ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

Dishes Recognition System Based on Deep Learning

Zhenlin He[1,a*], Zixuan Zhang[2,b], Guodong Feng[3,c], Ziyi Yan[4,d], Lubin Yi[5,e], Zihan Yi[6,f]

[1]South China University Of Technology, Guangzhou, China
[2]Hungarian-Chinese Bilingual Primary School and High School, Budapest, Hungary
[3]RDFZ Chaoyang Branch School, Beijing, China
[4]International Department of Beijing No.35 High School, Beijing, China
[5]Henan University of Economics and Law, Zhengzhou, China
[6]Nanjing University of the Arts, Nanjing, China
[a]1362069574@qq.com, [b]zhangzixuan04@gmail.com, [c]fengguodong@rdfzcygj.cn, [d]3500972686@qq.com, [e]951159341@qq.com, [f]1076244207@qq.com
**Corresponding author: 1362069574@qq.com*
These authors contributed equally to this work.

Abstract: *In recent years, food image recognition has emerged as a research hotspot in the field of computer vision. This paper proposes a dish recognition system based on image recognition and deep learning image processing and recognition, which can replace manual dish recognition. Because the food image is complex and dynamic, the system employs a Convolutional Neural Network (CNN). The dish detection algorithm has been completed using CNN. Multiple sets of comparative experiments were designed for the data before and after the improvement, and the ionfluence of the improvement points on the test data results was analyzed.*

Keywords: *Deep Learning; CNN; Image Recognition*

1. Introduction

With the rapid expansion of the mobile Internet's scale, sensing technology and artificial intelligence have gradually matured and made significant progress. In the last two years, the payment method in the cafeterias represented by college cafeterias has remained inefficient. This type of canteen has a high volume of people in a short period of time, is noisy and chaotic, does not print receipts, and has management difficulties. It is prone to long checkout lines and erroneous checkout links[1].

At the same time, a dinner plate is frequently required at the checkout. The scene of checking out with a card or mobile phone in one hand, the sound of the dinner plate falling on the floor one after the other during the peak period of the meal. In China, several "unmanned supermarkets" have emerged, improving people's shopping efficiency and eliminating long lines at checkout. With rising labor costs and a trend toward faster-paced living, it is clear that the service models of "self-service" and "machine instead of labor" will replace traditional service methods on a large scale.

2. Related Work

2.1. Principles of Convolutional Neural Networks

Convolutional Neural Network (CNN) is a feed-forward neural network. Artificial neurons can respond to surrounding units and can perform large-scale image processing[2]. Convolutional neural networks include convolutional layers and pooling layers. Convolutional neural networks are MLPs (multilayer perceptrons) inspired by biological thinking. They have different category levels, and the working methods and functions of each layer are also different.

2.2. CNN Network Hierarchy

The CNN network has a total of 5 hierarchies as follows:

•Input layer

ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

- Convolutional layer
- Activation layer
- Pooling layer
- Fully connected FC layer

2.2.1. Input Layer

Like traditional neural network/machine learning, the model requires input for preprocessing operations. The three common preprocessing methods are:

- Remove the mean
- Normalization
- PCA/SVD dimensionality reduction, etc.

2.2.2. Convolutional Layer

Local perception: In the process of the human brain recognizing a picture, it does not recognize the entire picture at once, but first perceives each feature in the picture locally, and then performs comprehensive operations on the parts at a higher level to obtain global information[3].

Through the local perception characteristics, the calculation parameters of the model are greatly reduced. But there are still many parameters just like this. This has the weight sharing mechanism: for each layer, the weights corresponding to all neurons should be equal, for example, the parameter vector of a neuron is [w1,w2,...,w100], Then other neurons in the same layer are also [w1,w2,...,w100], which is weight sharing.

