$

### 3.5 Adam optimizer

Adaptive Moment Estimation (Adam) [8] is another adaptive learning rate method to update parameters. It takes both advantages of Adgrad and MSprop which are two prevalent [8]. That is, it is not only can store the exponentially decaying average of past squared gradient, but also keep exponentially decaying average of past gradient [17].

Therefore, Adam can achieve a good result faster than Adgrad, RMSprop and AdaDelta etc [15]. It is suggested that this method achieves best overall results amongst the adaptive learning algorithms mentioned above. Moreover, Adam introduces the bias-correct concept [15]. Thus, Using Adam can acquire better accuracy rather than other learning method. The pseudo code will provide as follows to help explaining how these parameters work with SGD, and implementation details can be seen in codes file:

SGD with Adam

Initialization: parameter θ , time step t = 0 , two moment variables: m =0, r=0

Require : learning rate η, small value for stabilization δ (default : $10^{-8}$)

beta 1 & beta 2 for representing the exponential decaying for moment $\beta 1, \beta 2$

**While** stop convergence criterion not met **do**:

Compute gradient estimate: gradient (g)

t ← t+1

Compute biased first-moment update: $s \leftarrow \beta 1 s + (1 - \beta 1) g$

Compute biased second-moment update: $r \leftarrow \beta 2 r + (1 - \beta 2) g \cdot g$

Compute correct bias for first & second moment: $s' \leftarrow \frac{s}{1-\beta 1^{(t)}}$ $r' \leftarrow \frac{r}{1-\beta 2^{(t)}}$

Compute the update: $\Delta\theta \leftarrow -\eta \frac{s'}{\sqrt{r'}+\delta}$

$\theta \leftarrow \theta + \Delta\theta$

Figure 1: Adapted from Deep Learning (Goodfellow, 2016, p. 311)

### 3.6 Nadam Optimizer

Nadam is Adam RMSprop with Nesterov momentum. In rewritten NAG, we would take the first part of the step and apply it before taking the gradient of the cost function. The algorithm will be given as follow:

**Algorithm 8** Nesterov-accelerated adaptive moment estimation

$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$
$\hat{\mathbf{g}} \leftarrow \frac{\mathbf{g}_t}{1-\prod_{i=1}^{t} \mu_i}$
$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$
$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\prod_{i=1}^{t+1} \mu_i}$
$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$
$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$
$\bar{\mathbf{m}}_t \leftarrow (1-\mu_t)\hat{\mathbf{g}}_t + \mu_{t+1}\hat{\mathbf{m}}_t$
$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\bar{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t}+\varepsilon}$

Figure 2: (Adapted from [4])

### 3.7 Residual learning

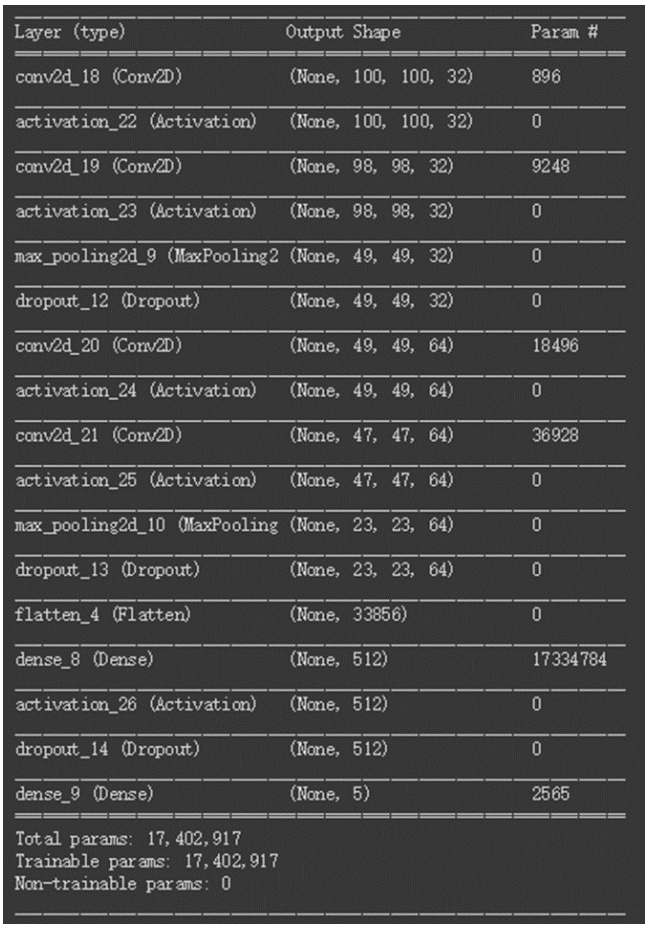
In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some (n+x)th layer. It has proved that training this form of networks is easier than training simple deep convolutional neural networks and the problem of degrading accuracy is resolved [6].

