# Neural Network Project

## 1. Neural network architecture

My neural network architecture is inspired by VGG16. Depending on the task requirements, I expand each convolution kernel into an *intermediate block* containing 2 to 3 convolutional layers. You can find the **design diagram** in the **Appendix** in last page.

### Main components:

1. ConvBNRelu

This is a component that contains one convolutional layer (Conv2d), a batch normalization(BatchNorm2d), a ReLU activation function, and an optional Dropout(dropout_rate=0.3 or 0.4). In the architecture, all convolution kernel sizes are 3*3.

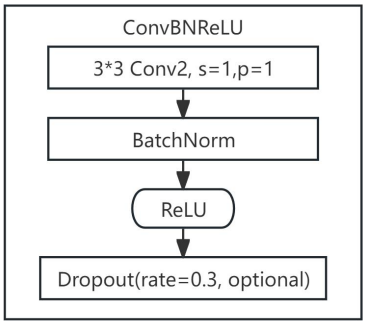



**Figure 1 ConvBNRReLU Convolutional Layer**

2. IntermediateBlock

This block consists of two to three *ConvBNRelu* layers. It also contains a fully connected layer to calculate the weight proportion of each convolutional layer to the output. This block enhances feature representation by learning a linear combination of multiple convolutional layers.

The output method of each block is consistent with the requirements, that is

$x' = a1C1(x) + a2C2(x) + \ldots + aLCL(x)$, where x' is output and L =2 or 3

In order to compute the vector a, the average value of each channel of x is computed and stored into a c-dimensional vector m. The vector m is the input to a fully connected layer that outputs the vector a.

There are 13 *Intermediate Blocks* in this architecture. The first 4 Blocks contain two ConvBNReLU. The last 9 Blocks contain three ConvBNReLU.

3. OutputBlock

The final output block contains two fully connected layer and a ReLU activation layer, as well as a Dropout layer(dropout_rate=0.5). The output block receives an image x (output of the last intermediate block) and outputs a logits vector o. In order to compute the vector o, the average value of each channel of x is computed. Then through two fully connected layers .

## 2. Training

### 1.HyperParameter:

Learning rate:0.1  Batch size:128  Epochs:60  Momentum:0.9  Weight Decay:1e-6

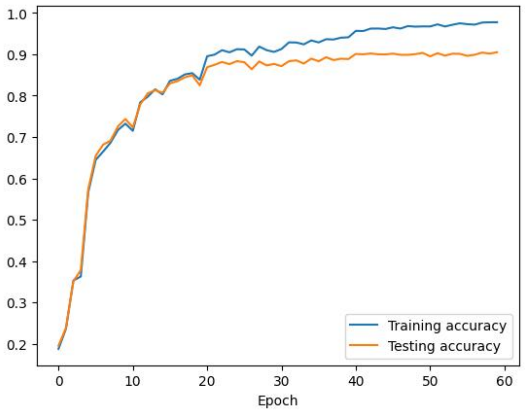
### 2. Technology

1. Optimizer:Using SGD (Stochastic Gradient Descent) as optimization algorithm, combining momentum and Nesterov accelerated gradients

2. Learning Rate Scheduler: The learning rate is dynamically halved every 20 epochs

3. L2 Regularization. 1/2*weight_decay*||w||^2. weight_decay=1e-6

## 3. Training results



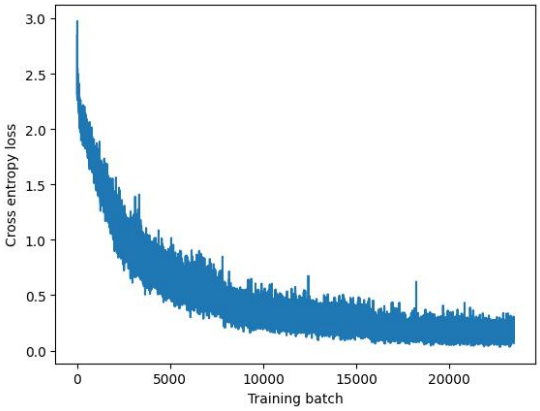




Figure 2 Accuracy-Epoch

Figure 3 Loss-Batch

Test set final accuracy: 0.905 (And if epoch=100, it can be increased to 0.91)

```
Epoch 100/250.
Training accuracy: 0.991100013256073. Testing accuracy: 0.9103000164031982.
```

