**DL hw1.** 姓名:許庭恩 學號:111321514

**CIFAIR10 dataset:**

image size(C * H * W): 3*32*32

10 classes:

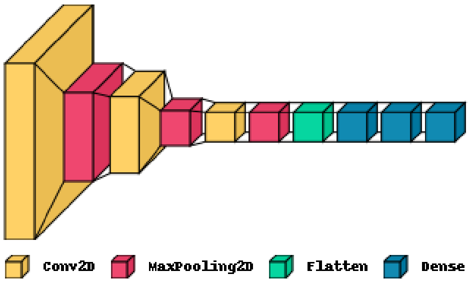
'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'

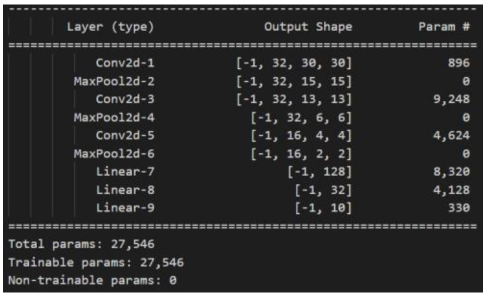
Train:50000pics (split => train: valid = 0.8: 0.2)

Test:10000pics

- **Use Early-Stop to avoid overfitting and keep the best model. (patience=10)**
- **Draw training statistics to know our training trend.**
- **Draw Architecture to know our model appearance.**
- **Use Grad-Cam to see our model focus on where.**

**bs: Batch size、lr: Learning rate、opt: Optimizer、aug: Augmentation、ce: Cross entropy**



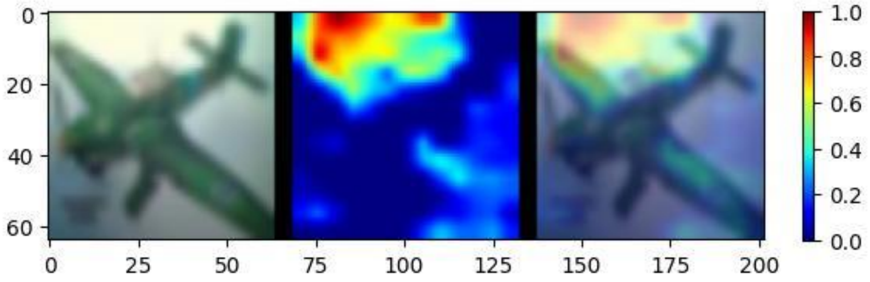
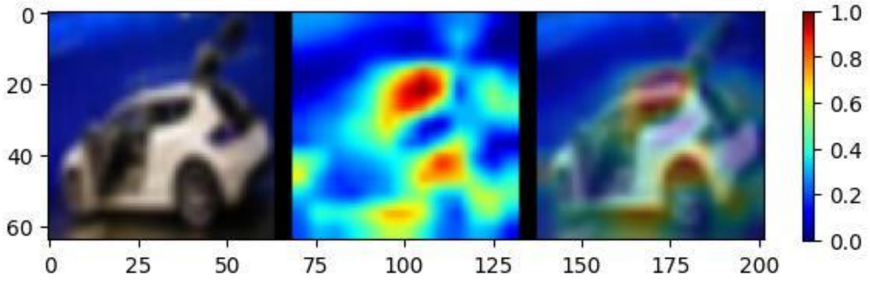
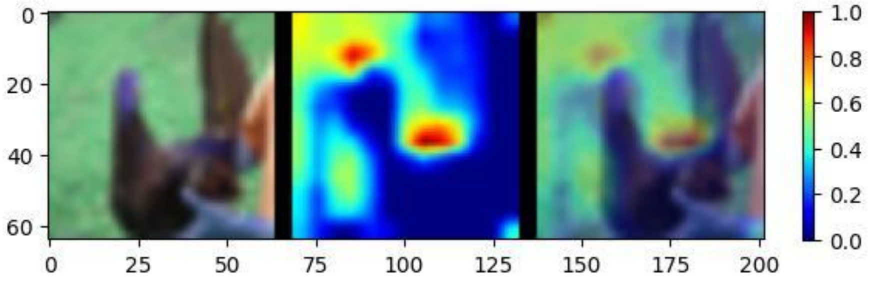
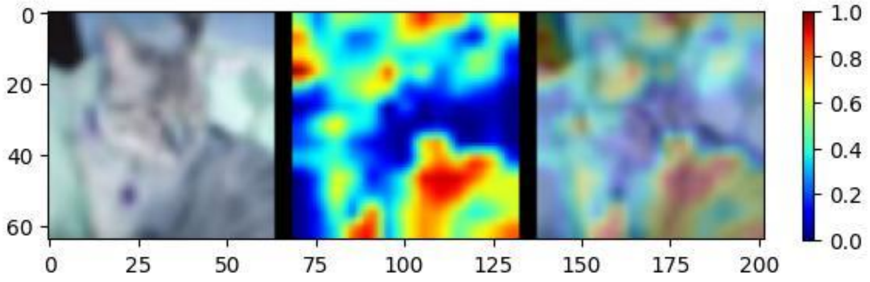
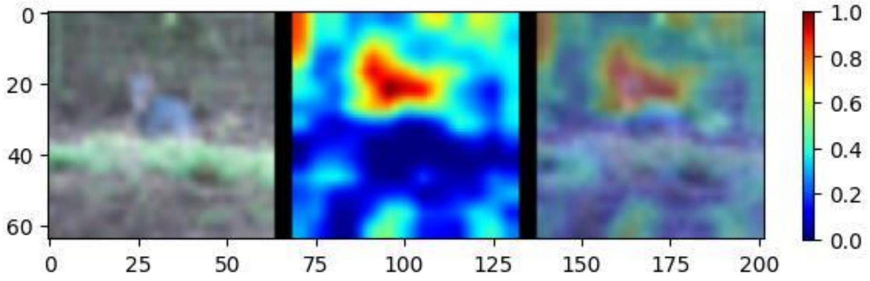


