Deep Learning for Visual Recognition
Visual Recognition

75% of our information comes through our eyes.

Let’s teach machines “how to see”!

Sensing (e.g., Camera)

Understanding (Machine Learning)
Sensing
Sensing
Understanding

\[ f : x \mapsto y \]

Machine learning models as function approximators
Digital images are already described by numbers (0~255), but raw pixel values are sensitive to view-point variation, scale variation, deformation, illumination change, and many other factors.
Before the Era of Deep Learning

① Describing images by hand-crafted features

② Learning a function on the feature space
Before the Era of Deep Learning

An example of the hand-crafted features: *Histogram of Oriented Gradients* (HOG)

*Input image*  
*Image gradient*  
*Computing histograms of orientations by soft voting per pixel*

*rectangular HOG blocks: represented by three parameters: the number of cells per block, the number of pixels per cell, and the number of channels (9) per cell histogram*
Deep Learning for Visual Recognition

A unified framework based on deep neural networks

*Image representation* and *recognition* within a *single network architecture*

- Input image
- Feature $x$
- $g(x)$

**Image representation**

**Function for decision making**

**Convolutional Neural Network (CNN) feature maps**
Recognition Tasks of Our Interest in This Course

- Image Classification
- Object Detection
- Semantic Segmentation
Training Perceptron

- Gradient computation by *error backpropagation*

\[
\begin{align*}
    z &= b + \sum_i w_i x_i \\
    y &= \frac{1}{1 + e^{-z}} \\
    L &= \frac{1}{2} \sum_n (\hat{y}^n - y^n)^2
\end{align*}
\]

\[
\begin{align*}
    \frac{\partial z}{\partial w_i} &= x_i \\
    \frac{\partial y}{\partial z} &= y(1 - y) \\
    \frac{\partial L}{\partial y^n} &= -(\hat{y}^n - y^n) \\
    \frac{\partial L}{\partial w_i} &= \sum_n \frac{\partial y^n}{\partial w_i} \frac{\partial L}{\partial y^n} = \sum_n \frac{\partial z^n}{\partial w_i} \frac{\partial y^n}{\partial z^n} \frac{\partial L}{\partial y^n} = -\sum_n x_i^n y_n (1 - y^n)(\hat{y}^n - y^n)
\end{align*}
\]

$n$: number of data
Multi-Layer Perceptron (MLP)

- Stacking layers of multiple perceptrons
- Advantages
  - Nonlinear classification
  - Achieving better performance
Multi-Layer Perceptron (MLP)

Single Perceptron

Multi-layer perceptron

non-linear decision boundary
Error Backpropagation in MLP

\[
\frac{\partial L}{\partial z_j} = \frac{dy_j}{dz_j} \frac{\partial L}{\partial y_j}
\]

\[
\frac{\partial L}{\partial y_i} = \sum_j \frac{dz_j}{dy_i} \frac{\partial L}{\partial z_j} = \sum_j w_{ij} \frac{\partial L}{\partial z_j} = \sum_j w_{ij} \frac{dy_j}{dz_j} \frac{\partial L}{\partial y_j}
\]

\[
\frac{\partial L}{\partial w_{ki}} = \sum_n \frac{\partial z_i^n}{\partial w_{ki}} \frac{dy_i^n}{dz_i^n} \frac{\partial L}{\partial y_i^n} = \sum_n \frac{\partial z_i^n}{\partial w_{ki}} \frac{dy_i^n}{dz_i^n} \sum_j w_{ij} \frac{dy_j^n}{dz_j^n} \frac{\partial L}{\partial y_j^n}
\]

*error signal from previous layer

j: number of nodes in previous layer; n: number of training data

update \( w_{ki} \)
Issues in MLPs

• Overfitting
  • Learned function may be too much optimized to be generalized.
    • Solution: Training with a large number of data and regularization techniques (e.g., dropout)

