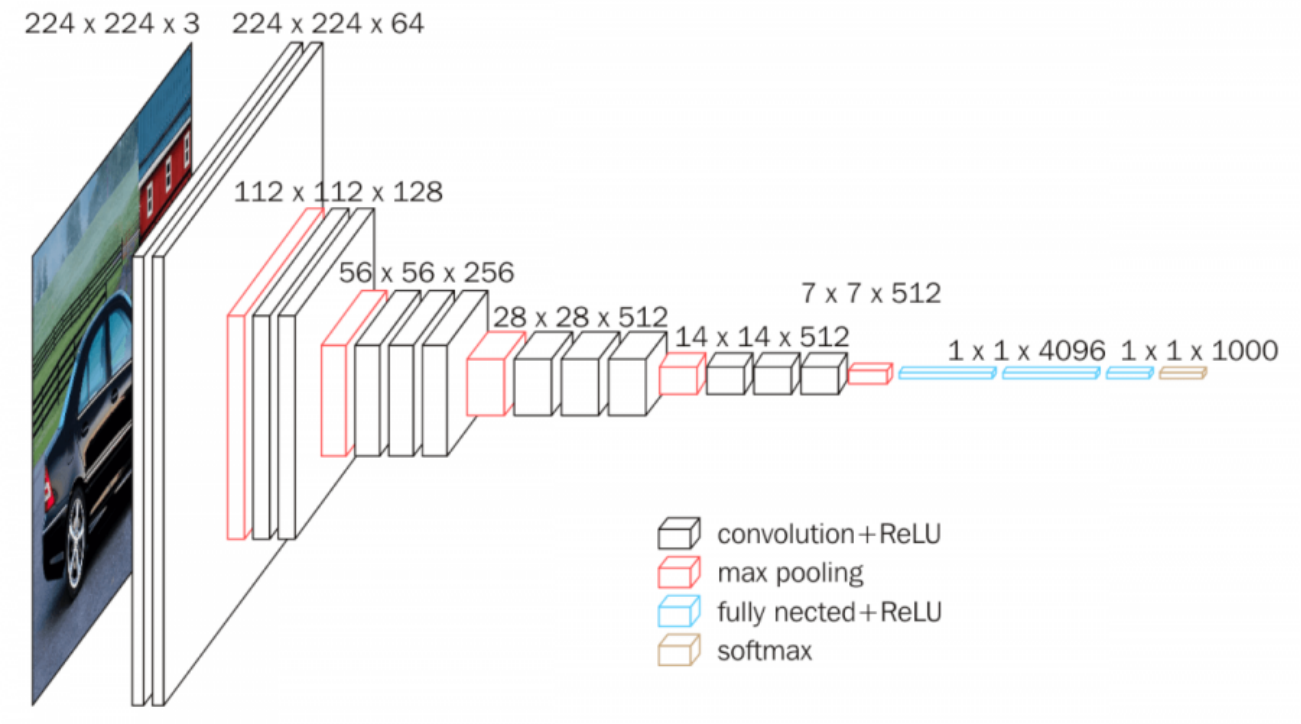
VGG net

```
1 from IPython.display import Image
2 image_url = 'https://bitbucket.org/hyuk125/lg_dic/raw/12b61c0c3c223378d52ae530
3 Image(image_url)
```




```
1 import numpy as np
2
3 import torch
4 import torchvision
5 import torchvision.transforms as transforms
6
7 import matplotlib.pyplot as plt
8 import os
```

```
1 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

저장 중...

CIFAR data import

OFFline pickle data

```
1 # def unpickle(file):
2 #     with open(file, 'rb') as fo:
3 #         dict = pickle.load(fo, encoding='bytes')
4 #     return dict
```

```
5 # data = unpickle(os.path.join(os.path.join(path, 'cifar-10-batches-py') ,'/da
```

Online PIL image data

```
1 # # For online situation
2 # trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
3 #                                         download=True, transform=transform)
```

OFFline PIL data