Figure 4 Accuracy when fully trained

## 4. Data Augmentation

1. 50% probability to randomly flip the image horizontally

2. Randomly rotate the image by an angle within ±15 degrees

# 5. Appendix

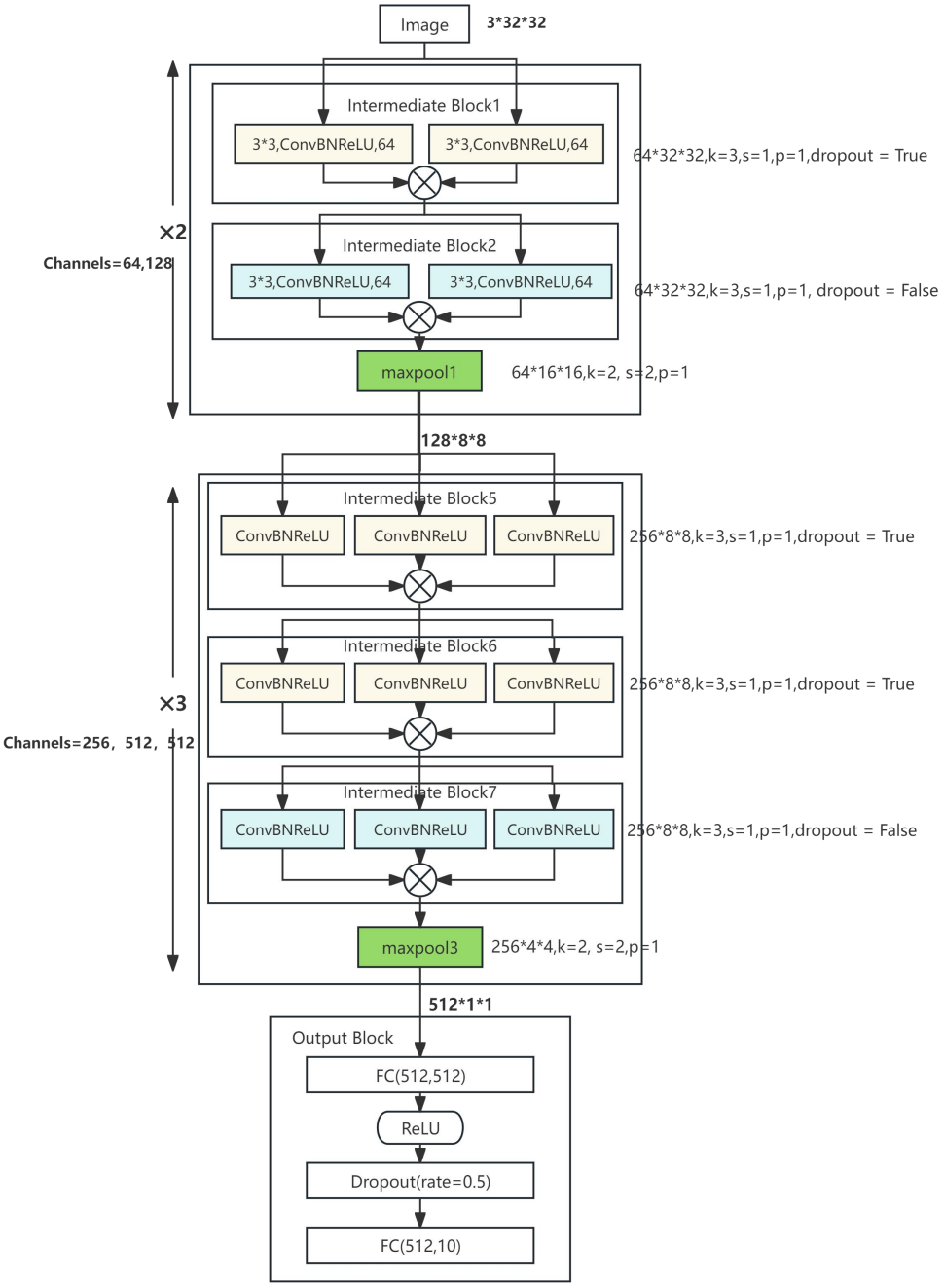



**Figure 5 Neural Network structure**