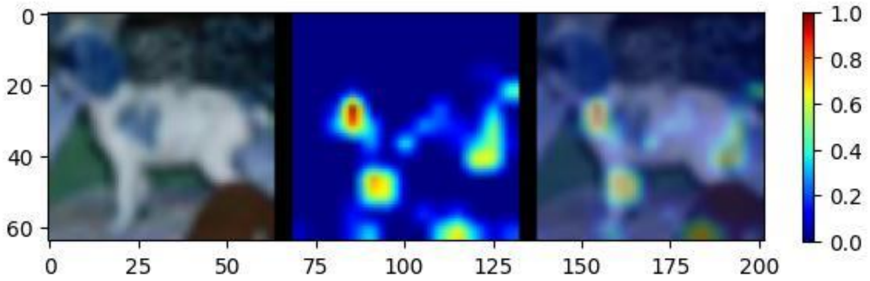
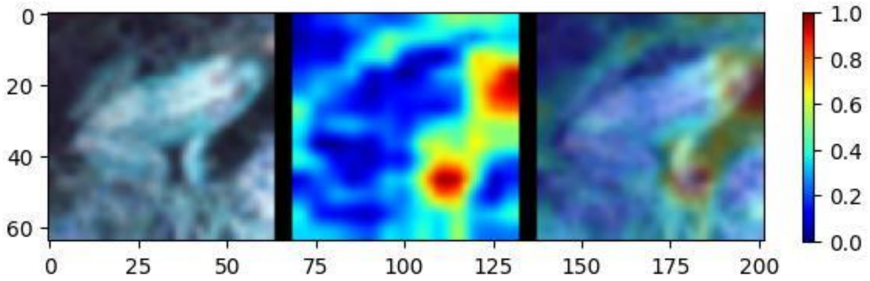
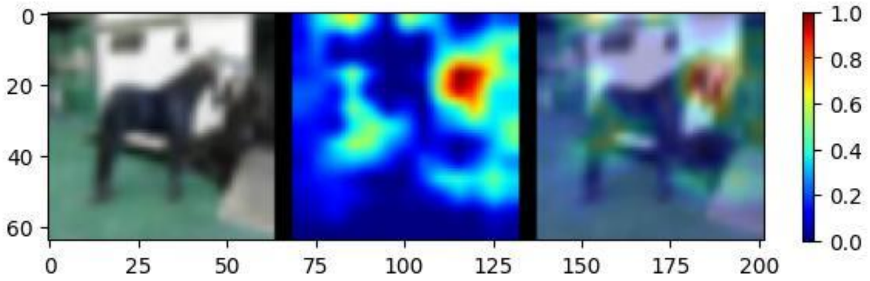
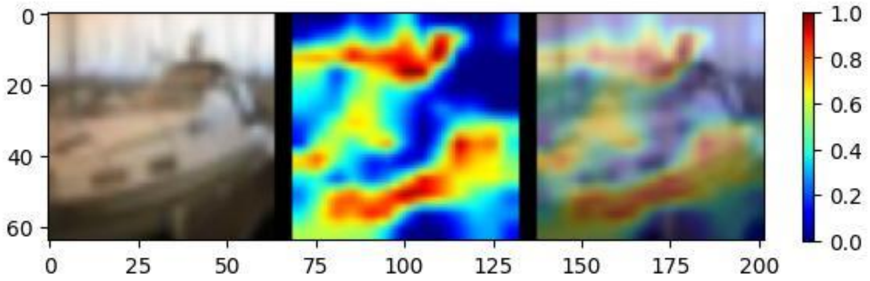
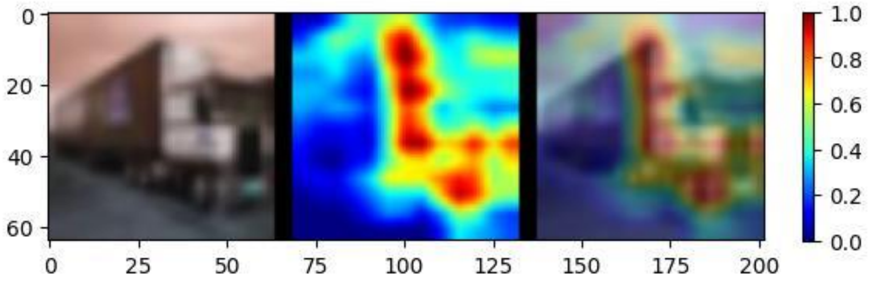
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 30, 30]             896
         MaxPool2d-2           [-1, 32, 15, 15]               0
            Conv2d-3           [-1, 32, 13, 13]           9,248
         MaxPool2d-4             [-1, 32, 6, 6]               0
            Conv2d-5             [-1, 16, 4, 4]           4,624
         MaxPool2d-6             [-1, 16, 2, 2]               0
            Linear-7                  [-1, 128]           8,320
            Linear-8                   [-1, 32]           4,128
            Linear-9                   [-1, 10]             330
================================================================
Total params: 27,546
Trainable params: 27,546
Non-trainable params: 0
```

**Model1 arch: 3 conv2d、3maxpool、3 dense**

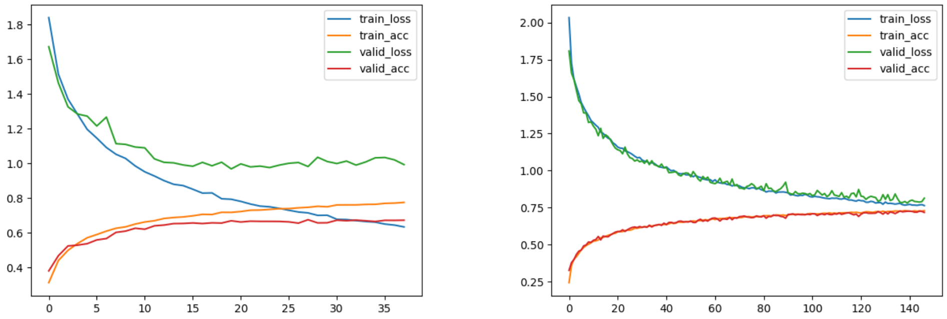
| | **M1** | **M2** | **M3** | **M4** | **M5** | **M6** | **M7** | **M8** |
|---|---|---|---|---|---|---|---|---|
| **aug** | False | False | False | False | True | True | True | True |
| **loss** | ce | ce | ce | ce | ce | ce | ce | ce |
| **opt** | Adam | Adam | Adam | Adam | Adam | Adam | RMSp | SGD |
| **bs** | 256 | 256 | 256 | 256 | 256 | 512 | 512 | 512 |
| **lr** | 0.001 | 0.003 | 0.005 | 0.01 | 0.001 | 0.001 | 0.001 | 0.001 |
| **use epoch** | 70 | 27 | 39 | 34 | 65 | 136 | 123 | 320 |
| **Train_loss** | 0.602 | 0.714 | 0.712 | 0.864 | 0.818 | 0.778 | 0.963 | 0.988 |
