CS4501: Introduction to Computer Vision
Convolutional Neural Networks (CNNs)
Previous

• Linear Classifier (Softmax / Max-margin)
• Generalization / Regularization / Overfitting / Underfitting
• Neural Networks
  • The Perceptron Model
  • The Multi-layer Perceptron (MLP)
• Forward-pass in an MLP (Inference)
• Backward-pass in an MLP (Backpropagation)
Today’s Class

• The Convolutional Layer
• Convolutional Neural Networks
• The LeNet Network
• The AlexNet Network and the ImageNet Dataset and Challenge
Convolutional Layer

Input image * Weights → Output image

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>7</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Weights:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Output image:

<table>
<thead>
<tr>
<th>11</th>
<th>2</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>
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

```python
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)
```

- **in_channels**: (e.g. 3 for RGB inputs)
- **out_channels**: (equals the number of convolutional filters for this layer)
- **kernel_size**
- **Input**: `in_channels`
- **Output**: `out_channels` (equals the number of convolutional filters for this layer)
Convolutional Network: LeNet

Yann LeCun
Chief AI Scientist at Facebook & Silver Professor at the Courant Institute, New York University
Verified email at cs.nyu.edu - Homepage
AI, machine learning, computer vision, robotics, image compression

<table>
<thead>
<tr>
<th>TITLE</th>
<th>CITED BY</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep learning</td>
<td>37330</td>
<td>2015</td>
</tr>
<tr>
<td>Y LeCun, Y Bengio, G Hinton</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nature 521 (7553), 436-444</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient-based learning applied to document recognition</td>
<td>35502</td>
<td>1998</td>
</tr>
<tr>
<td>Y LeCun, I Bottou, Y Bengio, P Haffner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proceedings of the IEEE 86 (11), 2278-2324</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LeNet in Pytorch

# 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.

```python
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 a neighborhood and replace such neighborhood.

https://computersciencewiki.org/index.php/Max-pooling/_Pooling
SpatialMaxPooling Layer

Take the max in a neighborhood and replace such neighborhood.

https://computersciencewiki.org/index.php/Max-pooling_/Pooling
LeNet Summary

- 2 Convolutional Layers + 3 Linear Layers
- Non-linear functions: ReLUs or Sigmoids
- Max-pooling operations
New Architectures Proposed

- Alexnet (Krizhevsky et al NeurIPS 2012)
- VGG (Simonyan and Zisserman 2014)
- GoogLeNet (Szegedy et al CVPR 2015)
- ResNet (He et al CVPR 2016)
- DenseNet (Huang et al CVPR 2017)
- EfficientNet (Tan and Le ICML 2019)
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

Input Matrix

Patch 1
Patch 2
...

Number of Patches

Kernel Matrix

Kernel 1
Kernel 2
...

Number of Kernels

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
ILSVRC:
Imagenet Large Scale Visual Recognition Challenge
[Russakovsky et al 2014]
The Problem: Classification

Classify an image into 1000 possible classes:
e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee, red fox, banjo, barbell, hourglass, knot, maze, viaduct, etc.

cat, tabby cat (0.71)
Egyptian cat (0.22)
red fox (0.11)
.....
The Data: ILSVRC

Imagenet Large Scale Visual Recognition Challenge (ILSVRC): Annual Competition

1000 Categories

~1000 training images per Category

~1 million images in total for training

~50k images for validation

Only images released for the test set but no annotations, evaluation is performed centrally by the organizers (max 2 per week)
The Evaluation Metric: Top K-error

True label: Abyssinian cat

cat, tabby cat (0.61)  
Egyptian cat (0.22)  
red fox (0.11)  
Abyssinian cat (0.10)  
French terrier (0.03)  
.....

Top-1 error: 1.0  Top-1 accuracy: 0.0  
Top-2 error: 1.0  Top-2 accuracy: 0.0  
Top-3 error: 1.0  Top-3 accuracy: 0.0  
Top-4 error: 0.0  Top-4 accuracy: 1.0  
Top-5 error: 0.0  Top-5 accuracy: 1.0
Top-5 error on this competition (2012)
Alexnet (Krizhevsky et al NIPS 2012)

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca
Alexnet

https://www.saagie.com/fr/blog/object-detection-part1
Pytorch Code for Alexnet

• In-class analysis

https://github.com/pytorch/vision/blob/master/torchvision/models/alexnet.py
Dropout Layer

(a) Standard Neural Net

(b) After applying dropout.

Srivastava et al 2014
What is happening?

https://www.saagie.com/fr/blog/object-detection-part1
**SIFT + FV + SVM (or softmax)**

1. **Feature extraction (SIFT)**
2. **Feature encoding (Fisher vectors)**
3. **Classification (SVM or softmax)**

**Deep Learning**

1. **Convolutional Network**
   - (includes both feature extraction and classifier)
Preprocessing and Data Augmentation
Preprocessing and Data Augmentation
Preprocessing and Data Augmentation

224x224
Preprocessing and Data Augmentation

224x224
True label: Abyssinian cat
• Using ReLUs instead of Sigmoid or Tanh
• Momentum + Weight Decay
• Dropout (Randomly sets Unit outputs to zero during training)
• GPU Computation!

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse coding [2]</td>
<td>47.1%</td>
<td>28.2%</td>
</tr>
<tr>
<td>SIFT + FVs [24]</td>
<td>45.7%</td>
<td>25.7%</td>
</tr>
<tr>
<td>CNN</td>
<td>37.5%</td>
<td>17.0%</td>
</tr>
</tbody>
</table>
VGG Network

Top-5:

https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

Simonyan and Zisserman, 2014.
Questions?