Why do we need weight sharing? The connection parameters of neurons in the same layer are only related to the feature extraction and have nothing to do with the specific location, so it can be guaranteed that the connections of all locations in the same layer are weight-shared. For example, the first layer of the hidden layer is generally used for edge detection, the second layer is to combine the edge curves learned in the first layer to obtain some features, such as angle, line shape, etc.; the third layer will learn more complex features, such as: “Eyes, eyebrows,” etc. For the same layer, they extract features in the same way. The neurons in the third layer are all used to extract the features of the "eye", so the parameters that need to be calculated are the same.

2.2.3. Incentive Layer

The so-called excitation is a nonlinear mapping of the output result of the convolutional layer.

If the activation function is not used (in fact, the activation function is equivalent to f(x)=x), in this case, the output of each layer is the linear function of the input of the previous layer. It is easy to conclude that no matter how many neural network layers there are, the output is a linear combination of inputs. The effect is the same as without hidden layers. This is the most primitive perception[4].

Commonly used excitation functions are:

- Sigmoid function
- Tanh function
- ReLU (Rectified Linear Unit)
- Leaky ReLU
- ELU (Exponential Linear Unit)
- Maxout

Suggestion from the incentive layer: ReLU first, because the iteration speed is fast, but the effect may not increase. If ReLU fails, consider using Leaky ReLU or Maxout. In this case, the general situation can be solved. The Tanh function has better results in text and audio processing.

2.2.4. Pooling Layer

Pooling: also known as under-sampling or down-sampling. It is mainly used for feature dimensionality reduction, compressing the number of data and parameters, reducing overfitting, and improving the fault tolerance of the model.

ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

There are:

- Max Pooling: Maximum pooling
- Average Pooling: average pooling

Through the pooling layer, the original feature map of 44 can be compressed to 22, thereby reducing the feature dimension.

2.2.5. Output Layer

After several previous convolutions + excitation + pooling, finally came to the output layer, the model will learn a high-quality feature image fully connected layer. In fact, before the fully connected layer, if the number of neurons is too large and the learning ability is strong, overfitting may occur. Therefore, the dropout operation can be introduced to randomly delete some neurons in the neural network to solve this problem. You can also perform operations such as local normalization (LRN) and data enhancement to increase robustness, which will not be introduced here.

When it comes to the fully connected layer, it can be understood as a simple multi-class neural network (such as BP neural network), and the final output is obtained through the softmax function. The entire model is trained.

2.3. Fully-Connected

A fully connected neural network (DNN) is the simplest neural network, it has the most network parameters and the largest amount of calculation[5].

The structure of DNN is not fixed. A general neural network includes an input layer, a hidden layer, and an output layer. A DNN structure has only one input layer, one output layer, and there are hidden layers between the input layer and the output layer. Each layer of the neural network has several neurons. The neurons between the layers are connected, the neurons in the layer are not connected, and the neurons of the next layer are connected to all the neurons of the previous layer.

A neural network with more hidden layers is called a deep neural network (the number of network layers in DNN does not include the input layer). The expressive power of a deep neural network is stronger than that of a shallow network. A neural network with only one hidden layer can fit Any function, but it requires many, many neurons.

- Advantages: Since DNN can fit almost any function, the nonlinear fitting ability of DNN is very strong. Often deep and narrow networks need to be more resource-efficient.
- Disadvantages: DNN is not easy to train, requires a lot of data, and a lot of skills to train a deep network.

