A CLASSIFICATION NERUAL NETWORK WITHOUT USING DEEP LEARNING PACKAGES

Weilun Wang, Sun’ao Liu
University of Science and Technology of China

Introduction

In this project, we implement a neural network to predict the class labels of a given image without using any deep learning packages.

For each testing image, the classifier will output the top three class labels with the highest confidence scores. If the true class label is one of the top three labels, we will say the classifier has correctly predicted the class label of the input image; otherwise, the classifier made a mistake.

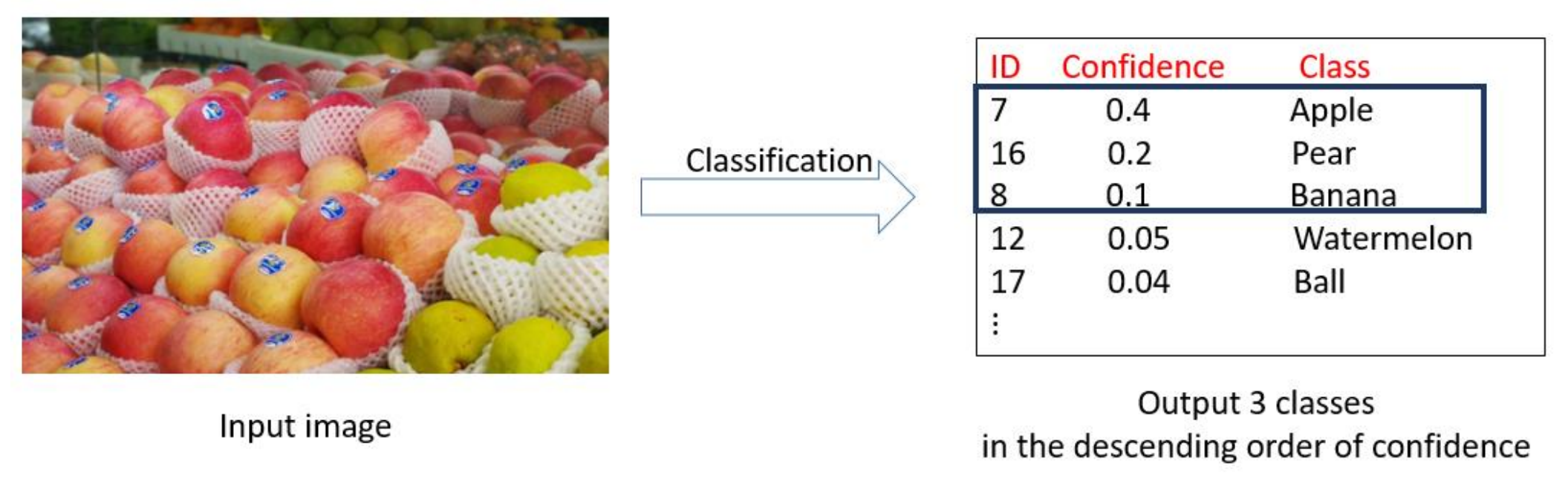

Figure 1. Example of the task

Dataset

This dataset contains 10,000 images in total, which are divided into 20 classes (each class has 500 images). The class labels are as follows.

Table 1: The class labels

Label index	1	2	3	4	5
Description	Goldfish	Frog	Koala	Jellyfish	Penguin
Label index	6	7	8	9	10
Description	Dog	Yak	House	Bucket	Instrument
Label index	11	12	13	14	15
Description	Nail	Fence	Cauliflower	Bell pepper	Mushroom
Label index	16	17	18	19	20
Description	Orange	Lemon	Banana	Coffee	Beach

In the actual training, we searched the Internet for 1000 64*64 RGB images as training materials for the unknown class.

Method

- **VGG-like model structure**

Our VGG-like model structure is shown in figure 3, with more details in next section.

- **Optimizer: Adam**

We use Adam algorithm [1] for our optimization. Adam is an adaptive learning rate optimization algorithm and is presented as follow. The name “Adam” derives from the phrase “adaptive moments”.