• Large amount of training time
  • Computing gradients could take too much time due to the large number of parameters.
    • Solution: Using GPUs that enable massively parallel computation
  • Vanishing gradient problem when using the sigmoid activation function

  \[
  \frac{\partial E}{\partial w_{ki}} = \sum_n \frac{\partial z^n_i}{\partial w_{ki}} \frac{\partial y^n_i}{\partial z^n_i} \frac{\partial E}{\partial y^n_i} = \sum_n \frac{\partial z^n_i}{\partial w_{ki}} \sum_j w_{ij} \frac{\partial y^n_j}{\partial z^n_j} \frac{\partial E}{\partial y^n_j}
  \]

  • Gradients in the lower layers are typically extremely small.
  • Optimizing multi-layer networks takes huge amount of time.
  • Solution: Using ReLU or its variants instead of sigmoid
Widely Used Loss Functions

- Cross entropy loss (for classification in general)
  - Entropy of probability distribution $P$
    \[
    H(P) = E_P[- \log P] = - \sum_X P(X) \log P(X)
    \]
  - Cross entropy between two probability distributions $P$ and $Q$
    \[
    H(P, Q) = E_P[- \log Q] = - \sum_X P(X) \log Q(X)
    \]
  - Cross entropy loss
    - $P$: Groundtruth label distribution
    - $Q$: Predicted probability distribution *for multi-class classification—> softmax + cross-entropy

\[
L(y, i) = - \log \left( \frac{\exp(y_i)}{\sum_{j=1}^{C} \exp(y_j)} \right) = -y_i + \log \left( \sum_{j=1}^{C} \exp(y_j) \right)
\]
Widely Used Loss Functions

• Multi-label soft-margin loss
  • Multi-class **multi-label** classification
  • Sum of binary classification losses, each defined per class

\[
L(y, \hat{y}) = -\frac{1}{nC} \sum_{d=1}^{n} \sum_{c=1}^{C} \left\{ \hat{y}_c \log \frac{\exp(y_c)}{1 + \exp(y_c)} + (1 - \hat{y}_c) \log \frac{1}{1 + \exp(y_c)} \right\}
\]

*multi-label classification --> sigmoid [pos, neg] + cross-entropy

• Mean squared error *(for regression)*

\[
L(y, \hat{y}) = \frac{1}{n} \sum_{d=1}^{n} (y_d - \hat{y}_d)^2
\]
Stochastic Gradient Descent (SGD)

• Update weights for each sample

\[ L^n = \frac{1}{2} (y^n - \hat{y}^n)^2, \quad w_i(t + 1) = w_i(t) - \epsilon \frac{\partial L^n}{\partial w_i} \] 

(+) Fast and online
(−) Sensitive to noise

• Minibatch SGD: Update weights for a small set of samples

\[ L^B = \frac{1}{2} \sum_{n \in B} (y^n - \hat{y}^n)^2, w_i(t + 1) = w_i(t) - \epsilon \frac{\partial L^B}{\partial w_i} \] 

(+) Fast and semi-online
(+ Robust against noise

• Most optimization techniques used for deep learning are based on SGD.
Convolutional Neural Networks for Image Classification
MLPs for Visual Data?

- MLPs do not scale well to visual data like image and video.

  - Requiring a huge number of weight parameters
  - Easily over-fitted and wasteful

Solution? *Using convolution kernels* instead of the fully-connected weights!
Convolution

\[
g(x, y) = f(x, y) \ast h(u, v)
\]

where

\[
g(x, y) = \sum_{u,v} f(x + u, y + v) h(u, v)
\]

**Input**

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

**Kernel**

\[
\begin{array}{cc}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

**Output**

\[
\begin{array}{cccc}
1 & 4 & 3 & 4 \\
1 & 2 & 4 & 3 \\
1 & 2 & 3 & 4 \\
1 & 3 & 3 & 1 \\
3 & 3 & 1 & 1 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
Examples of Convolution in Image Processing
Convolutional Neuron vs. Perceptron

• Convolutional neuron is a generalization of the perceptron.

When the kernel is smaller than the input:

When the kernel size is the same with that of the input (identical to perceptron):
Convolutional Neural Network (CNN)

Deep neural network with

**Convolutional layers** + Pooling operations + MLP (Fully-connected layers)

- **224 × 224 × 3** image
- **3 × 3 × 3** filter
- **222 × 222 × 64** activation map
- Applying ReLU per activation

Deep neural network with **Convolutional layers** + Pooling operations + MLP (Fully-connected layers)
Convolutional Neural Network (CNN)

Deep neural network with
Convolutional layers + **Pooling operations** + MLP (Fully-connected layers)

- **Pooling operations**
  - Max pooling
  - Average pooling
  - $L_2$-norm pooling
  - And many others ...

