Machine Learning
EE514 – CS535

Introduction to Deep Learning and Convolutional Neural Networks

Zubair Khalid
School of Science and Engineering
Lahore University of Management Sciences

https://www.zubairkhalid.org/ee514_2022.html
Outline

- Deep Learning Overview
- Convolutional Neural Networks
Deep Learning (DL)

Overview:

- We have already studied deep learning
- Deep Learning = Deep Neural Network
  - Using a neural network with several layers of nodes
- Deep: high number of hidden layers

Deep neural network – generalize very well as they are capable of learning the true underlying features.
Deep Learning (DL)

Difference between ML and DL:

- High number of layers in deep neural network enables
  - feature identification
  - processing in a series of stages

Multi-layer networks have been around but what has changed recently?
Deep Learning (DL)

**Difference between ML and DL:**

Now we have more
- Data; deep learning needs more data
- Computing power (availability of GPUs, parallel processing)
- New tricks to learn the weights of the network
**Deep Learning (DL)**

**New way to train Deep Neural Networks:**

We train layers of the network sequentially.

First, train this layer  
Then this layer  
Then this layer  
Then this layer  
Then this layer

We train each of the non-output layer to act as an **autoencoder**.
Deep Learning (DL)

AutoEncoders:
- A type of neural network that is used to learn data encodings (unsupervised).
- In general, autoencoder has three parts;
  - Encoder
  - bottleneck (code, latent representation)
  - Decoder

- A simple example:
  An auto-encoder (one hidden layer network) is trained to reproduce the input using standard learning algorithm.

Idea:
- Learn a lower-dimensional representation (encoding) for a higher-dimensional data.
- Capture the most important parts of the input.

In other words, training autoencoder forces the ‘hidden layer’ units to become good feature detectors.
**Denoising AutoEncoders:**

**Idea:** The denoising autoencoder gets rid of noise by learning a representation of the input where the noise can be filtered out easily.
Deep Learning (DL)

Overview:

- This is the overall idea!
- There are many types of deep neural networks, different architectures, different types of autoencoder, and different training algorithms
- Fast growing research in the area!
Convolutional Neural Networks (CNNs)

Overview:

Motivation:
Consider an object detection (classification) problem from images using neural network.

For example: CIFAR-10 dataset
- 10 classes, Input image is 32x32x3 = 3072

Fully connected neural network
- Treats input as a vector
- Each neuron in the first layer will have 3072 weights

For 400x400x3 image, each neuron has 480,000 weights

Very large number of parameters!

Why? Regular Neural Network treats input as a vector

Solution: Exploit the structure in the input data
Convolutional Neural Networks (CNNs)

Overview:

- Convolutional Neural Networks exploits the structure in the input, that is, a fact that the input consists of images.

- Instead of treating image as an input vector and each layer as a column of neurons, we
  - take image as an input
  - arrange neurons in 3 dimensions: width, height and depth in each layer

- Each layer transforms an input volume (3D) to an output 3D volume.
Convolutional Neural Networks (CNNs)

Overview:

Regular Neural Network

Convolutional Neural Network (CNN)

In CNNs, the structure of image is exploited, and each layer transforms a volume of activations to an output volume through differentiable function that may or may not have parameters.

In CNN, we use three main types of layers to build network architecture:
- Convolutional layer
- Pooling layer
- Fully-connected layer
Convolutional Neural Networks (CNNs)

Convolutional Layer:

- Convolution Operation:
Convolutional Neural Networks (CNNs)

Convolutional Layer:

- Convolution in 2D:

Convolution leverages three important ideas that can help improve a machine learning system:

- Sparse interactions
  - Kernel smaller than the input

- Parameter sharing

- Equivariant Representations
  - Equivariance to translations
Convolutional Neural Networks (CNNs)

**Convolutional Layer:**

- Convolutional layer parameters consists of a set of learnable filters.

- Intuitively, network learn filters that activate when they see some type of visual feature e.g.,
  - an edge of some orientation or boundary of the shape on the first layer
  - wheel like patterns on higher layers of network

- Each filter in a set of filters produces a separate 2-dimensional activation map.

- These 2D maps are stacked along the depth dimension to produce output volume.
Convolutional Neural Networks (CNNs)

Convolutional Layer:

- Instead of connecting each neuron to all the neurons in the previous volume, CNN connects the neuron to a local region in the input volume controlled by hyperparameter referred to as receptive field (denoted by $F$).

- Extent of this connectivity is always equal to the depth of input volume.
  - Connections are local along height and depth but always full along the depth of input volume.
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Example:

Input: 32x32x3 image
Receptive field: 5x5

Each neuron in the convolutional layer will connect to 5x5x3 region in input volume.

Total weights: 76 = 5x5x3 weights + 1 bias parameter
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Spatial Arrangement of Neurons in the Output Volume:

- 3 hyper-parameters control the arrangement of neurons in the output volume
  - Depth
  - Stride
  - Zero-padding

- Depth (denoted by k):
  - It is equal to the number of filters we want to use.
  - Each filter is assumed to reveal something different in the input.
  - The neurons that are all looking at the same region of the input as a depth column.
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Spatial Arrangement of Neurons in the Output Volume:

- **Stride (denoted by S):**
  - Controls the amount of translation in the convolution operation.
  - **Stride=1:** filter is translated (moved) one pixel when we slide the filter.
  - **Stride=2:** filter is translated (moved) two pixel when we slide the filter.
  - **Stride=2** produces smaller output volume as compared to **stride=1**.
**Convolutional Neural Networks (CNNs)**

**Convolutional Layer:**

**Spatial Arrangement of Neurons in the Output Volume:**

- **Zero-padding (denoted by P):**
  - To handle the convolution along the boundary points, we zero-pad input around the borders. The amount of zero-padding controls the spatial size of the output volume.