- **Initialization: Xavier**

In our program, we use the Xavier Initialization [2] :

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}]$$

where $n_j$and n_{j+1} denote the input and output dimension for the j^{th} layer, respectively.

- **Accelerate convolution: im2col**

The implement of naive convolution is very inefficient. In order to accelerate this process, we used the method to change convolution into matrix product. The key function is im2col, which can change the input image into proper matrix form. A simple example is shown in figure 2.

- **Batch Normalization**

We also used Batch Normalization [3] to improve the performance of our network. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer , in some cases eliminating the need for Dropout.

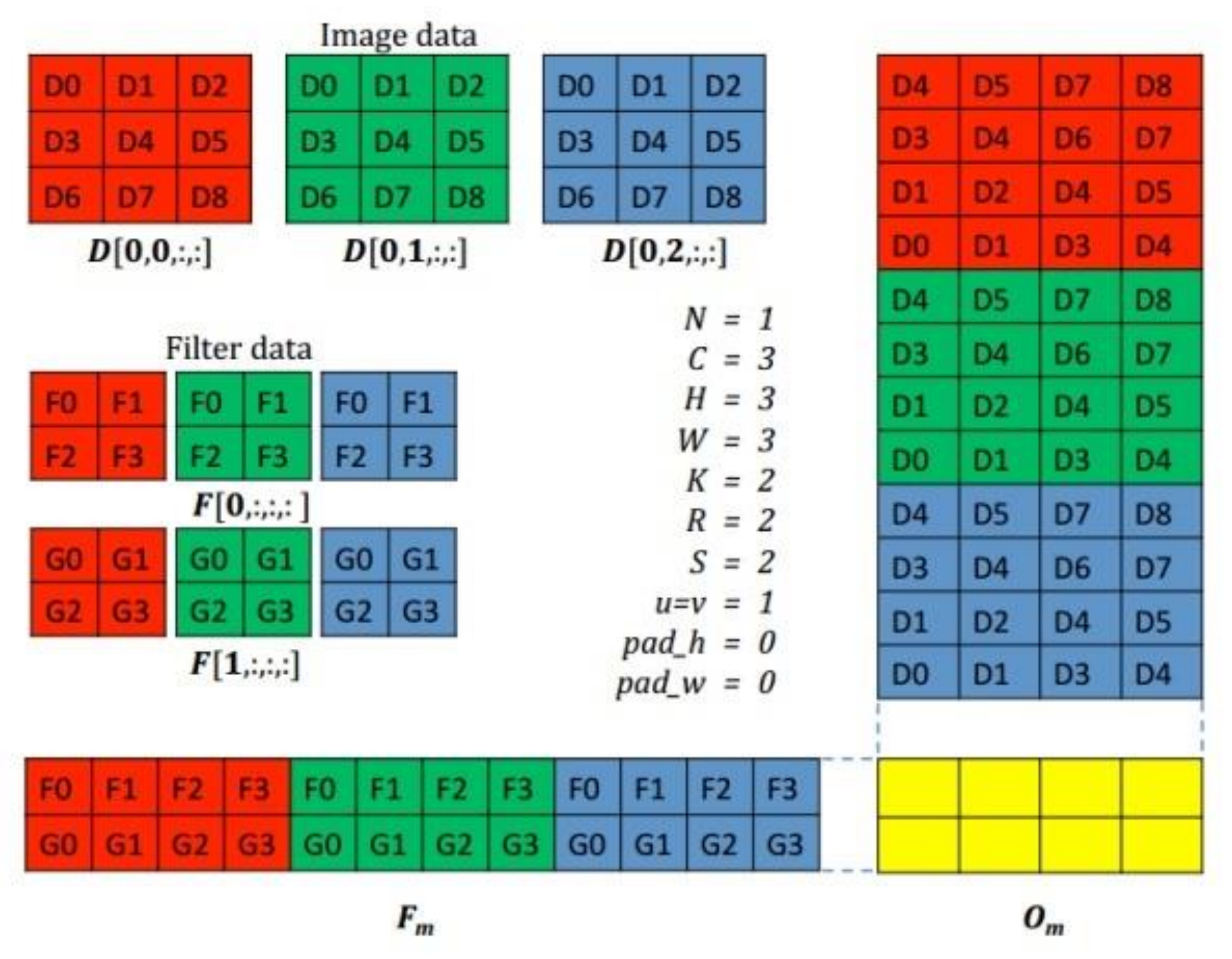

Figure 2. Example of im2col

Results

We choose three different structures and test their performance on this problem separately. We have chosen some small models because small models are more likely to converge than large models in the case of limited computing resources, which may result in better performance.