- **Why pooling?**
  - To achieve spatial invariance
  - To abstract image information
  - To reduce the number of parameters and memory usage

Example: Max pooling
Convolutional Neural Network (CNN)

Deep neural network with
Convolutional layers + Pooling operations + MLP (Fully-connected layers)

A typical CNN architecture

- **Conv layer** with 64 kernels
- **Conv layer** with 64 kernels
- **Pooling**
- **Conv layer** with 128 kernels
- **Conv layer** with 128 kernels
- **Pooling**
- **Conv layer** with 256 kernels
- **Conv layer** with 256 kernels
- **Pooling**
- **Pooling**
- **FC layer**
- **FC layer**
- **FC layer**
- **Softmax**

Increasing the number of channels while decreasing the resolution of activation maps
A typical CNN architecture:

- **Conv layer**: 64 kernels
- **Conv layer**: 64 kernels
- **Pooling**
- **Conv layer**: 128 kernels
- **Conv layer**: 128 kernels
- **Pooling**
- **Conv layer**: 256 kernels
- **Conv layer**: 256 kernels
- **Pooling**

... (repeated structure)

**FC layer**: 64
**FC layer**: 128
**Softmax**

The final activation map is converted into a vector by global pooling or concatenation.
A typical CNN architecture with Convolutional layers + Pooling operations + MLP (Fully-connected layers).

Top-9 patches activating each kernel:
- Lower layers capture edges and blobs while upper layers detect more abstract features like textures and shapes.
• SGD with error backpropagation

Image courtesy: Oxford VGG (http://www.robots.ox.ac.uk/~vgg/practicals/cnn/)
Image Classification Tasks

Scene recognition

Image Classification Tasks

Scene recognition

Place2 benchmark (http://places2.csail.mit.edu/)
Image Classification Tasks

Object classification

Classification CNN: Backbone for Many Other Tasks

- **Backbone Network** (image representation)
  - **CNN**
  - **Feature Map**
  - **Classification CNN**
    - **Image-level classification**
      - **Person**
      - **Bike**
    - **Classification + Regression**
      - \{\text{Person, Bike}\}
    - **Dense pixel-level classification**

Convolutional Neural Networks for Object Detection
Object Detection: A Naïve Approach

- Motivated by the great success of deep learning in image classification

Object Detection = Box localization + Box classification

(region proposal based on computing hierarchical grouping of similar regions from over-segmented images (hierarchical clustering))

Object proposals from image to correct bounding box and edge labeling

(left) Selective Search for Object Detection, IJCV 2013

(right) Edge Boxes: Locating Object Proposals from Edges, ECCV 2014
Region-based CNN* (R-CNN)

• Summary
  • Independent evaluation of each proposal
  • Bounding box regression improves detection accuracy.
  • Mean average precision (mAP): 53.7% with bounding box regression in VOC 2010 test set

*Girshick et al., Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014
Region-based CNN (R-CNN)

- Learning a transformation of bounding box
  - Region proposal: \( P = (P_x, P_y, P_w, P_h) \)
  - Ground-truth: \( G = (G_x, G_y, G_w, G_h) \)
  - Transformation: \( d(P) = (d_x, d_y, d_w, d_h) \)

  \[
  \begin{align*}
  \hat{G}_x &= P_w d_x(P) + P_x \\
  \hat{G}_y &= P_h d_y(P) + P_y \\
  \hat{G}_w &= P_w \exp(d_w(P)) \\
  \hat{G}_h &= P_h \exp(d_h(P))
  \end{align*}
  \]

  Approximate \( d_i(P) \) by \( w_i^T \phi_5(P) \).

