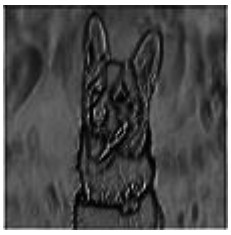
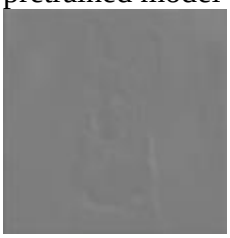
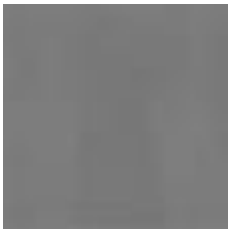
楊英豪 B10803207

Example 1.1

This is the visualization of the output, extracted from the layer-index-5 and layer-index-10.
Note: The picture shown below is original image, feature maps from layer-index-5 and layer-index-10


Analysis on two different feature maps from two convolutional layers with and without pretrained model, The picture below is a layer-index-5 and layer-index-10 (from left to right) without any pretrained model

I took another 2 pictures from the internet with the same class “Dog”, then I use the same piece of code, just need to change the file path, to get the layer-index-5 and layer-index-10 (from left to right) using VGG pretrained model




I then compared the two pictures in the layer-index-5 which is the step 6-8 in the documents, then calculate the cosine similarity between two pictures, which is less than 0.6 which means

they are the same based on the code, although the predicted object was not accurate, maybe because not all species of dogs were included in the training data

```
TOP_1
Probablity:0.23888780176639557
Predicted: 'Labrador retriever'

TOP_2
Probablity:0.2074865847826004
Predicted: 'kuvasz'

TOP_3
Probablity:0.17493028938770294
Predicted: 'golden retriever'

The second picture classification predict:
Probablity TOP-3:


TOP_1
Probablity:0.9785510301589966
Predicted: 'Pomeranian'

TOP_2
Probablity:0.007434830069541931
Predicted: 'keeshond'

TOP_3
Probablity:0.004152935929596424
Predicted: 'chow

Verification:
They are the same!
Their cosine_distance:tensor([0.4755], device='cuda:0', grad_fn=<SumBackward1>)
Their euclidean_dist:80.19735717773438
```

Next I use the VGG-16 model that is pretrained to train the CIFAR-100 datasets. The result of this training can be as expected, it can't predict that the object in the image is a dog

```
Probability TOP-3:


TOP_1
Probability:0.576285719871521
Predicted: 'sea'

TOP_2
Probability:0.3022152781486511
Predicted: 'cloud'

TOP_3
Probability:0.047054633498191833
Predicted: 'plain'
```

In the point 13, where it's asked to use VGG-16 normalization batch model to train the CIFAR-100 from scratch, torchvision.models.vgg16_bn(pretrained=False) here pretrained to false, and there will be normal batch inside. Because VGG is a large network, if there is no pretrained data the predictions will be far off, it's very hard to train the network from scratch and get a good prediction. I also use the parameters specified in the slides. I also use the same class method of Pretrained_VGGNet(), but changed the parameter by adding my own model, which is trained using the same for loop training in the original code. As shown below the top 3 probability isn't good especially compared to the pretrained model, because VGGNet dataset size is small and it's hard to extract features, and the number of epoch is so low which is 10, that's why the model that is trained by scratch won't produce good results

```
my_model = torchvision.models.vgg16_bn(pretrained=False)
```

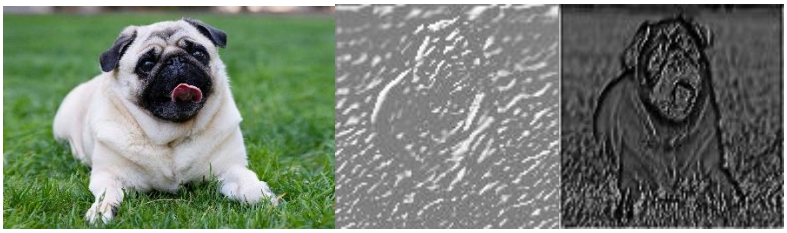
```
batch_size = 128
learning_rate = 0.005
input_size = 32
num_epoch = 10
```

```
class Pretrained_VGGNet():
    def __init__(self,img_path, my_model):
        self.img_path=img_path
        # Load pretrained model
        self.pretrained_model = my_model
```

