# Convolutional neural network

Peng Tan (pengtan3@vt.edu)

Feb 25st 2021

## 1 Introduction

### 1.1 Background

Convolutional Neural Network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. [1]

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or convolution kernels that in traditional algorithms are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage. [1]

## 2 Experiment

### 2.1 Datasets

In this Experiment, we just use cifar10 and cifar100 as our dataset to train and test our model. Cifar10 and Cifar100 are subsets of the 80 million tiny images dataset.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The CIFAR-100 dataset consists of 60000 32x32 colour images in 100 classes, with 600 images per class. There are 50000 training images and 10000 test images.

### 2.2 Experiment design

We will use keras Sequential Model API to build the whole model, which include Convolution2D layer, BatchNormalization layer, Dropout layer, MaxPool layer, Flatten layer, Dense layer.

First, building an Convolution layer to clarify the shape of Input data and set initial filter number. In this project, The shape of Input data is [32, 32, 3]. After passing convolution layer, the dimension of data will transfer from [32, 32, 3] to [32, 32, 32]. And then build BatchNormailzation layer to enhance the learning of each node.Building dropout layers to avoid model overfitting.

Secondly, building Maxpool layer on the layer. The purpose of Maxpool is to reduce or downsample the dimensionality of the input image.After passing Maxpool layer, the dimension of data will transfer from [32, 32, 32] to [16, 16, 32].

Third, Repeat the above layer building process to obtain several blocks of convolution layer in order to training model and then flatten the output from above layer and connected to full connected layer. In this step, the dimension of data will transfer from [4, 4, 128] to [256,]

Finally, the last layer will be used to classify. So, In this dense layer, the activation function will be softmax. When the data set is applied to cifar10 or cifar100, the number of output nodes will be 10 or 100 respectively.

Finally, When compiling our model, we will use categorical-crossentropy as loss function, SGD or Adam as optimizer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
re_lu (ReLU)                 (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
=================================================================
Total params: 621,258
Trainable params: 620,810
Non-trainable params: 448
_________________________________________________________________
```

Figure 1: The Summary of model layers for cifar10

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
batch_normalization_3 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
dropout_4 (Dropout)          (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_4 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_7 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               524544
_________________________________________________________________
re_lu_1 (ReLU)               (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               25700
=================================================================
Total params: 644,388
Trainable params: 643,940
Non-trainable params: 448
_________________________________________________________________
```

Figure 2: The Summary of model layers for cifar100

# 3 Discussion

## 3.1 Training Performance and Validation Performance

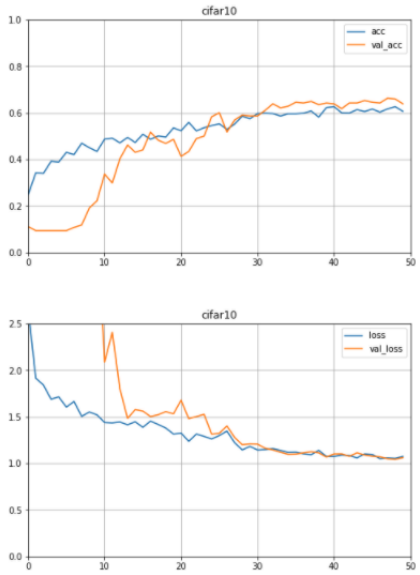


Figure 3: The Performance of CNN in cifar10 dataset

In the Training Performance and Validation Performance in cifar10 dataset(Figure3), we can find that the accuracy increased a lot at first 20 epoch and converge after epoch 30.

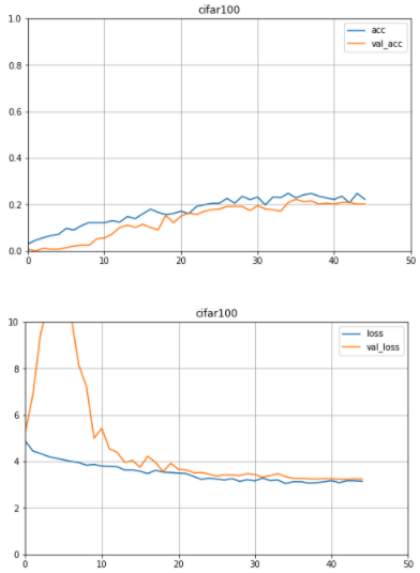


Figure 4: The Performance of CNN in cifar100 dataset

In the Training Performance and Validation Performance in cifar100 dataset(Figure4), we can find that the accuracy only 20 percent after training the model.