| **Train_acc** | 78.55 | 74.63 | 74.41 | 69.89 | 71.12 | 72.57 | 66.17 | 65.09 |
| **Valid_loss** | 1.000 | 0.981 | 0.999 | 1.047 | 0.849 | 0.785 | 0.928 | 0.989 |
| **Valid_acc** | 68.08 | 67.48 | 66.76 | 65.33 | 70.3 | 72.74 | 67.76 | 65.29 |
| **Test_acc** | 67.97 | 68.12 | 66.24 | 64.48 | 70.74 | 72.56 | 67.45 | 65.87 |
| | | | | | | BEST | | |

## M6 model Grad-Cam shows focus on

| | |
|---|---|
| airplane |  |
| automobile |  |
| bird |  |
| cat |  |
| deer |  |

| | |
|---|---|
| dog |  |
| frog |  |
| horse |  |
| ship |  |
| truck |  |

No augmentation vs augmentation train statistics



**In the above experiment, we tried tuning hyperparameters, including the optimizer, learning rate, and batch size. And tried using augmentation. Then visualize the model architecture.**

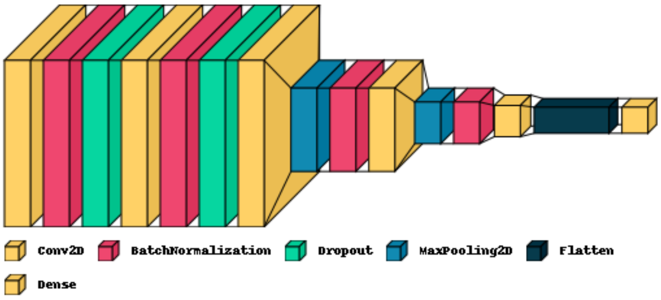
**Experiments showed that the optimizer Adam is better than SGD and RMSprop. The batch size and learning rate not only affect the speed of image training but also affects the overall performance.**

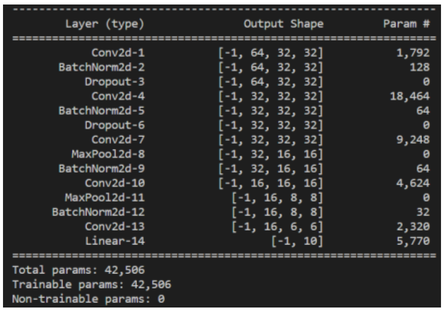
**Our data augmentation method uses rotation and mirroring. If the training data is not augmented, the training statistics curve will become steeper and the performance will be lower. Although it takes more time to train after data augmentation, overall the results are good.