We trained 15 epochs on the training set with three different models and tested the accuracy of the model on the validation set after each epoch. We selected the best performance of 15 epochs as the best performance under one structure. Model structures and their results are shown as follows.

Table 2: Model structures and accuracy

Model	4-layers model	6-layers model	VGG-like model

Structure	Conv_relu(32,3,3)
	Max pooling(2,2)
	Fc(100)
	Fc(21)
		Conv_relu_bn(32,5,5)
		Max pooling(2,2)
		Conv_relu(64,3,3)
		Conv_relu_bn(64,3,3)
		Max pooling(2,2)
		Fc(128)
		Fc(21)
			Conv_relu(32,3,3)
			Max pooling(2,2)
			Conv_relu_bn(64,3,3)
			Max pooling(2,2)
			Conv_relu(128,3,3)
			Conv_relu_bn(128,3,3)
			Max pooling(2,2)
			Conv_relu(256,3,3)
			Conv_relu_bn(256,3,3)
			Max pooling(2,2)
			Fc(256)
			Fc(256)
			Fc(21)

Accuracy	0.674	0.702	**0.750**

Based on the performance on the validation set, we finally chose VGG-like model as our final model and got an accuracy of **0.662** on test set. Its full structure is shown in figure 3.

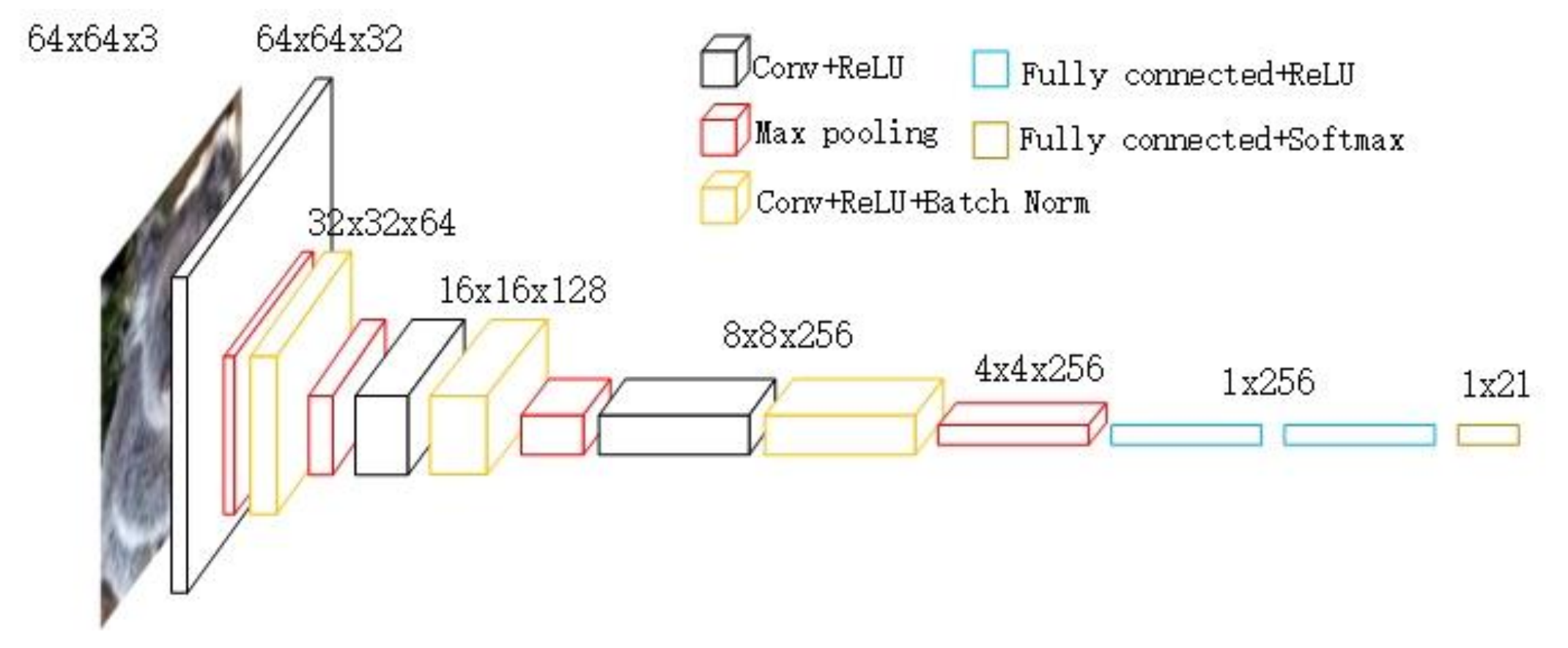

Figure 3. VGG-like model structure

The loss and accuracy history of the model during training are as follows.

