CS4501: Introduction to Computer Vision
Deeper Convolutional Neural Network Architectures
Last Class

• Neural Networks – multilayer perceptron model (MLP)
• Backpropagation
• Convolutional Neural Networks
Today’s Class

- More on Convolutional Neural Networks
- Convolutional Neural Networks proposed
Convolutional Layer
Convolutional Layer

Weights
Convolutional Layer

Weights

Matrix:

4 5 7 6 6
3 2 8 0 7
6 3 2 1 1
4 3 2 1 1

Output:

11 2 15
13 8 12
4 1

Weights:

0 0 0
1 0 1
0 0 0
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

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

- **in_channels**: (e.g. 3 for RGB inputs)
- **out_channels**: (equals the number of convolutional filters for this layer)
- **kernel_size**

Input

Output

- `input_channels` (e.g. 3 for RGB inputs)
- `out_channels` (equals the number of convolutional filters for this layer)
Convolutional Network: LeNet
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
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
LeNet Summary

• 2 Convolutional Layers + 3 Linear Layers

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

• Alexnet (Krizhevsky et al NIPS 2012)
• VGG (Simonyan and Zisserman 2014)
• GoogLeNet (Szegedy et al CVPR 2015)
• ResNet (He et al CVPR 2016)
• DenseNet (Huang et al CVPR 2017)
ILSVRC:
Imagenet Large Scale Visual Recognition Challenge
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

<table>
<thead>
<tr>
<th>Top-K</th>
<th>Error</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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

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

Feature extraction (SIFT) → Feature encoding (Fisher vectors) → Classification (SVM or softmax)

Deep Learning

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
Other Important Aspects

• 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.
GoogLeNet

https://github.com/kuangliu/pytorch-cifar/blob/master/models/googlenet.py

Szegedy et al. 2014

BatchNormalization Layer (Ioffe and Szegedy 2015)

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_1...m\}$;

**Parameters to be learned:** $\gamma, \beta$

**Output:** $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

\[
\begin{align*}
\mu_\mathcal{B} & \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \quad \text{// mini-batch mean} \\
\sigma_\mathcal{B}^2 & \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \quad \text{// mini-batch variance} \\
\hat{x}_i & \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \quad \text{// normalize} \\
y_i & \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad \text{// scale and shift}
\end{align*}
\]

ResNet (He et al CVPR 2016)

Sorry, does not fit in slide.

http://felixlaumon.github.io/assets/kaggle-right-whale/resnet.png

https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py
Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)

ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10

152 layers
Questions?