**

**Using grad-cam technology to see where the neural network is focusing. We can see that the focus on the heat map of the airplane is not very good. They are all focused on the background. Conversely, it's fine where trucks and horses focus. The part of the truck, the model obviously focuses on the position of the front of the car. As for the part of the horse, the model has attention on the head of the horse. Through the display of the heat map, we can know why the accuracy is not so ideal.**

**Therefore, we want to use more convolutions to deepen the information that the neural network can learn.**

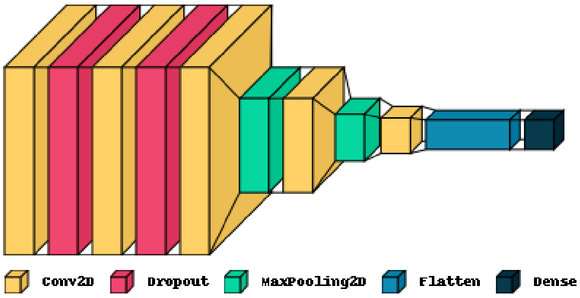
**The next experiment will add more convolutional layers, add dropout, and batch normalization.**

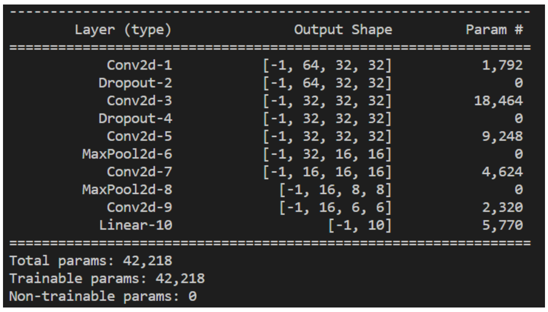




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
           Dropout-3           [-1, 64, 32, 32]               0
            Conv2d-4           [-1, 32, 32, 32]          18,464
       BatchNorm2d-5           [-1, 32, 32, 32]              64
           Dropout-6           [-1, 32, 32, 32]               0
            Conv2d-7           [-1, 32, 32, 32]           9,248
         MaxPool2d-8           [-1, 32, 16, 16]               0
       BatchNorm2d-9           [-1, 32, 16, 16]              64
           Conv2d-10           [-1, 16, 16, 16]           4,624
        MaxPool2d-11             [-1, 16, 8, 8]               0
      BatchNorm2d-12             [-1, 16, 8, 8]              32
           Conv2d-13             [-1, 16, 6, 6]           2,320
           Linear-14                   [-1, 10]           5,770
================================================================
Total params: 42,506
Trainable params: 42,506
Non-trainable params: 0
```

