This is CS5045!

GCR:ioc7cdl

P.S. These slides are useless if you do not attend classes

Convolution Neural Networks

Neural network with Images

We know it is good to learn a small model.

From this fully connected model, do we really need all the edges?

Can some of these be shared?

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

To capture Spatial/temporal dependencies.

Neural network with Images

Some patterns are much smaller than the whole image.

Neural network with Images

Same pattern appears in different places: They can be compressed!

What about training a lot of such “small” detectors and each detector must “move around”.

Classical Computer Vision

Edge detection

Convert a 2D image into a set of curves

- Extracts salient features of the scene
- More compact than pixels

Origin of Edges

Edges are caused by a variety of factors

Edge detection

How can you tell that a pixel is on an edge?

Profiles of image intensity edges

Edge is Where Change Occurs

Change is measured by derivative in 1D

Biggest change, derivative has maximum magnitude

Or 2nd derivative is zero.

Image gradient

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity

The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Gradient operators

Δ_1

$$\begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$$

Δ_2

$$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

(a)

Δ_1

$$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

Δ_2

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$$

(b)

Δ_1

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Δ_2

$$\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

(c)

Δ_1

$$\begin{matrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{matrix}$$

Δ_2

$$\begin{matrix} 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -3 & -3 & -3 & -3 \end{matrix}$$

(d)

(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$

$\frac{d}{dx}f(x)$

Where is the edge?

Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

Convolution

Convolution is a mathematical way of combining two signals to form a third signal.

Convolution Neural Networks

CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Dot product →

3 -1

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Repeat this for each filter

Two 4 x 4 images
Forming 2 x 4 x 4 matrix

For Grey Images

For Grey Images

1	0.6	0.3	1	1
1	0	0	0.3	1
1	1	0	0.6	1
1	0.6	0	0	0.3
1	1	1	1	1

0	1	0
1	2	1
0	1	0

For Grey Images

For Grey Images

For Grey Images

For Grey Images

For Grey Images

This sliding operation is called **convolution**.*

For Grey Images

filters

...

1	0.6	0.3	1	1
1	0	0	0.3	1
1	1	0	0.6	1
1	0.6	0	0	0.3
1	1	1	1	1

data

results

For Grey Images

CNN

Result array from a convolution operation using a filter is called a feature map.

The number of feature maps is called the number of channels as well. If you use 10 filters, then there will be 10 output channels.

Color image: RGB 3 channels

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Color image

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

CNN with RGB Images

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 + -498 + 164 + 1 = -25$$

Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Convolution v.s. Fully Connected

Fully-connected

So what kind of values should be in the filters?

Before the era of deep learning, the values in the filter were fixed. Have you ever used "sharpen filter" in a photo editor?

The values in a convolution filter are the weights of a Convolutional Neural Network.

filter (old)

0	1	0
1	2	1
0	1	0

fixed values

filter (deep learning)

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

being optimized values

(weights)

fewer parameters!

Fewer parameters

Even fewer parameters

CNN

That means, we put some random values into the filters and set how many filters we use in each layer before starting training, but the values in the filters change as the training goes on.

By doing so, the values in the filters are getting optimized, more meaningful feature maps are being extracted, so the model makes better prediction.

CNN

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image.

Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc.

With added layers, the architecture adapts to the High-Level features as well, giving us a network that has a wholesome understanding of images in the dataset, similar to how we would.

- Convolution layer takes an input feature map of dimension $W \times H \times N$ and produces an output feature map of dimension $\widehat{W} \times \widehat{H} \times M$
- Each layer is defined using following parameters:
  - # Input channels (N)
  - # Output channels (M)
  - Kernel size
  - Padding
  - Stride

Figure: In this example, 5x5 input is convolved with 3x3 kernel with stride=padding=1 to produce an output of size 5x5.

Figure: In this example, 5x5 input is convolved with 3x3 kernel with **stride=2** and padding=1 to produce an output of size 3x3.

Parameters

- For one layer,
- ***i***, no. of input maps (or channels)
- ***f***, filter size (just the length)
- ***o***, no. of output maps (or channels. this is also defined by how many filters are used)
- One filter is applied to every input map
- *num_params*
  = weights + biases
  $= [i \times (f \times f) \times o] + o$

Parameters

Greyscale image with 2×*2* filter, output 3 channels

- $\boldsymbol{i} = 1$ (greyscale has only 1 channel)
- $\boldsymbol{f} = 2$
- $\boldsymbol{o} = 3$
- *num_params*
  $= [\boldsymbol{i} \times (\boldsymbol{f} \times \boldsymbol{f}) \times \boldsymbol{o}] + \boldsymbol{o}$
  $= [1 \times (2 \times 2) \times 3] + 3$
  $= \mathbf{15}$

Filters
Output
w1
w2
w3
w4
+
b1
=
Input
w5
w6
w7
w8
+
b2
=
1 feature map
w9
w10
w11
w12
+
b3
=
3 feature maps