## 4 Experiment and Results

In this section, we will use pre-trained model which called ResNet and our self-constructed model.

### 4.1 Self-constructed model

The architecture will be introduced as follow:



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_18 (Conv2D) | (None, 100, 100, 32) | 896 |
| activation_22 (Activation) | (None, 100, 100, 32) | 0 |
| conv2d_19 (Conv2D) | (None, 98, 98, 32) | 9248 |
| activation_23 (Activation) | (None, 98, 98, 32) | 0 |
| max_pooling2d_9 (MaxPooling2 | (None, 49, 49, 32) | 0 |
| dropout_12 (Dropout) | (None, 49, 49, 32) | 0 |
| conv2d_20 (Conv2D) | (None, 49, 49, 64) | 18496 |
| activation_24 (Activation) | (None, 49, 49, 64) | 0 |
| conv2d_21 (Conv2D) | (None, 47, 47, 64) | 36928 |
| activation_25 (Activation) | (None, 47, 47, 64) | 0 |
| max_pooling2d_10 (MaxPooling | (None, 23, 23, 64) | 0 |
| dropout_13 (Dropout) | (None, 23, 23, 64) | 0 |
| flatten_4 (Flatten) | (None, 33856) | 0 |
| dense_8 (Dense) | (None, 512) | 17334784 |
| activation_26 (Activation) | (None, 512) | 0 |
| dropout_14 (Dropout) | (None, 512) | 0 |
| dense_9 (Dense) | (None, 5) | 2565 |

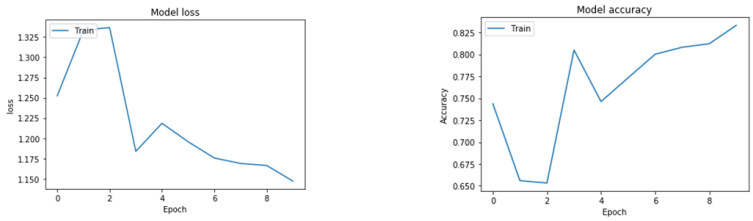
Total params: 17,402,917
Trainable params: 17,402,917
Non-trainable params: 0

Figure 3:

Modules: Basic model + binary crossentropy loss + Adam.
Input size = (224,224,3). Number of implementations: 5, Epoch = 10

| Training accuracy (AVG) | Validation accuracy (AVG) | Running time(AVG) | Last Loss |
|---|---|---|---|
| 83% | 5.1% | 29 Min | 1.14 |

The loss curve and accuracy curve will be given as follow:




(a) Plot 1: Loss Curve

(b) Plot 2: Accuracy Curve

Figure 4:

## 4.2 Pre-trained Model

ResNet-50 is a convolutional neural network that is trained on more than a million images from the ImageNet database [6]. The network is 50 layers deep and can classify images into 1000 object categories. The architecture is showed below:

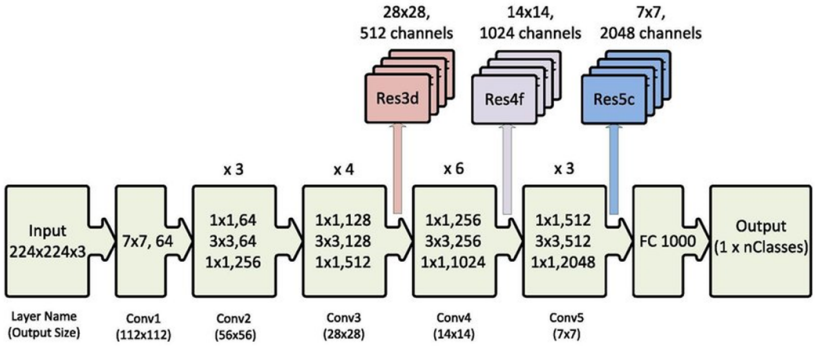



Figure 5: (Adapted from Research Gate)

In our implementation process, we need to freeze the ResNet model. This is simply because ResNet is pre-trained model for ImageNet contest. However, our datasets are different from ImageNet datasets. Without freezing all layers from ResNet model, there will lead to a very low accuracy. Here is our self-constructed model after ResNet:

| Layer (type) | Output Shape | Param # |
|---|---|---|
| resnet50 (Model) | (None, 7, 7, 2048) | 23587712 |
| global_average_pooling2d_6 ( | (None, 2048) | 0 |
| dense_25 (Dense) | (None, 512) | 1049088 |
| batch_normalization_v1_12 (B | (None, 512) | 2048 |
| dense_26 (Dense) | (None, 128) | 65664 |
| batch_normalization_v1_13 (B | (None, 128) | 512 |
| dense_27 (Dense) | (None, 20) | 2580 |

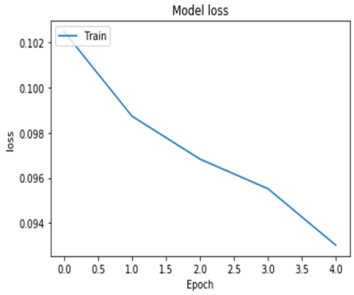
Total params: 24,707,604
Trainable params: 1,118,612
Non-trainable params: 23,588,992

Figure 6: Pre-trained model + Base Model

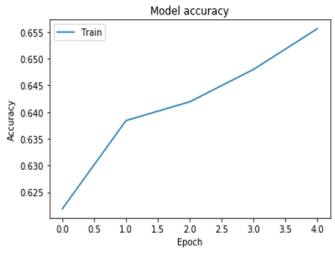
Modules: ResNet50 Model + Base model + Binary Crossentropy + Nadam
Input Size = (224,224,3), Number of implementations : 5, Epoch = 10

| Training accuracy (AVG) | Validation accuracy (AVG) | Running time(AVG) | Last Loss |
|---|---|---|---|
| 65% | 70.03% | 13 Min | 0.0931 |

The loss curve and accuracy curve will be given as follow:




(a) Plot 3: Loss Curve




(b) Plot 4: Accuracy Curve

Figure 7:

# 5 Conclusions and Discussion

During our experiment process, the pre-trained model outperforms self-constructed model. However, the accuracy still stays on 70%. There are several reasons that we cannot improve testing accuracy a lot. First, the label is not equally distributed. The label 0, 5 and 15 are relatively small compared to others. Second, the datasets are relatively small for image recognition. In conclusion, the overall testing accuracy for multi-label image recognition is 70%.

During our work on the multi-class classification, we experienced the process to build Convolutional Neural Network. We experienced the procedure of learning, planning and designing a classification project. The most important gain from this project is that we learned the details and principles of how to process a single image with multiple labels. The influences of learning modules and writing codes on us are mutual. Meanwhile, the implementation process gives us better understanding of the whole task.

There is a lot of space for improvement, however, in terms of computational cost and final accuracy. The current time spent is acceptable, if we can use better GPU like GTX 2080Ti, it will save lots of time. In addition to this, we can try different pre-trained models like VGG16. After we finished, we learned huge on the insight of how neural network works. This is a positive experience on our studying and future career.