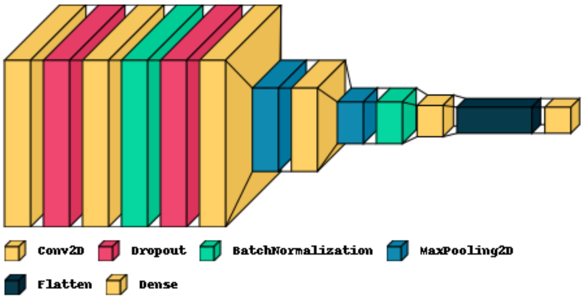
**Model2 arch: 5 conv2d 、2 maxpool(2x2)、2 drop_out、4 batch_n、1 dense**

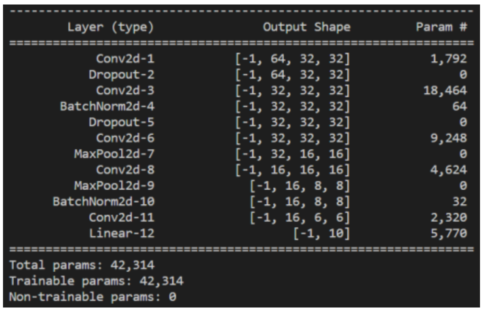




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
           Dropout-2           [-1, 64, 32, 32]               0
            Conv2d-3           [-1, 32, 32, 32]          18,464
           Dropout-4           [-1, 32, 32, 32]               0
            Conv2d-5           [-1, 32, 32, 32]           9,248
         MaxPool2d-6           [-1, 32, 16, 16]               0
            Conv2d-7           [-1, 16, 16, 16]           4,624
         MaxPool2d-8             [-1, 16, 8, 8]               0
            Conv2d-9             [-1, 16, 6, 6]           2,320
           Linear-10                   [-1, 10]           5,770
================================================================
Total params: 42,218
Trainable params: 42,218
Non-trainable params: 0
```

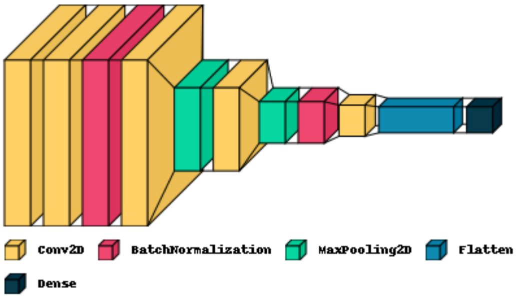
**Model3 arch: 5 conv2d 、2 maxpool(2x2)、2 drop_out、1 dense**

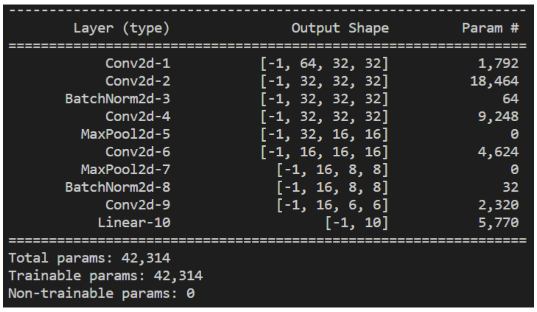




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
           Dropout-2           [-1, 64, 32, 32]               0
            Conv2d-3           [-1, 32, 32, 32]          18,464
       BatchNorm2d-4           [-1, 32, 32, 32]              64
           Dropout-5           [-1, 32, 32, 32]               0
            Conv2d-6           [-1, 32, 32, 32]           9,248
         MaxPool2d-7           [-1, 32, 16, 16]               0
            Conv2d-8           [-1, 16, 16, 16]           4,624
         MaxPool2d-9             [-1, 16, 8, 8]               0
      BatchNorm2d-10             [-1, 16, 8, 8]              32
           Conv2d-11             [-1, 16, 6, 6]           2,320
           Linear-12                   [-1, 10]           5,770
================================================================
Total params: 42,314
Trainable params: 42,314
Non-trainable params: 0
```

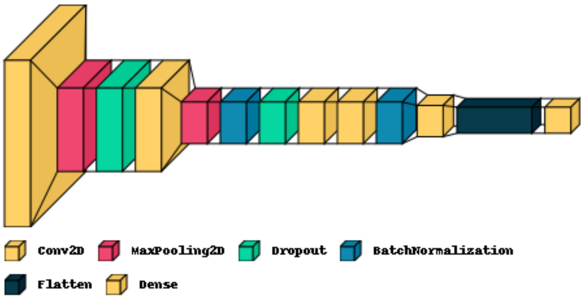
**Model4 arch: 5 conv2d 、2 maxpool(2x2)、2 drop_out、2 batch_n、1 dense**