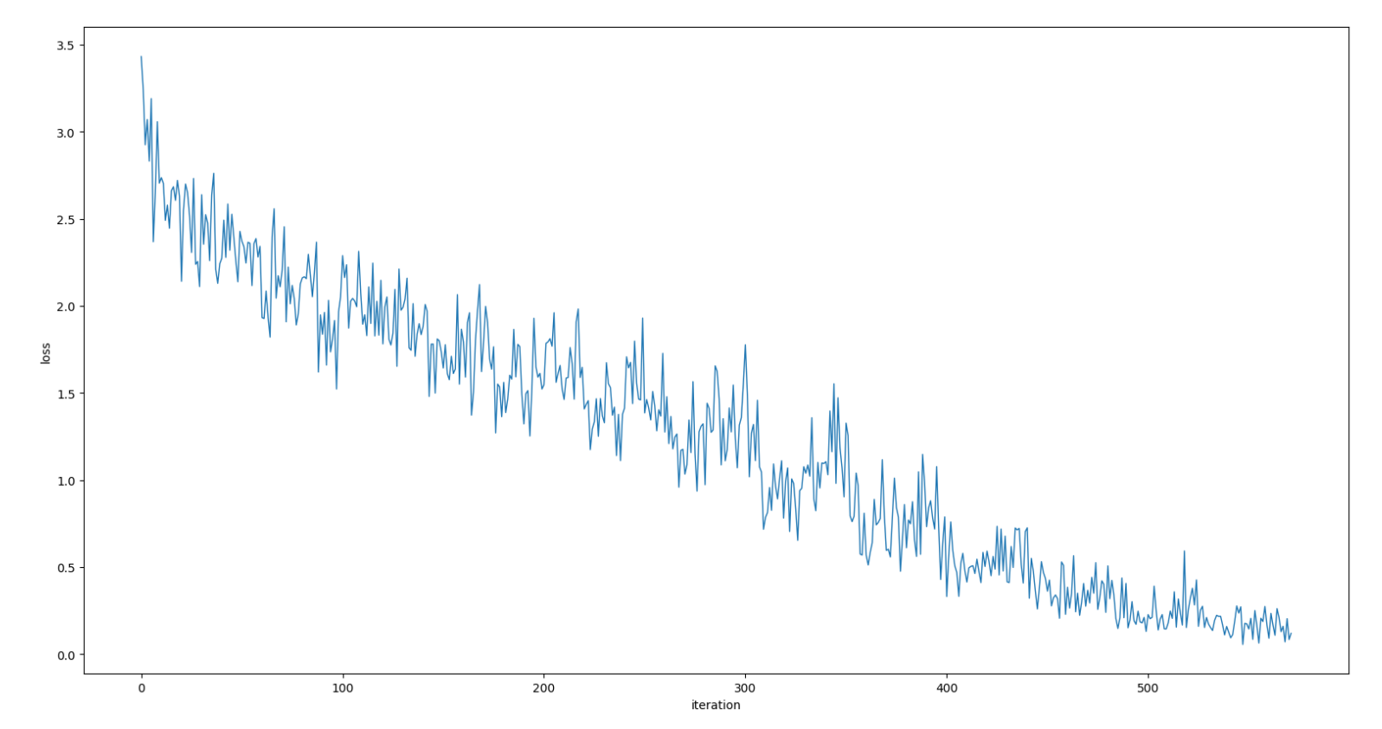

Figure 4. Loss

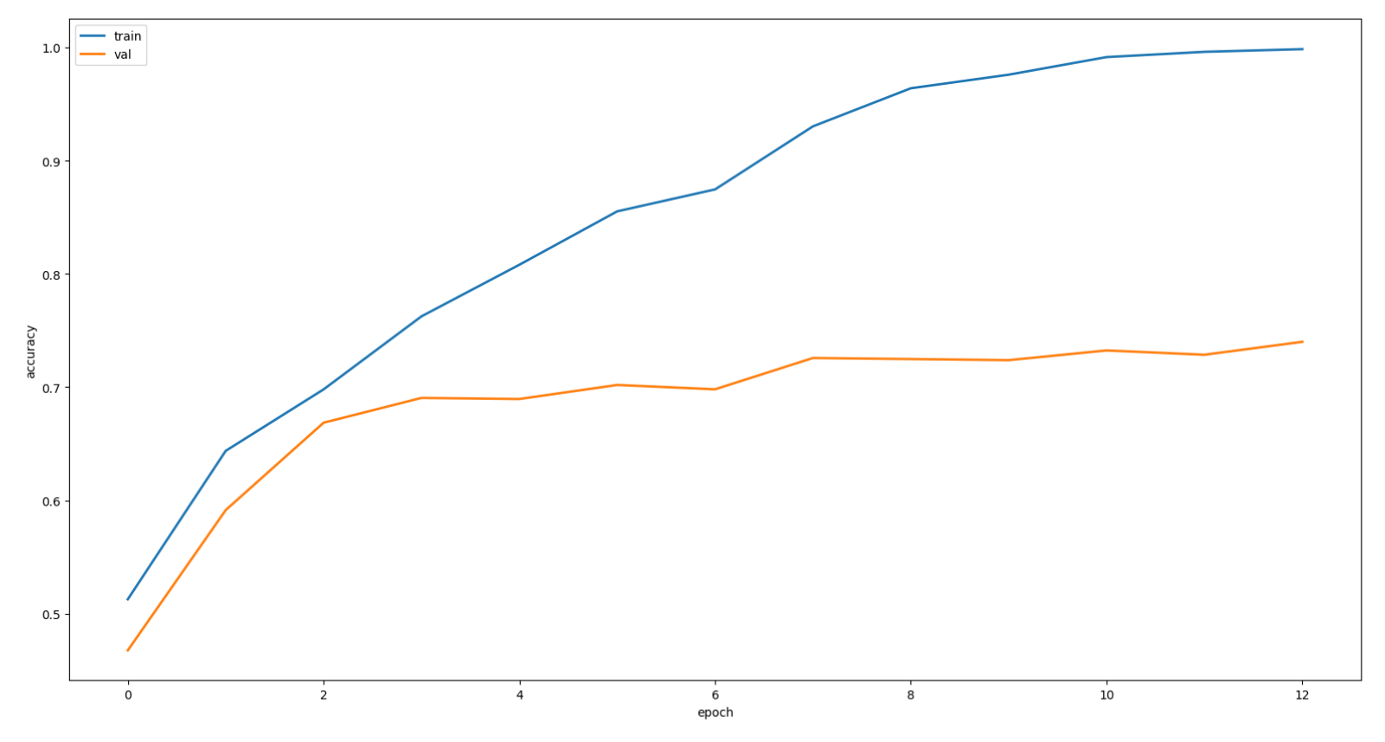

Figure 5. Accuracy

Reference

[1]. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980, 2014.
[2]. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, in Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249-256, 2010.
[3]. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, arXiv preprint arXiv:1502.03167, 2015.

Contact

Weilun Wang PB15061353
wwlustc@mail.ustc.edu.cn
Sun’ao Liu PB15061324
lsa1997@mail.ustc.edu.cn