```
1 transform = transforms.Compose(
2     [transforms.ToTensor(),
3      transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
4 path = 'https://bitbucket.org/hyuk125/lg_dic/raw/4350ceab62a7d5d0e22fd9a578b6a
5 trainset = torchvision.datasets.CIFAR10(root=path, train=True,
6                                         download=False, transform=transform)
7
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-4-4bbb43b7a9c6> in <module>()
      4 path =
'https://bitbucket.org/hyuk125/lg_dic/raw/4350ceab62a7d5d0e22fd9a578b6abe0c6660cec/dataset_da

      5 trainset = torchvision.datasets.CIFAR10(root=path, train=True,
----> 6                                         download=False, transform=transform)

/usr/local/lib/python3.7/dist-packages/torchvision/datasets/cifar.py in __init__(self, root,
train, transform, target_transform, download)
     66
     67         if not self._check_integrity():
---> 68             raise RuntimeError('Dataset not found or corrupted.' +
     69                                ' You can use download=True to download it')
     70

RuntimeError: Dataset not found or corrupted. You can use download=True to download it
```

SEARCH STACK OVERFLOW

저장 중...

Make Dataloader

```
1 trainloader = torch.utils.data.DataLoader(trainset, batch_size=1,
2                                           shuffle=True, num_workers=2)
```

Define Model

```
import torch.nn as nn
import torch.nn.functional as F
```

Original VGG network

conv : conv(in, out, kernel size=3, padding=1)
conv_2_block(in, out): conv-ReLU-conv-ReLU-MaxPooling
conv_3_block(in, out): conv-ReLU-conv-ReLU-conv-ReLU-MaxPooling

class Net(nn.Module):

```
def __init__(self, base_dim, num_classes=10):
    super(Net, self).__init__()
    self.feature = nn.Sequential(
        conv_2_block(3,base_dim),
        conv_2_block(base_dim,2*base_dim),
        conv_3_block(2*base_dim,4*base_dim),
        conv_3_block(4*base_dim,8*base_dim),
        conv_3_block(8*base_dim,8*base_dim),
    )
    self.fc_layer = nn.Sequential(
        nn.Linear(8*base_dim * 7 * 7, 4096),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(4096, 4096),
        nn.ReLU(),
        nn.Dropout(),
        nn.Linear(4096, num_classes)
    )

def forward(self, x):
    x = self.feature(x)
    x = x.view(x.size(0), -1)
    x = self.fc_layer(x)
    return x
```

1000).to(device)

저장 중...

Our small VGG model

- input - 32 * 32 * 3
- layer1 - 32 * 32 * 64 || 32 * 32 * 64 || maxpooling
- layer2 - 16 * 16 * 128 || 16 * 16 * 128 || maxpooling
- layer3 - 8 * 8 * 256 || 8 * 8 * 256 || maxpooling
- layer4 - Flatten || Fully connected 256 || Full connected 256 || output

과제1-1. Small VGG 모델을 작성하시오

```
class Net(nn.Module):
    #답변
```

Learning the model

```
learning_rate = 0.0002
```

```
import torch.optim as optim

criterion = nn.CrossEntropyLoss().to(device)
meters(), lr = learning_rate)
```

저장 중...

```
epochs = 2
```

과제1-2. 만든 small VGG 모델을 학습시키시오

```
for epoch in range(epochs):
    #답변
```

```
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

정확도 판단

Test dataset import

```
1 testset = torchvision.datasets.CIFAR10(root=path, train=False,
2                                        download=False, transform=transform)
3 testloader = torch.utils.data.DataLoader(testset, batch_size=len(testset),
4                                          shuffle=False, num_workers=2)
```

Confusion matrix and scores

```
1 test_iter = iter(testloader)
2 test_x, test_labels = test_iter.next()
```



```
))
s, 1)
```

Confusion matrix

```
1 from sklearn.metrics import confusion_matrix
2 predicted = predicted.cpu()
3 print(confusion_matrix(test_labels, predicted))
```

Precision

```
1 from sklearn.metrics import precision_score
2 print(precision_score(test_labels, predicted, average=None))
3 print(precision_score(test_labels, predicted, average='weighted'))
```

Recall

```
1 from sklearn.metrics import recall_score
2 print(recall_score(test_labels, predicted, average=None))
3 print(recall_score(test_labels, predicted, average='weighted'))
```

✓ 0초 오전 11:45에 완료됨

저장 중...