### 3.2 Study the performance of Y-networks with Optimize

From Figure5, we can find that the model with 3 layers, [64, 128, 256] filter-number, 0.25 dropout rate, 'adam' optimizer is the best model for cifar10 dataset.

We can find that the model with 3 layers, [32, 64, 128] filter-number, 0.25 dropout rate, 'adam' optimizer is the best model for cifar100 dataset.

| | dataset | layers | filter_number | dropout | optimizer | test_accuracy |
|---|---|---|---|---|---|---|
| 0 | cifar10 | 2 | 32 | 0.25 | adam | 0.6465 |
| 1 | cifar10 | 2 | 64 | 0.25 | adam | 0.5972 |
| 2 | cifar10 | 3 | 64 | 0.25 | adam | 0.6522 |
| 3 | cifar10 | 3 | 32 | 0.35 | adam | 0.6211 |
| 4 | cifar10 | 2 | 32 | 0.25 | SGD | 0.5799 |
| 5 | cifar100 | 2 | 32 | 0.25 | adam | 0.2279 |
| 6 | cifar100 | 3 | 32 | 0.25 | adam | 0.2431 |
| 7 | cifar100 | 3 | 64 | 0.25 | adam | 0.2128 |
| 8 | cifar100 | 3 | 32 | 0.35 | adam | 0.2272 |
| 9 | cifar100 | 3 | 32 | 0.25 | SGD | 0.1880 |

Figure 5: The performance of Y-networks with Optimize

#### 3.2.1 Visualize feature maps

Feature maps are generated by applying Filters to the input image or the feature map output of the prior layers. Feature map visualization will provide insight into the internal representations for specific input for each of the Convolutional layers in the model.

Each convolutional layer will reduce the number of features in a dataset by creating new features from the existing ones. For example, In the first convolutional layer, the original img [32, 32,3] will be transferred to [32, 32, 32]. These new 32 pictures will be able to summarize most of the information contained in the original set of features.

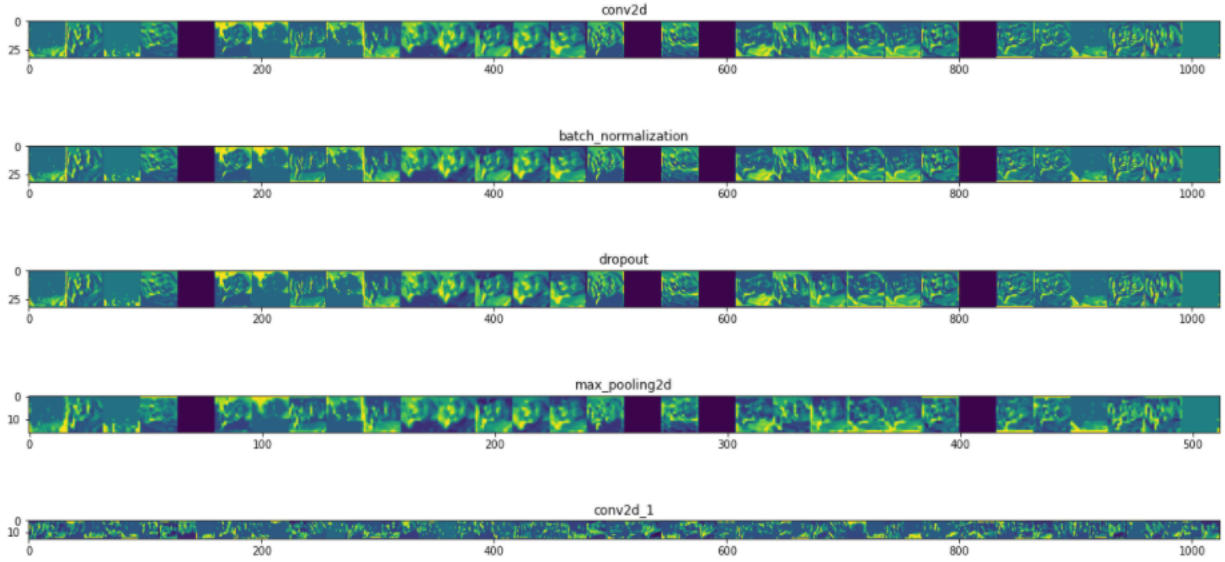


Figure 6: Feature Map

## References

[1] Wikipedia, *Convolutional neural network*, Oct. 13, 2020. https://en.wikipedia.org/wiki/Convolutional_neural_network.