  CNN feature on top

\[
 w_i^* = \arg\min_{w_i} \sum_{k=1}^{N} \left( d_i^k - w_i^T \phi_5(P^k) \right)^2 + \lambda \|w_i\|^2
\]
Fast R-CNN*

- A fast version of R-CNN
  - 9x faster in training and 213x faster in testing than R-CNN
  - A single feature computation and ROI pooling using object proposals
  - Bounding box regression into network
  - Single stage training using multi-task loss

*Girshick, Fast R-CNN, ICCV 2015
Faster R-CNN*

- Fast R-CNN + Region Proposal Network
  - Proposal computation into network
  - Marginal cost of proposals: 10ms

*Ren et al., Faster R-CNN, NIPS 2015
• Details of the region proposal network
  • 9 anchors per location (3 aspect ratios x 3 scales)

• Groundtruth label per anchor

\[ p^* = f(x) = \begin{cases} 
-1, & \text{if IoU < 0.3,} \\
1, & \text{if IoU > 0.7,} \\
0, & \text{otherwise.} 
\end{cases} \]

where IoU is intersection over union:

\[ \text{IoU} = \frac{\text{Anchor } \cap \text{ GTBox}}{\text{Anchor } \cup \text{ GTBox}} \]

• Trained with a binary classification loss for anchor selection and a regression loss for box refinement

\[ p_i: \text{the predicted probability of anchor } i \text{ being an object (} p^* \text{ ground truth)} \]

\( w_a, h_a: \text{anchor's width, height}
\)

\( x_a, y_a: \text{anchor's center} \)

@vmirly

*9 anchors
Object Detection Performance

• R-CNN family achieves the state-of-the-art performance in object detection.

• Stronger backbone network, better detection performance.

Pascal VOC 2007 Object Detection mAP (%)
Faster R-CNN with ResNet
Faster R-CNN with ResNet
Convolutional Neural Networks for Semantic Segmentation
Semantic Segmentation

- Grouping pixels based on their semantics (i.e., class labels)
• Image segmentation vs. semantic segmentation

In *image segmentation*, the two parts indicated by green circles belong to different segments since they have different colors and textures.

In *semantic segmentation*, they should be in the same segment since they come from the same semantic entity “*person*”.
• Object proposal
  • Detecting (candidate) boxes likely to enclose objects

(Left) Edge Boxes: Locating Object Proposals from Edges, ECCV 2014

(Right) Selective Search for Object Detection, IJCV 2013
Early Approaches

• Early approaches to semantic segmentation

**Semantic Segmentation =**

Mask localization + Mask classification

- Object proposals
- CNN for image classification

• Some object proposals are based on segmentation (e.g., selective search) thus readily provide object candidate masks.

• Semantic segmentation can be easily achieved by classifying segmentation proposals with a CNN.
Early Approaches

• Examples
  • R-CNN* and Simultaneous detection and segmentation**

• Limitations
  • Their performance is bounded by region-proposal accuracy, which is not satisfactory since proposal techniques are unsupervised and take only low-level image features (e.g., colors and textures) into account.

*Girshick et al., Region-Based Convolutional Networks for Accurate Object Detection and Segmentation, TPAMI 2016
**Hariharan et al., Simultaneous Detection and Segmentation, ECCV 2014
Recent Approaches

- **End-to-end CNN architectures**
- Categorized roughly into two classes
  - **Fully Convolutional Networks** (FCNs)
    - FCN
    - DeepLab
  - **Convolutional Encoder-Decoders**
    - U-Net
    - Deconvolution network

Noh et al., Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

Long et al., Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
Semantic segmentation = Pixel-level classification

Trade-off between the resolution and the semantic level of prediction
- Classification demands high-level semantic features and large receptive fields, which are typically achieved by several pooling layers.

However, this approach eventually decreases the resolution of convolutional feature maps and that of semantic segmentation results accordingly.

How to achieve both of them at the same time?

A larger receptive field is required for more robust high-level reasoning.
The first end-to-end architecture for semantic segmentation
Interpreting fully connected layers of classification nets as 1x1 convolutions

Fully Convolutional Networks (FCN)*

*Long et al., Fully Convolutional Networks for Semantic Segmentation, CVPR 2015
Fully Convolutional Networks (FCN)

- A fully-connected layer and its 1x1 convolution interpretation
  - The output of the network becomes a tensor with spatial information.
  - The output can be interpreted as class scores over local image regions.