Convolutional Neural Networks (CNNs)

**Convolutional Layer:**

**Spatial Arrangement of Neurons in the Output Volume:**

For

- $F$ – receptive field size, $S$ – stride, $P$ – amount of zero padding and $W$ – Input volume size
  
  - Output volume slice size: $1 + (W-F+2P)/S$

- **Example:**
  
  - 7x7 input, 3x3 filter, 0 padding and 1 stride
    
    - $1 + (W-F+2P)/S = 5$
    
    - 5x5 output

- With 2 stride
  
  - $1 + (W-F+2P)/S = 3$
  
  - 3x3 output
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Parameter Sharing

- Number of parameters can be further reduced by parameter sharing.
- Idea:
  - Neurons at each depth slice share the same weights and a bias.
  - At each depth level, we have a 2D slice and we use same parameters for every neuron at each depth level.
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Parameter Sharing - Example

- Input: 227x227x3
- First convolutional layer: \( F=11, S=4, P=0, \text{ depth}=96 \)
- Output slice size: \( 1+(W-F+2P)/S = 55 \)
- Without parameters sharing:
  - Number of parameters per depth slice: \( 55x55x(11x11x3+1) \)
- With parameters sharing:
  - Number of parameters per depth slice: \( 11x11x3+1 \)
Convolutional Neural Networks (CNNs)

Convolutional Layer:

Parameter Sharing

- Q: What is the benefit of parameter sharing?
  - A: 1) Significant reduction in the number of parameters. 
  - 2) Convolutional layer output can be computed by simply convolving filter with an input.

- Each neuron of the depth slice has same parameters which means
  - Shared weights can be interpreted as a filter.
  - The depth slice output is simply a convolution of the filter and the input.

- Parameter sharing is also intuitive because if the filter is detecting an edge at some spatial position, we also want to detect the edge in a similar way at all other positions.
**Convolutional Neural Networks (CNNs)**

**Convolutional Layer:**

**Summary:**

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- 4 Hyper-parameters define the convolutional layer
  - Number of filters, $k$
  - Spatial extent of each filter, $F$
  - Stride, $S$
  - Zero-padding, $P$
- Produces a volume of size $W_2 \times H_2 \times D_2$
  - $W_2 = 1 + (W_1 - F + 2P)/S$
  - $H_2 = 1 + (H_1 - F + 2P)/S$
  - $D_2 = k$ (depth)
- With parameters sharing, the number of parameters are $F \times F \times D_1$ weights and 1 bias per depth slice and $F \times F \times D_1 \times k$ weights and $k$ biases overall
- The $d$-th depth slice output is given by the convolution of $d$-th filter and the input volume.
Convolutional Neural Networks (CNNs)

AlexNet:
Convolutional Neural Networks (CNNs)

**Pooling Layer**:  
- We usually use pooling layer between the convolutional layers in CNNs.  
- The role of pooling layer is to progressively reduce the spatial size of the volume to reduce  
  - the number of parameters  
  - computation time  

  - **Idea**: The pooling layer operates independently on every depth slice of the input and resizes it spatially using the ‘Max’ operation.

- **Example**:

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

max pool with 2x2 filters and stride 2

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```
Convolutional Neural Networks (CNNs)

**Pooling Layer:**
- Pooling layer is defined by two hyper-parameters
  - Spatial extent – $F$
  - Stride – $S$
- In the example, $F=2$ and $S=2$
- Input: a volume of size $W_1 \times H_1 \times D_1$
- Output: a volume of size $W_2 \times H_2 \times D_2$
- $W_2 = 1 + (W_1 - F)/S$
- $H_2 = 1 + (H_1 - F)/S$
- $D_2 = D_1$ (same depth)
- Pooling layer does not have any parameters.
Convolutional Neural Networks (CNNs)

Pooling Layer:
- Instead of Max-Pooling, other pooling techniques are also adopted such as
  - average pooling
  - $L_2$ norm pooling
- These days, research has suggested to use bigger strides at the convolutional layer level instead of frequent pooling layers.
Convolutional Neural Networks (CNNs)

AlexNet:

- **Conv 1**: 227 x 227 x 3 to 55 x 55 x 96 with a 11 x 11 filter, stride 4, followed by a Max Pool layer.
- **Pool 1**: 27 x 27 x 96.
- **Conv 2**: 27 x 27 x 96 to 27 x 27 x 256 with a 3 x 3 filter, stride 2, followed by a Max Pool layer.
- **Pool 2**: 13 x 13 x 256.

- **Conv 3**: 13 x 13 x 256 to 13 x 13 x 384 with a 3 x 3 filter, stride 1, same output.
- **Conv 4**: 13 x 13 x 384 to 13 x 13 x 256 with a 3 x 3 filter, stride 1, same output.
- **Conv 5**: 13 x 13 x 256 to 6 x 6 x 256 with a 3 x 3 filter, stride 2, followed by a Max Pool layer.
- **Pool 5**: 6 x 6 x 256.

- **FC 6**: 4096 layers.
- **FC 7**: 4096 layers.

- **Softmax**: 1000 classes.