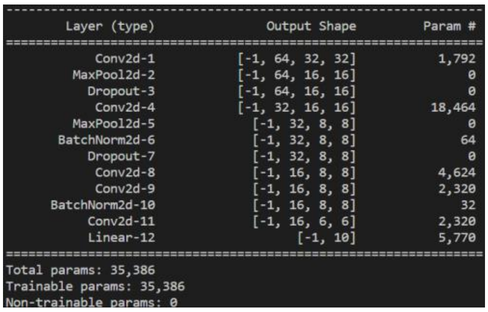




```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
            Conv2d-2           [-1, 32, 32, 32]          18,464
       BatchNorm2d-3           [-1, 32, 32, 32]              64
            Conv2d-4           [-1, 32, 32, 32]           9,248
         MaxPool2d-5           [-1, 32, 16, 16]               0
            Conv2d-6           [-1, 16, 16, 16]           4,624
         MaxPool2d-7             [-1, 16, 8, 8]               0
       BatchNorm2d-8             [-1, 16, 8, 8]              32
            Conv2d-9             [-1, 16, 6, 6]           2,320
           Linear-10                   [-1, 10]           5,770
================================================================
Total params: 42,314
Trainable params: 42,314
Non-trainable params: 0
```

**Model5 arch: 5 conv2d、2 maxpool(2x2)、2 batch_n、1 dense**



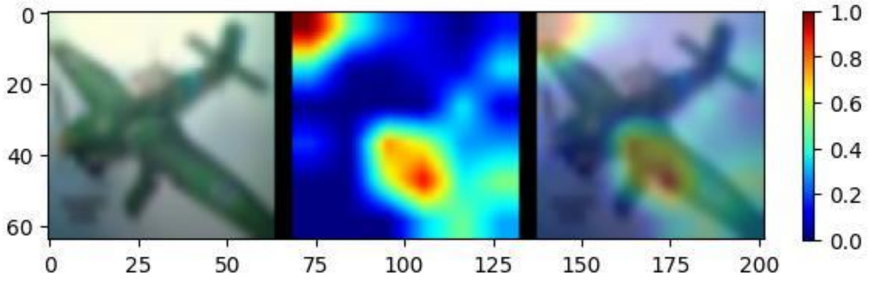
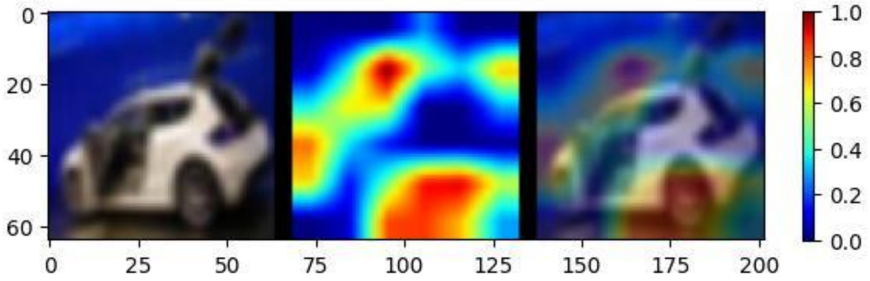
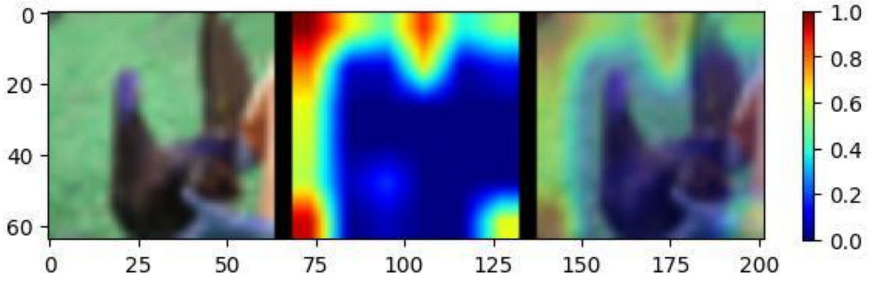
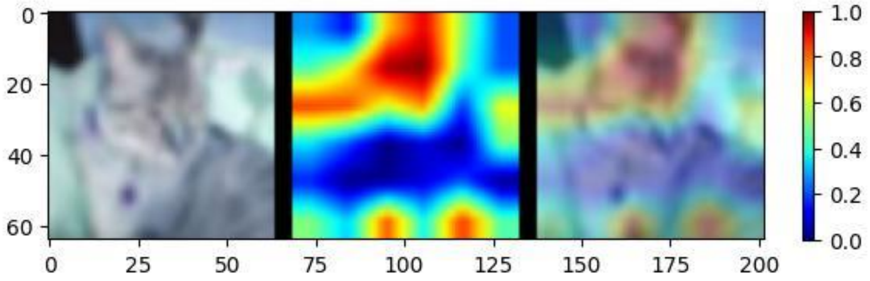
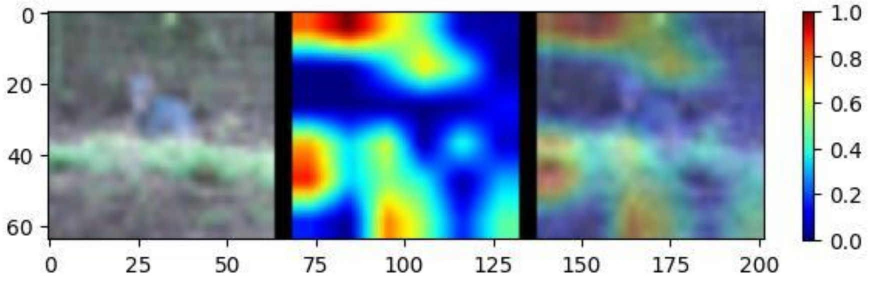


