Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
Deep Convolutional Neural Networks (Deep ConvNets)

- **Convolution** uses local weights shared across the whole image
- **Pooling** shrinks the spatial dimensions
- **Activation** (ReLU, Sigmoid, Tanh,..)
- **Fully connected**
Deep ConvNets for image classification

- **AlexNet** 8 layers, 62M parameters

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton
ImageNet Classification with Deep Convolutional Neural Networks.
In *NIPS, 2012.*
Deep ConvNets for image classification

- **VGG16** (very deep) 16 layers, 138M parameters
  - Increasing the depth

Karen Simonyan and Andrew Zisserman

Very Deep Convolutional Networks for Large-Scale Image Recognition.

In *ICLR*, 2015.
Deep ConvNets for image classification

- **GoogLeNet / Inception** 22 layers, 7M parameters

![Inception module diagram]

- **Inception module**

  ![Inception module diagram]

  [Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich]

  Going Deeper with Convolutions.

  In *CVPR*, 2015.
Deep ConvNets for image classification

• **ResNet** 152 layers, 60M parameters

![Diagram showing residual learning](image)

*Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun*

Deep Residual Learning for Image Recognition.

*In CVPR, 2016.*
Deep ConvNets for image classification

Revolution of Depth

152 layers

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

ImageNet Classification top-5 error (%)

3.57
6.7
7.3
11.7
16.4
25.8
28.2

shallow
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
   • **AlexNet**: 2012 the (deep) revolution
     • Archi and learning
     • Ablation study
ImageNet 2012: the (deep) revolution

- 1.2 million labeled images
- 1000 classes
- Mono-class
- TOP5
Architecture of the IMAGENET Challenge 2012 Winner:

A Large CNN [@Fergus NIPS 2013]

Krizhevsky et al. [NIPS2012]

- Same model as LeCun’98 but:
  - Bigger model (8 layers)
  - More data (10^6 vs 10^3 images)
  - GPU implementation (50x speedup over CPU)
  - Better regularization (DropOut)

- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week
Filtering

- Convolutional
  - Dependencies are local
  - Translation equivariance
  - Tied filter weights (few params)
  - Stride 1, 2, ... (faster, less mem.)

Non-Linearity

- Non-linearity
  - Per-feature independent
  - Tanh
  - Sigmoid: $1/(1+\exp(-x))$
  - Rectified linear
    - Simplifies backprop
    - Makes learning faster
    - Avoids saturation issues
  
  → Preferred option

Pooling

- Spatial Pooling
  - Non-overlapping / overlapping regions
  - Sum or max
  - Boureau et al. ICML'10 for theoretical analysis

Normalization

- Contrast normalization (between/across feature maps)
  - Local mean = 0, local std. = 1, “Local” → 7x7 Gaussian
  - Equalizes the features maps
Learning the deep CNN 2012

• Basics:
  • SGD, Backprop
  • Cross Validation
  • Grid search

• “New”
  • Huge computational resources (GPU)
  • Huge training set (1 million images)
  • Data augmentation - Pre-processing
  • Dropout
  • ReLu
  • *Contrast normalization*

• And (dixit LeCun) few hacks to initialize the weights, prevent the weights from blowing up for large values of the learning rate ...
Data Augmentation

lots of jittering, mirroring, and color perturbation of the original images generated on the fly to increase the size of the training set

Crop, flip,.. in train AND in test
Dropout: an efficient way to average many large neural nets

For each training example, randomly omit each hidden unit with probability 0.5
Due to sharing of weights, model strongly regularized
- Pulls the weights towards what other models want.
Better than L2 and L1 regularization that pull weights towards zero

@Hinton, NIPS 2012
Dropout: what do we do at test time?

Option 1:
Sample many different architectures and take the geometric mean of their output distributions

Option 2: (Faster way)
Use all the hidden units
but after halving their outgoing weights

Rq: In case of single hidden layer, this is equivalent to the geometric mean of the predictions of all models
For multiple layers, it’s a pretty good approximation and it’s fast
How well does dropout work?