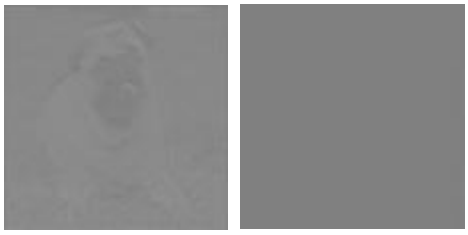
```
Probability TOP-3:


TOP_1
Probability:0.011873838491737843
Predicted: 'orange'

TOP_2
Probability:0.01173925306648016
Predicted: 'snake'

TOP_3
Probability:0.011567912064492702
Predicted: 'cra'
```

Exercise 1.1

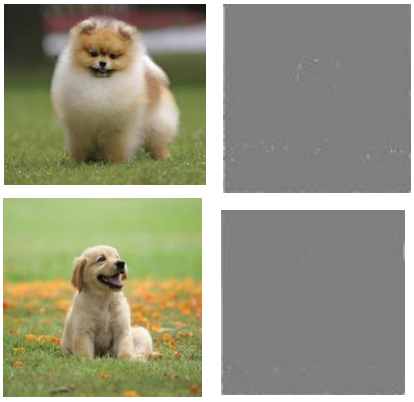
Here, I use the same dog image as the input, I then visualize the feature maps extracted from the layer index 8 as shown below, I also extracted from the layer index 5 with the pretrained model



And then I also use without the pretrained model from layer 5 and 8 (left to right), as can be seen the one without the pretrained model have a difficult time in extracting features from the image



Here I use the same dog pictures as I did in the example but this time I get the layer index 6, the features doesn't look clear



And I calculated the cosine similarity between two features extracted from the two same classes

```
Verification:
They are the same!
Their cosine_distance:tensor([0.4755], device='cuda:0', grad_fn=<SumBackward1>)
Their euclidean_dist:80.19735717773438
```

Then I download 2 images with the category of lion and mouse, the top 3 probabilities surprises me, both the lion and the mouse both got a palm_tree as their object (left is mouse, right is lion). Even though the number of epoch is 30, but the model still can't predict it correctly.

```
Probability TOP-3:

TOP_1
Probability:0.9906501173973083
Predicted: 'palm_tree'

TOP_2
Probability:0.003451545722782612
Predicted: 'sea'

TOP_3
Probability:0.002739907940849662
Predicted: 'pine_tree'
```

```
Probability TOP-3:

TOP_1
Probability:0.9748525619506836
Predicted: 'palm_tree'

TOP_2
Probability:0.01634875126183033
Predicted: 'pine_tree'

TOP_3
Probability:0.008007489144802094
Predicted: 'maple_tree'
```

I then try to use the vgg19 batch normalization model from scratch

Then for the image of lion and mouse

```
)
Probability TOP-3:

TOP_1
Probability:0.010596008040010929
Predicted: 'skyscraper'

TOP_2
Probability:0.010476424358785152
Predicted: 'plain'

TOP_3
Probability:0.01047483365982771
Predicted: 'leopard'
```

```
Probability TOP-3:

TOP_1
Probability:0.010592072270810604
Predicted: 'sweet_pepper'

TOP_2
Probability:0.010585903190076351
Predicted: 'road'

TOP_3
Probability:0.010551286861300468
Predicted: 'plain'
```

For both pretrained and the model that I trained both got a bad prediction, maybe because low epoch number , small input image size that makes it hard to recognize the features