CS6501: Deep Learning for Visual Recognition
Convolutional Neural Networks
Today’s Class

Automatic Differentiation (AutoGrad)
Convolutional Neural Networks
• Revisiting Convolutions
• The Convolutional Layer
• Strided Convolutions / Grouped Convolutions / Dilated Convolutions
• Spatial Pooling Operations
Automatic Differentiation

You only need to write code for the forward pass, backward pass is computed automatically.

Pytorch (Facebook -- mostly): https://pytorch.org/

Tensorflow (Google -- mostly): https://www.tensorflow.org/

DyNet (team includes UVA Prof. Yangfeng Ji): http://dynet.io/
Convolutional Layer

Input image \(*\) Weights \(\rightarrow\) Output image

Input image:

\[
\begin{matrix}
4 & 5 & 7 & 6 & 6 \\
3 & 2 & 8 & 0 & 7 \\
6 & 7 & 7 & 1 & 5 \\
3 & 0 & 1 & 1 & 1 \\
4 & 3 & 2 & 1 & 7 \\
\end{matrix}
\]

Weights:

\[
\begin{matrix}
0 & 0 & 0 \\
1 & 0 & 1 \\
0 & 0 & 0 \\
\end{matrix}
\]

Output image:

\[
\begin{matrix}
11 & 2 & 15 \\
13 & 8 & 12 \\
\end{matrix}
\]
Convolutional Layer
Convolutional Layer

Weights
Convolutional Layer
Convolutional Layer

Weights
Convolutional Layer (with 4 filters)

Input: 1x224x224

weights: 4x1x9x9

Output: 4x224x224

if zero padding, and stride = 1
Convolutional Layer (with 4 filters)

Input: 1x224x224

weights: 4x1x9x9

Output: 4x112x112

if zero padding, but stride = 2
Convolutional Layer in pytorch

\[
\text{class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True) [source]}
\]

Input

\[
in_{\text{channels}} \text{ (e.g. 3 for RGB inputs)}
\]

Output

\[
\text{out}_{\text{channels}} \text{ (equals the number of convolutional filters for this layer)}
\]

\[
\text{kernel}_{\text{size}} \\
\text{out}_{\text{channels}} \times \text{in}_{\text{channels}}
\]

[diagram showing input and output dimensions and kernel size]
Convolutional Network: LeNet

Yann LeCun

Gradient-based learning applied to document recognition
Y LeCun, L Bottou, Y Bengio, P Haffner
Proceedings of the IEEE 86 (11), 2278-2324
LeNet in Pytorch

```python
# LeNet is French for The Network, and is taken from Yann Lecun's 98 paper
# on digit classification http://yann.lecun.com/exdb/lenet/
# This was also a network with just two convolutional layers.
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        # Convolutional layers.
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)

        # Linear layers.
        self.fc1 = nn.Linear(16*5*5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        out = F.relu(self.conv1(x))
        out = F.max_pool2d(out, 2)
        out = F.relu(self.conv2(out))
        out = F.max_pool2d(out, 2)

        # This flattens the output of the previous layer into a vector.
        out = out.view(out.size(0), -1)
        out = F.relu(self.fc1(out))
        out = F.relu(self.fc2(out))
        out = self.fc3(out)
        return out
```
SpatialMaxPooling Layer

take the max in this neighborhood
LeNet Summary

• 2 Convolutional Layers + 3 Linear Layers

• + Non-linear functions: ReLUs or Sigmoid
  + Max-pooling operations
New Architectures Proposed

• Alexnet (Krizhevsky et al NIPS 2012) [Required Reading]

• VGG (Simonyan and Zisserman 2014)

• GoogLeNet (Szegedy et al CVPR 2015)

• ResNet (He et al CVPR 2016)

• DenseNet (Huang et al CVPR 2017)
Convolutional Layers as Matrix Multiplication

https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
Convolutional Layers as Matrix Multiplication

https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
Convolutional Layers as Matrix Multiplication

Pros?
Cons?

https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/
CNN Computations are Computationally Expensive

• However highly parallelizable
• GPU Computing is used in practice
• CPU Computing in fact is prohibitive for training these models
Questions?