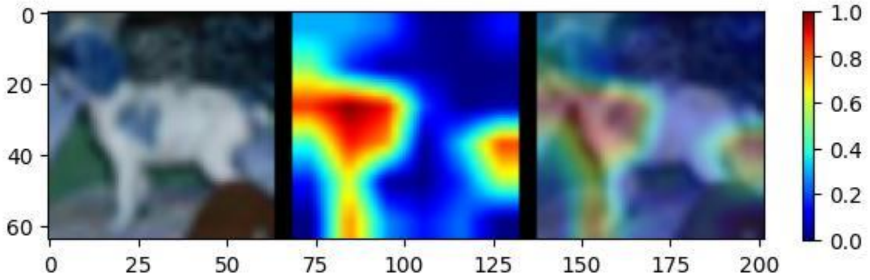
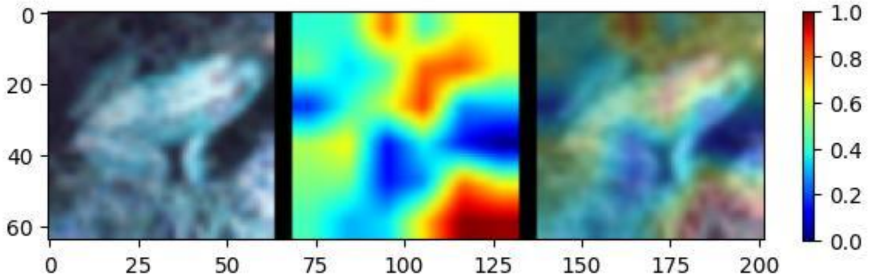
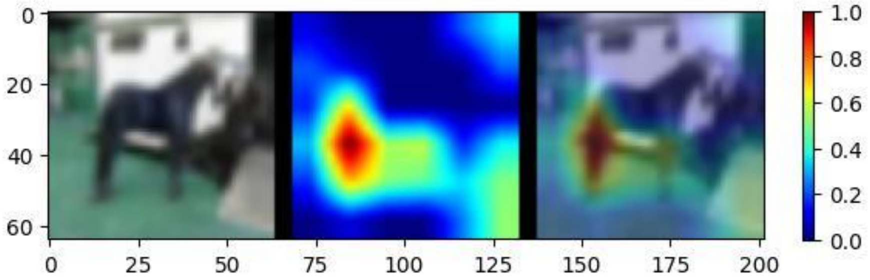
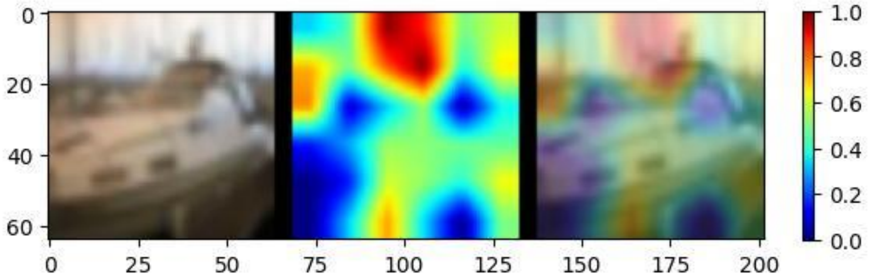
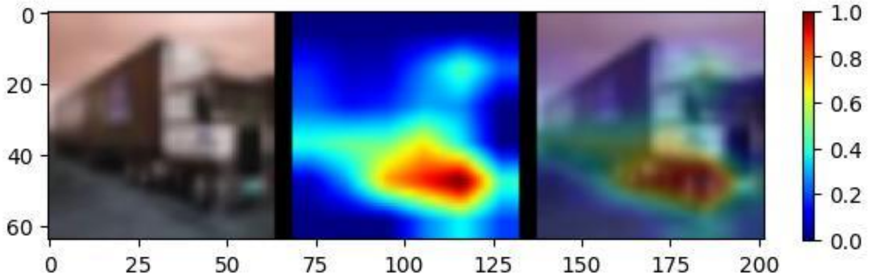
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
         MaxPool2d-2           [-1, 64, 16, 16]               0
           Dropout-3           [-1, 64, 16, 16]               0
            Conv2d-4           [-1, 32, 16, 16]          18,464
         MaxPool2d-5             [-1, 32, 8, 8]               0
       BatchNorm2d-6             [-1, 32, 8, 8]              64
           Dropout-7             [-1, 32, 8, 8]               0
            Conv2d-8             [-1, 16, 8, 8]           4,624
            Conv2d-9             [-1, 16, 8, 8]           2,320
      BatchNorm2d-10             [-1, 16, 8, 8]              32
           Conv2d-11             [-1, 16, 6, 6]           2,320
           Linear-12                   [-1, 10]           5,770
================================================================
Total params: 35,386
Trainable params: 35,386
Non-trainable params: 0
```

**Model6 arch: 5 conv2d、2 maxpool(2x2)、2 drop_out、2 batch_n、1 dense**

| | Model2 | Model3 | Model4 | Model5 | Mode6 |
|---|---|---|---|---|---|
| **aug** | True | | | | |
| **loss** | ce | | | | |
| **opt** | Adam | | | | |
| **bs** | 512 | | | | |
| **lr** | 0.001 | | | | |
| **use epoch** | 30 | 95 | 91 | 59 | 78 |
| **Train_loss** | 0.698 | 0.7299 | 0.595 | 0.5628 | 0.6841 |
| **Train_acc** | 75.43 | 74.31 | 78.99 | 80.26 | 75.87 |
| **Valid_loss** | 0.791 | 0.701 | 0.578 | 0.6177 | 0.644 |
| **Valid_acc** | 73.93 | 75.45 | 80.05 | 79.17 | 77.9 |
| **Test_acc** | 72.55 | 75.34 | 79.29 | 78.0 | 77.81 |
| | | | BEST | | |

### Model4 Grad-Cam shows focus on

| | |
|---|---|
| airplane | <br> |
| automobile | <br> |
| bird | <br> |
| cat | <br> |
| deer | <br> |

| | |
|---|---|
| dog |  |
| frog |  |
| horse |  |
| ship |  |
| truck |  |