RGB image with 2×2 filter, output of 1 channel

- $\boldsymbol{i}$ = 3 (3 channels)
- $\boldsymbol{f}$ = 2
- $\boldsymbol{o}$ = 1
- *num_params*
  $= [\boldsymbol{i} \times (\boldsymbol{f} \times \boldsymbol{f}) \times \boldsymbol{o}] + \boldsymbol{o}$
  $= [\mathbf{3} \times (\mathbf{2} \times \mathbf{2}) \times \mathbf{1}] + \mathbf{1}$
  $= \mathbf{13}$

Input
Filter
w1
w2
w3
w4
w5
w6
w7
w8
w9
w10
w11
w12
+
+
+
b
=
Output
1 feature map
3 feature maps

Image with 2 channels, with 2×2 filter, and output of 3 channels

- $\boldsymbol{i} = 2$ (2 channels)
- $\boldsymbol{f} = 2$
- $\boldsymbol{o} = 3$
- *num_params*
  $= [\boldsymbol{i} \times (\boldsymbol{f} \times \boldsymbol{f}) \times \boldsymbol{3}] + \boldsymbol{3}$
  $= [\mathbf{2} \times (\mathbf{2} \times \mathbf{2}) \times \boldsymbol{3}] + \boldsymbol{3}$
  $= \mathbf{27}$

Filter
Input
w1
w2
w3
w4
w5
w6
w7
w8
w9
w10
w11
w12
w13
w14
w15
w16
w17
w18
w19
w20
w21
w22
w23
w24
2 feature maps
element-wise addition
+ b1 =
element-wise addition
+ b2 =
element-wise addition
+ b3 =
Output
3 feature maps

Formula For Output Channel

- output = (input + 2p –k)/s+1
- Where k is kernel size
- E.g.
- s = 1; p = 0, k = 3, input = 28
- Output = (28-3)/1 + 1 = 26x26

Problem: In AlexNet, the input image is of size 227x227x3. The first convolutional layer has 96 kernels of size 11x11x3. The stride is 4 and padding is 0. Therefore, the size of the output image right after the first bank of convolutional layers is

$$O = \frac{227 - 11 + 2 \times 0}{4} + 1 = 55$$

So, the output image is of size **55x55x96** (one channel for each kernel)

The whole CNN

Pooling Layers

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature.

Decreases the computational power required to process the data through dimensionality reduction

Extract dominant features.

Max - Min - Avg pooling

Max Pool

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

$\xrightarrow{2 \times 2 \text{ Max-Pool}}$

20	30
112	37

Max Pooling

Max Pooling also performs as a Noise Suppressant.

It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Max Pooling

Pool size
Stride
-4 0 -2 4 1
3 1 0 2 1
1 0 1 1 1
4 6 5 1 0
-1 2 0 0 0
Features
Max Pooling
3 4
6 5
Min Pooling
-4 -2
-1 0
Average Pooling
0 1
2 1
Output

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image

CNN

A CNN compresses a fully connected network in following ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

CNN

Convolutional Neural Networks is extension of traditional Multi-layer Perceptron, based on 3 ideas:

- Local receive fields
- Shared weights
- Spatial / temporal sub-sampling

The whole CNN

Flattening

-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(28,28,1)) )
```

…

…

There are 25 3x3 filters.

Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

```
model2.add(MaxPooling2D((2,2)))
```

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

Convolution

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(28,28,1)) )
```

How many parameters for each filter? 9

25 x 26 x 26

Max Pooling

```
model2.add(MaxPooling2D((2,2)))
```

25 x 13 x 13

Convolution

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225= 25x9

50 x 11 x 11

Max Pooling

```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

Convolution

25 x 26 x 26

Max Pooling

25 x 13 x 13

Convolution

50 x 11 x 11

Max Pooling

50 x 5 x 5

Flattened

```
model2.add(Flatten())
```

1250

Fully connected feedforward network

```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

Output

What is Convolutional NN ?

CNN

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

CNN

Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

AlphaGo

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

AlphaGo's policy network

The following is quotation from their Nature article:

Note: AlphaGo does not use Max Pooling.

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

CNN in speech recognition

CNN in text classification

Source of image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

Conv NN: Detection

Groundtruth:
strawberry
strawberry (2)
strawberry (3)
strawberry (4)
strawberry (5)
strawberry (6)
strawberry (7)
strawberry (8)
strawberry (9)
strawberry (10)
apple
apple (2)
apple (3)

Groundtruth:
tv or monitor
tv or monitor (2)
tv or monitor (3)
person
remote control
remote control (2)

Sermanet, CVPR 2014

Conv NN: Scene parsing

Farabet, PAMI 2013

CNN: indoor semantic labeling RGBD

Figure 2: Some scene labelings using our Multiscale Convolutional Network trained on RGBD images.

Conv NN: Action Detection

Taylor, ECCV 2010

CONVOLUTIONAL NEURAL NETWORKS
IS THERE ANYTHING THEY CAN'T DO?
memegenerator.net

References

Some slides are taklenfrom Dr Atif Tahir’s slides.

https://www.philgineer.com/2021/11/convolutional-neural-network-explained.html

https://github.com/vdumoulin/conv_arithmetic?source=post_page-----3bd2b1164a53----------

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53