# References


[1] Bui, V., & Chang, L. C. (2016). Deep learning architectures for hard character classification. In Proceedings on the International Conference on Artificial Intelligence (ICAI) (p. 108). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[2] Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In Advances in neural information processing systems (pp. 2843-2851).

[3] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[4] Dozat, T. (2016). Incorporating nesterov momentum into adam.

[5] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). Cambridge: MIT press.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2016). Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770778). IEEE. https://doi.org/10.1109/CVPR.2016.90

[7] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., & Karpathy, A. (2014). Large-Scale Video Classification with Convolutional Neural Networks. In IEEE Conference on Computer Vision and Pattern Recognition. Proceedings (pp. 17251732). https://doi.org/10.1109/CVPR.2014.223

[8] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[9] Krizhevsky, A., Sutskever, I., & Hinton, G. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 8490. https://doi.org/10.1145/3065386

[10] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 22782324. https://doi.org/10.1109/5.726791

[11] Lee, C. Y., Gallagher, P. W., & Tu, Z. (2016, May). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In Artificial Intelligence and Statistics (pp. 464-472).

[12] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

[13] Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNNSVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), 1318-1325.

[14] Romanuke, V. (2017). Appropriate number and allocation of ReLUs in convolutional neural networks. Naukovi Visti NTUU KPI, (1), 69-78.

[15] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[16] Russell, B., Torralba, A., Murphy, K., & Freeman, W. (2008). LabelMe: A Database and Web-Based Tool for Image Annotation. International Journal of Computer Vision, 77(1), 157173. https://doi.org/10.1007/s11263-007-0090-8

[17] Xu, C. (2019). Deep Learning (COMP5329), lecture 3, week 5: Optimization for Training Deep Models [Lecture PowerPoint slides]

# A How to run the codes

1. Open Google Colaboratory
2. Upload A2.ipynb to Google Colaboratory
3. Click Run All

Notice that you should log in to your personal Gmail account to get varification code in first cell.

# B Predicted outputs for test data

The predicted outputs for test data are contained in file Predicted_labels.txt, which can be found in folder /Code/Output.

The file should contain numerical labels (ranging from 0 to 19) for 15516 images. Using Python to read the file and print its shape, type and first 100 predicted labels should display here

[2, 8, 6, 13, 6, 16, 3, 8, 3, 8, 11, 11, 9, 3, 11, 17, 17, 9, 8, 11, 1, 12, 3, 2, 13, 11, 9, 18, 19, 19, 17, 17, 7, 14, 19, 9, 7, 9, 11, 11, 9, 12, 17, 13, 7, 11, 16, 9, 6, 19, 8, 18, 18, 19, 11, 16, 8, 9, 6, 19, 11, 19, 11, 8, 16, 11, 0, 9, 0, 11, 2, 8, 11, 1, 19, 19, 9, 17, 9, 3, 12, 18, 8, 11, 11, 2, 1, 4, 9, 9, 9, 11. 19, 6, 12, 0, 6, 1, 3, 9]

# C predicted label model

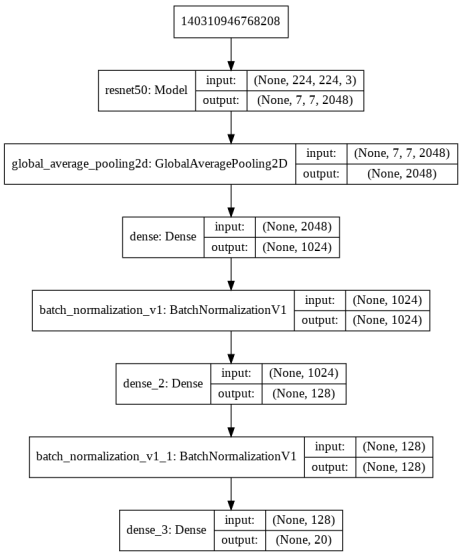



Figure 8: predicted_label model

# D Hardware and Software Specifications

All testing were carried out using Google Colaboratory with GPU accleration.