**In the second experiment, we tried to use more convolutional layers to make the model better.**

**In Model 2, we use 5 conv2d、2 maxpool(2x2)、2 drop_out、4 batch_n、1 dense. Originally thought that it could achieve better performance than 3-layer convolution. But the experimental results are very poor.**

**Assuming whether it is because of adding too much batch normalization. In Model 3, we remove all batch normalization. Performance improved this time.**

**Next, we will re-add the 2 layers of the batch normalization back. And They are all placed after 2 layers of convolution. The result was the best.**

**Then, remove the dropout layer of the best model and change where batch normalization is placed. But neither can get better results.**

**We can see that compared with model 1, model 4 has a better focus on airplane pictures. The wing of the plane can be noticed. Overall, only poorer in bird focus.**

**Conclusion:**

**More convolutional layers do not necessarily lead to better performance. It may be easy to overfit in training because of too many parameters.**

**Appropriately discarding some parameters can make the overall model better. Batch normalization does not only need to be used to achieve good results, it needs some experiments to know where to put it to get the greatest benefit. The amount of data fed into a step during training will also affect the entire training result. The Adam optimizer is the best among these self-tried architectures.**

**The learning rate will affect the time required for training. If it is too low, the optimal solution may not be found and stuck in the regional solution. If it is too high, it will oscillate back and forth.**

**Finally, the heat map of grad-cam can clearly let us know where the model focuses on each category of images. This allows us to avoid the situation where the accuracy rate drops significantly due to background changes when the system is officially launched.**

**Experiments equipment:**

**GPU: RTX3060 12G VRAM**