PyTorch practice

Convolutional networks: VGG-Net

Matteo Boschini, Angelo Porrello, Lorenzo Bonicelli

November 26, 2021

University of Modena and Reggio Emilia

Agenda

ConvNets: VGG-16

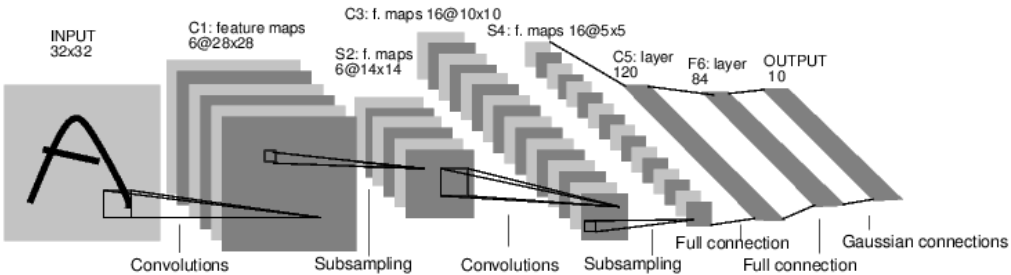
References

ConvNets: VGG-16

Neural Networks: LeNet [5]

Neural networks can be constructed using the `torch.nn` package.

An `nn.Module` contains layers, and a method `forward(input)` that returns the output.



Let's define the `__init__()` method:

```
class Net(nn.Module):

    def __init__(self):
        pass

    def forward(self, x):
        pass

net = Net()
print(net)
```

```
def __init__(self):
        super(Net, self).__init__()

        # 1 input image channel
        # 6 output channels
        # 3x3 square convolution kernel
        self.conv1 = nn.Conv2d(1, 6, 3)
        self.conv2 = nn.Conv2d(6, 16, 3)

        # a linear operation: y = Wx + b
        # 16 feature maps, with resolution 6 * 6
        self.fc1 = nn.Linear(16 * 6 * 6, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

nn.Module skeleton

Let's define the `forward(x)` method:

```
class Net(nn.Module):

    def __init__(self):
        pass

    def forward(self, x):
        pass

net = Net()
print(net)
```

```
# Let's define the entire network
def forward(self, x):
        # 3x3 2d convolution
        x = self.conv1(x) # 3x3 2d convolution
        x = F.relu(x) # RELU activation function
        x = F.max_pool2d(x, (2, 2)) # 2x2 max pooling
        x = F.max_pool2d(F.relu(self.conv2(x)), (2, 2))
        x = x.view(-1, 16 * 6 * 6)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

nn.Module skeleton

Once the network has been defined, we are able to print it:

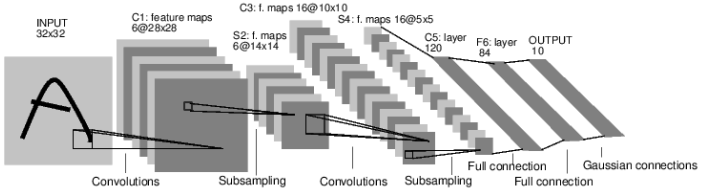
```
class Net(nn.Module):

    def __init__(self):
        self.conv1 = ...
        self.conv2 = ...
        ...

    def forward(self, x):
        x = ...
        x = ...
        return x
```

```
if __name__ == "__main__":
    net = Net()
    print(net)
```

```
Net(
  (conv1): Conv2d(1, 6, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(3, 3), stride=(1, 1))
  (fc1): Linear(in_features=576, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```



Let's try a random 32x32 input. Note: expected input size of this net (LeNet) is 32x32.

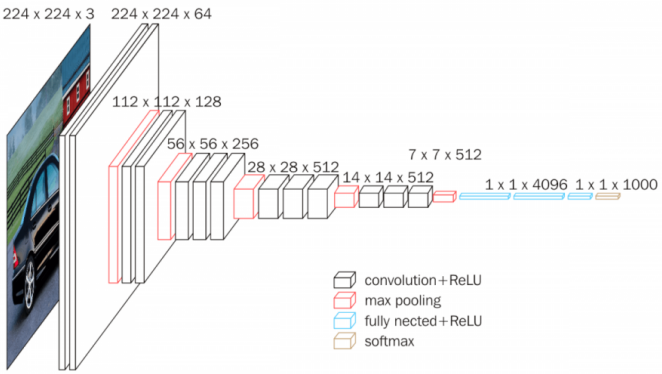
To use this net on MNIST dataset [3], please resize the images from the dataset to 32x32.