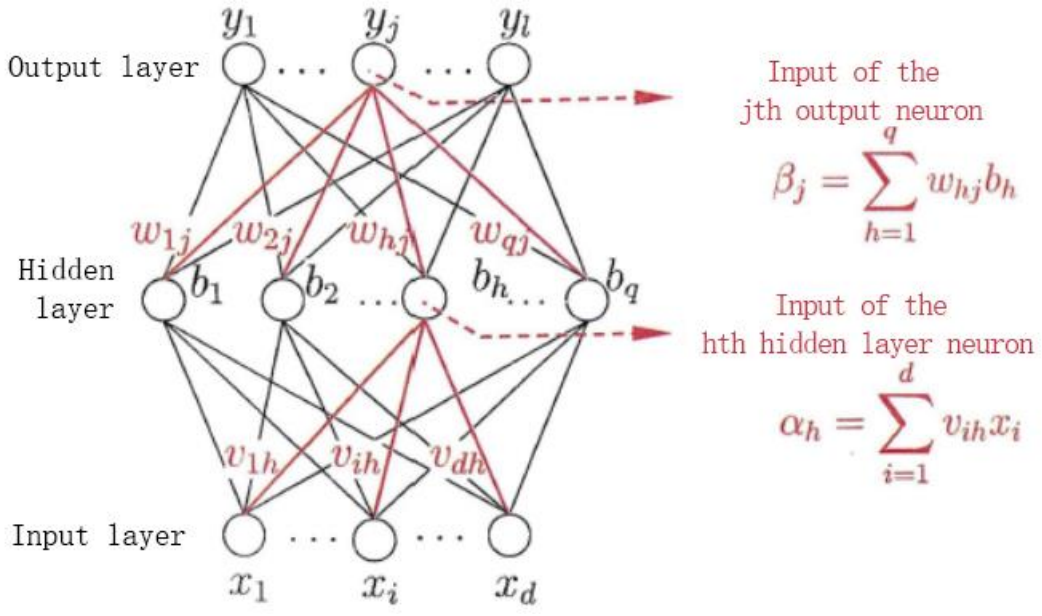

Figure 1: The structure of neural network DNN

ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

2.4. Sensor

DNN can also be called multi-layer perceptron (MLP). The network structure of DNN is too complex and the number of neurons is too large. To facilitate the explanation, we design the simplest DNN network structure-perceptron.

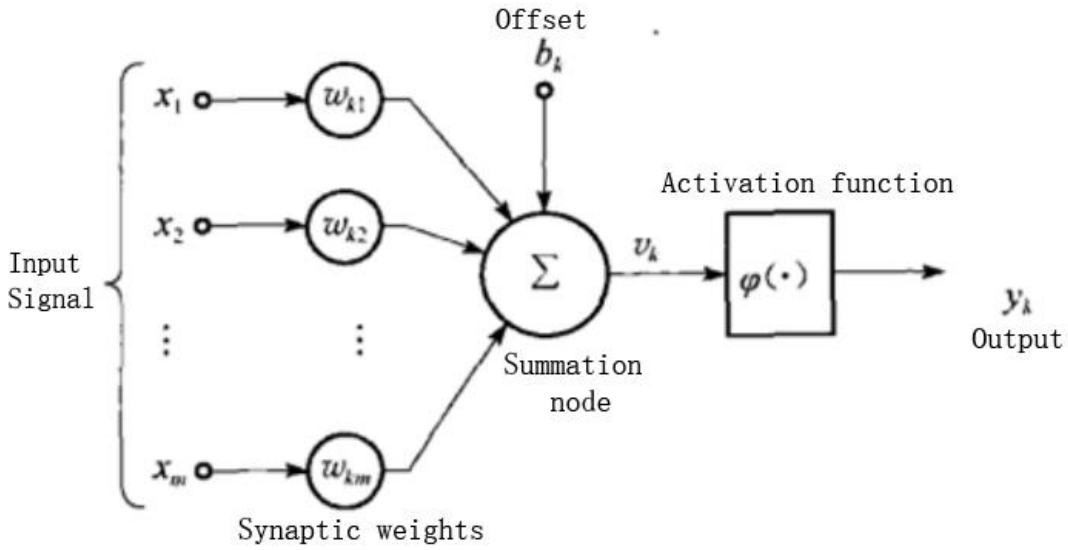

Figure 2: The DNN sensor structure

The neuron in DNN consists of five parts:

- Input: A perceptron can receive multiple inputs (x1,x2,...,xn|xi∈R)(x1,x2,...,xn|xi∈R)
- Weight: each input weights wi∈Rwi∈R
- Bias term: b∈Rb∈R, which is w0w0 in the figure above
- Activation function: It is also called a nonlinear unit. There are many activation functions for neural networks. I have a blog dedicated to the activation function.
- Output: y=f(w∗ x+b)y=f(w∗ x+b)

3. Methodologies

3.1. Data Collection&Data Augmentation

Comparing to some classical data set including ILSVRC or CIFAR-10/100, we only need to consider a relatively small set of 6 types of cuisines. Basically as the data set for a neural network enlarged, the test performance will be more accurate. Here, we create a small data set that contains 54000 pictures to verify that our algorithm works as expected, which is a good start point to test the codes and make an efficient classifier for dishes. By using web spider and paragraphing on our own, we derive 300 pictures for every 6 cuisines and assign a value as their labels( ). Then, we rotate each picture to a certain angle to gain 9000 pictures per dish. A Gaussian filter is also employed to make the pictures smooth. To avoid aliasing, we preserve the RGB channels while resizing the pictures to 32*32 pixels. Last, we transfer the data to numpy.array with a shape (pic_numbers,32,32,3) and divide the data into train set and test set (ratio = 4:1). The label according to each single picture is transfered to the one-hot-encoding format to compare with the result of classifying later on..

3.2. The Architecture

After data collection, we define the model to extract features from those numerical data. Our model contains 6 layers, with 4 convolutional and 2 fully-connected. The first convolutional layer filters the 32*32*3 pictures with 32 kernels of 3*3*3. The other 3 convolutional layers all have the same input and

ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

output depth as 32. Each convolutional layer is combined with LeakyReLU activation function, and the second and fourth layer are followed with the pooling layers. Taking the output of the convolutional layers, the first fully-connected layer has 2048 neurons with 128 output nodes. The output of the last fully-connected layer would be scores for the likelihood of 6 different cuisines.

Briefly describe our model as above, it is plain to see that our model has a relatively single size as a dishes classifier only need to distinguish 6 cuisines. In fact, we have tried pictures ranging from 16*16 to 256*256 and found that there will be a prompt of test accuracy from 16 to 32, with negligible gains thereafter. We also notice that the training time is only 2-3 minutes on the 54000*32*32*3 data set, but the time is about 10 times longer when it comes to the same pictures of128*128. So we employ the current model, which is inexpensive and competitive when put into commercial use, since the kinds of cuisines might change frequently and the data will be enlarged. Along with the small imput size, the convolutional layers also have small-scale stride as 1 pixel and kernels. In this way, depth is increased for better performance.

We also employ some techniques to solve the problem of overfitting which will be introduced below.

3.2.1. Activation Function

A standard model to activate an output of an neuron is the function $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. However, the gradient of this function lies between (0 , 0.25), which indicates the gradient of the loss function has the multiply of these minor gradients by 4 times and becomes negligible to back propagation. The chain rule also implies that the different gradient on each layer will lead to a huge discrepancy on different depth. To solve this problem called vanishing gradient, we tried the ReLU function: $f(x) = \max(0, x)$ and finally adopted the LeakyReLU: $f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$. These functions are one-side saturating, which is better for feature extraction. To simplify, when kernels distinguish features from inputs, a high value represents high level of matching and a low value describes obscure, while a negative value is meaningless. They should be undiscriminating to those mismatching. In spite of this, we still introduce a coordinate to the negative values to prevent dying. The neurons of negative values are part of back propagation and if we set their gradients to 0, the neuron chains containing them will no more able to update.

Using LeakyReLU is fast and accurate. It is 6-7 times faster to use LeakyReLU than tanh(x) unints in our model. Also,when we changed the activation function ReLU, the accuracy will drop by an average 0.5%.