A Fully Connected Layer

Classifying a single feature vector

\[ X \rightarrow W \rightarrow W^TX \]

A 1 × 1 Convolution Layer

Classifying every feature vector of the convolutional feature map

\[ X \rightarrow W \rightarrow W^TX \]
Fully Convolutional Networks (FCN)

• Fully Convolutional
  • No fully connected layers, only convolutional layers
  • Able to handle images of any arbitrary sizes and aspect ratios

• Limitation: Predicted score map in a very low-resolution

• For enlarging the score map
  • Adding a simple bilinear interpolation on the top of the network

Bilinear interpolation

*Interpolation is done first along the y direction and then along the x direction

trainable upscaling
Fully Convolutional Networks (FCN)

- For enlarging the score map (cont’d)
  - Adding skip-connections

This approach integrates activations from lower layers into prediction so that it preserves higher spatial resolution and capture lower-level semantics at the same time.
Fully Convolutional Networks (FCN)

- Experimental results

<table>
<thead>
<tr>
<th>Input image</th>
<th>GT</th>
<th>FCN-32s</th>
<th>FCN-16s</th>
<th>FCN-8s</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="image" /></td>
<td><img src="image2.jpg" alt="image" /></td>
<td><img src="image3.jpg" alt="image" /></td>
<td><img src="image4.jpg" alt="image" /></td>
<td><img src="image5.jpg" alt="image" /></td>
</tr>
<tr>
<td><img src="image6.jpg" alt="image" /></td>
<td><img src="image7.jpg" alt="image" /></td>
<td><img src="image8.jpg" alt="image" /></td>
<td><img src="image9.jpg" alt="image" /></td>
<td><img src="image10.jpg" alt="image" /></td>
</tr>
<tr>
<td><img src="image11.jpg" alt="image" /></td>
<td><img src="image12.jpg" alt="image" /></td>
<td><img src="image13.jpg" alt="image" /></td>
<td><img src="image14.jpg" alt="image" /></td>
<td><img src="image15.jpg" alt="image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>mean IU VOC2011 test</th>
<th>mean IU VOC2012 test</th>
<th>inference time</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN [12]</td>
<td>47.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDS [17]</td>
<td>52.6</td>
<td>51.6</td>
<td>~ 50 s</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>62.7</td>
<td>62.2</td>
<td>~ 175 ms</td>
</tr>
</tbody>
</table>

Faster and more accurate than the previous approaches based on CNNs

- Faster
  - The end-to-end architecture that does not rely on off-the-shelf proposals

- More accurate
  - Not bounded by quality of proposals
  - Feature representation and decision maker that are jointly optimized
Deconvolution Network*

- Overall architecture: *Convolutional encoder-decoder*
  - A convolutional encoder
    - A series of convolution and max-pooling layers
    - A common architecture for image classification
  - A decoder with deconvolution layers
    - A series of deconvolution and un-pooling layers
    - A mirrored version of the convolutional encoder

*Noh et al., Learning Deconvolution Network for Semantic Segmentation, ICCV 2015*
• Overall architecture: *Convolutional encoder-decoder* (cont’d)
  • Paired pooling and unpooling layers
    • An unpooling layer is associated with a pooling layer.
    • A pair of pooling and unpooling layers share the pooling switch.

• Advantages of the deep deconvolution decoder
  • Learned to recover fine shapes in the original image resolution
  • End-to-end trainable
Deconvolution Network

- Two key components: Unpooling and deconvolution

*switches record the locations of the local maximum

- Learned filters in deconvolutional layers correspond to bases to reconstruct shape of an input object
Deconvolution Network

- Deconvolution is also known as *transposed convolution*.
- The convolution mask is transposed, weighted, and attached.

![Deconvolution Network Diagram](image_url)

Image courtesy by Vincent Dumoulin
Deconvolution Network

- How the unpooling and deconvolution work
  - Activation maps are **coarsely upsampled** by unpooling.
  - The coarse activation maps are **densified** by deconvolution.
Deconvolution Network

- Qualitative results

*ensemble with FCN-8s*