Significantly improve generalization:
- For very deep nets, or at least when there are huge fully connected layers (eg. AlexNet first FC layer, VGG next, ...)
- Less useful for fully convolutional nets

Useful to prevent feature co-adaptation (feature only helpful when other specific features present)

Later in course (week 11)
⇒ Dropout as a Bayesian Approximation
⇒ Representing Model Uncertainty in Deep Learning
Outline

1. Recap MLP
2. Convolutional Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
   • AlexNet: 2012 the (deep) revolution
     • Archi and learning
     • Ablation study
       • Number of layers
       • Tapping off features at each layer
       • Transfo Robustness vs layers
Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR’09]
- 18.2% top-5 error
- Our reimplementation: 18.1% top-5 error
Architecture of Krizhevsky et al.

- Remove top fully connected layer
  - Layer 7

- Drop 16 million parameters

- Only 1.1% drop in performance!
Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance
Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6, 7

- Now only 4 layers

- 33.5% drop in performance

→ Depth of network is key
Translation (Vertical)
Scale Invariance

Layer 1

Layer 7

Output
Rotation Invariance

Layer 1

Output

Layer 7

Rotation Degrees

Canonical Distance

Lawn Mower
Shih–Tzu
African Crocodile
African Grey
Entertainment Center

Rotation Degrees

Canonical Distance

Lawn Mower
Shih–Tzu
African Crocodile
African Grey
Entertainment Center
What’s next AlexNet?

How to improve AlexNet architecture?
+++Deep?
+++Convolutional?
+++Fully connected?
All?
⇒A lot of empirical studies
  ⇒Tuning various design parameters
  ⇒what really works?
⇒Winners: VGG, GoogleNets, ResNet
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
   • AlexNet
   • GoogLeNet
GoogLeNet (2014)

Winner of ILSVRC -2014. Very deep network with 22 layers:
• Network–in-network–in-network
• Removed fully connected layers → small # of parameters (5M weights)
GoogLeNet (2014)

Inception layer
GoogLeNet (2014)

Auxiliary classifiers

Main classifier
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
   • AlexNet
   • GoogleNet
   • VGG
VGG Net: Archi post-2012 revolution

K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
VGG Net

Basic Idea: Investigate the **effect of depth** in large scale image recognition

- **Fix other parameters** of architecture, and steadily increase depth
Fixed configuration:

- Convolutional Layers: from 8 to 16
- Fully Connected Layers: 3
- Stride: 1
- ReLu: Follow all hidden layers
- Max-Pooling: 2x2 window
- Padding: s/t spatial resolution is preserved
- #Convolutional filters: Starting from 64, double after each max-pooling layer until 512
- Filter sizes: 3x3 and 1x1
### ConvNet Configuration

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 weight layers</td>
<td>11 weight layers</td>
<td>13 weight layers</td>
<td>16 weight layers</td>
<td>16 weight layers</td>
<td>19 weight layers</td>
<td></td>
</tr>
</tbody>
</table>

**input (224 × 224 RGB image)**
- conv3-64
- conv3-64
- **LRN**
- conv3-64
- conv3-64

**maxpool**
- conv3-128
- conv3-128
- **conv3-128**

**maxpool**
- conv3-256
- conv3-256
- **conv3-256**

**maxpool**
- conv3-512
- conv3-512
- **conv1-512**

**maxpool**
- conv3-512
- conv3-512
- **conv3-512**

**maxpool**
- FC-4096
- FC-4096
- FC-1000
- soft-max

**Diagram:**
- 3x3 conv, 64
- 3x3 conv, 64, pool/2
- 3x3 conv, 128
- 3x3 conv, 128, pool/2
- 3x3 conv, 256
- 3x3 conv, 256, pool/2
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- fc, 4096
- fc, 4096
- fc, 1000

*Table credit: Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR2015*
VGG Net

Results:
• First place in localization (25.3% error), second in classification (7.3% error) in ILSVRC 2014 using ensemble of 7 networks
• Outperforms Szegedy et.al (GoogLeNet) in terms of single network classification accuracy (7.1% vs 7.9%)
Observations with VGG testing:

- Deepnets with small filters outperform shallow networks with large filters
  - Shallow version of B: 2 layers of 3x3 replaced with single 5x5 performs worse
- LRN does not improve performance
- Classification error decreases with increases ConvNet depth
- Important to capture more spatial context (config D vs C)
- Error rate saturated at 19 layers
- Scale jittering at training helps capturing multiscale statistics and leads to better performance
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Modern Deep Architectures
   • The big picture
   • GoogleNet
   • VGG
   • ResNet