```
if __name__ == "__main__":
    net = Net()
    B = 1 # batch size
    fin = 1 # one input channel (grayscale)
    w, h = 32, 32
    x = torch.randn(B, fin, w, h)
    output = net(x)
    print(output)
```

```
tensor([[ 0.0250,  0.0280, -0.0704,
        -0.1894, -0.0126,  0.0648,
         0.0774,  0.0855, -0.1219,
        -0.0386]], grad_fn=<AddmmBackward>)
```

VGG16 [4] is a convolutional neural network model proposed in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition".

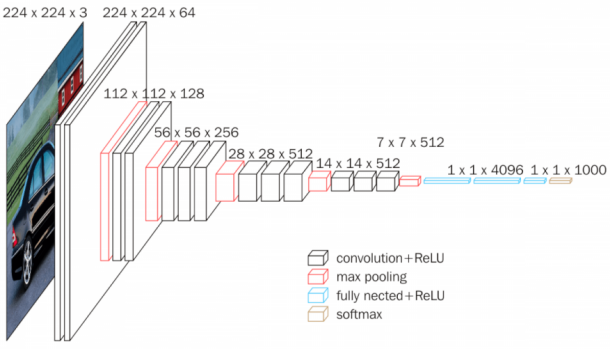
The model achieves 92.7% top-5 test accuracy in ImageNet [1], which is a dataset of over 14 million images belonging to **1000 classes**.

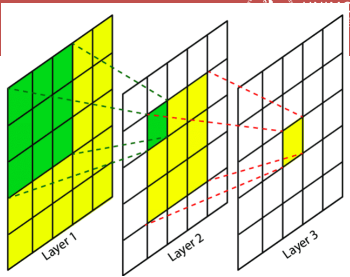


Neural Networks: VGG-16

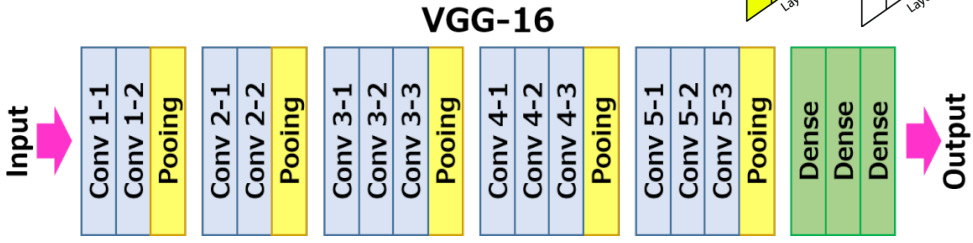
It makes the improvement over AlexNet [2] by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.

- conv. stride 1 – no loss of information
- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers





Depth matters!



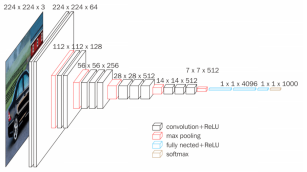
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Let's code VGG-16!

- Main paper: `https://arxiv.org/pdf/1409.1556.pdf`
- Slides: `http://www.robots.ox.ac.uk/~karen/pdf/ILSVRC_2014.pdf`
- GitHub: `https://github.com/Abhisek-/VGG`

Useful Resources: `https://pytorch.org/docs/stable/nn.html`

- `nn.Sequential`
- `nn.Conv2d`
- `nn.ReLU`
- `nn.MaxPool2d`
- `nn.AdaptiveAvgPool2d`
- `nn.Linear`
- `nn.Dropout`
- `torch.flatten`




```
class VGG16(nn.Module):

    def __init__(self):
        pass

    def forward(self, x):
        pass

net = VGG16()
print(net)
```

Your Tasks Today

1. Code VGG16 in PyTorch
2. Make sure that it works on ImageNet-like input (i.e. 3x224x224 tensors)
3. Optional: make it also work on CIFAR-like input (3x32x32 tensors)
4. Very optional: train it on the CIFAR-10 dataset

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton.
Imagenet classification with deep convolutional neural networks.
In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[3] Y. LeCun.
The mnist database of handwritten digits.
http://yann. lecun. com/exdb/mnist/, 1998.

[4] K. Simonyan and A. Zisserman.
Very deep convolutional networks for large-scale image recognition.
In *International Conference on Learning Representations*, 2015.

[5] Y. B. P. H. Yann LeCun, Leon Bottou.
P. gradient-based learning applied to document recognition.
Proceedings of the IEEE 86, 1998.