3.2.2. Dropout

When training with a small dataset, overfitting occurs. To prevent overfitting， introducing different classifiers and using the average is a good approach, since they will counterbalance the errors to some degree. Dropout is a method to train different models while less time-consuming than training traditionally. Instead, it hides each neuron in fully-connected layers with equal probability p=0.4 . When a neuron is dropped, it no longer acts in propagation. Dropout only appears at training, and the weights will be divided by (1-p) at test.

This method also reduces co-adaptations of neurons. With dropout, two neurons may not occur simultaneously, and each neuron have to learn more robust feature, which exists at other model with different dropout pattern. All models share weights and update one after another.

3.2.3. Training and Testing

We train our model with a batch size of 64, and the data set will be shuffled after certain times of iteration. To initialize, the weights in each layer obey Gaussian distribution with mean=0 and standard deviation of 0.1. Also, the biases are all of constant 1 in every 6 layers. We add regularization to our loss function to prevent overfitting. To specify, the loss function is:

$$L = \text{loss} + \gamma\{\sum_{i=0}^{1}[\sum(W_{fci} ** 2)/2 + \sum(b_{fci} ** 2)/2)]\} \quad (1)$$

Where loss is the mean of softmax cross entropy between logits and labels, γ is the regularization coordinate, W_{fci} and b_{fci} are weights and biases in the ith fully-connected layer.

Once the loss function is derived, we adapt adamoptimizer to update the weights, which is robust and outperforms other stochastic optimizers. The updating rule is:

ISSN 2616-5775 Vol. 5, Issue 2: 48-53, DOI: 10.25236/AJCIS.2022.050208

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ m_t^* = \frac{m_t}{1-\beta_1^t}, v_t^* = \frac{v_t}{1-\beta_2^t} \\ W_{t+1} = W_t - \frac{\alpha}{\sqrt{v_t^* + \epsilon}} m_t^* \end{cases}, (2)$$

Where α is the stepsize, β_1 and β_2 are exponential decay rates, ϵ is $1e-8$, and $gt = \nabla \theta ft(\theta t-1)$. To compare several optimizers, we record the accuracy and iteration steps in fig1.

Table 1: This caption has one line so it is centered

Optimizer	Accuracy	Iteration steps
Stochastic Gradient Descend	79.20%	4070
Momentum	90.90%	1850
NAG	94.40%	1830
Adam	95.20%	1480
AdaDelta	90.50%	2570

Where to stop training is a crucial question. Basically, we set the iteration step as 2000 to converge. However, compare to the 100% accurate performance on training data, the accuracy of test is considered as weak--less than 85%. By reducing the iteration step to 1400, the model has approximately 94% accuracy of test data but with great fluctuation on both train and test data, since the steps to converge is different at each time. So we use the average accuracy of last 10 batches and set the threshold to 98%.

4. Conclusion

We have introduced a dishes recognition system based on deep learning. Our system aimed towards classifying problems with relatively small datasets, combines 4 convolutional and 2 fully-connected layers. We also employed methods of data augmentation, one-side saturating activation layer, and a drop pattern to prevent overfitting. Finally, we compared different optimizers to maximize the accuracy to 95.2%, which demonstrates our system as a highly adaptive one.

References

[1] Yu Zheng, Qiuyu Chen, Jianping Fan, Xinbo Gao. (2020) Hierarchical convolutional neural network via hierarchical cluster validity based visual tree learning, Neurocomputing, 409, 408-419.
[2] Yuanpan Zheng,Guangyang Li,Ye Li. (2019) A review of research on the application of deep learning in image recognition[J]. Computer Engineering and Applications,55(12):20-36.
[3] Wei Li.(2014) Research and application of deep learning in image recognition [D]. Wuhan University of Technology.
[4] W.L. Ding,Y. Li,Y. Chen. (2016) Food category recognition algorithm based on region color features[J]. Electronic Science and Technology. (06)
[5] Yao Zhu,Yijiang Liu. Deep convolutional neural network based dish recognition[J]. Journal of Changzhou Information Vocational Technology College,2020,19(04):35-40.