11-695: Competitive Engineering
Convolution I

Spring 2018

Outline

CarnegieMellon

Recap: Tensorflow

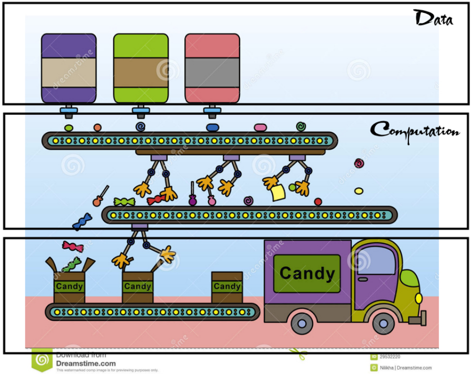
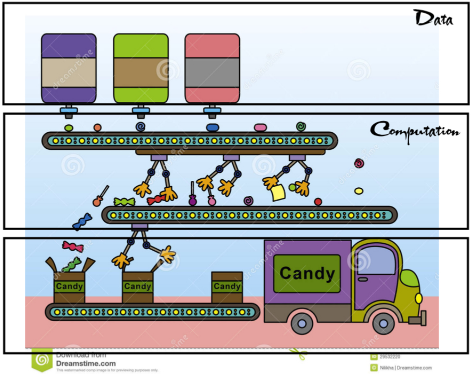


Image credit: Dreamstime.com

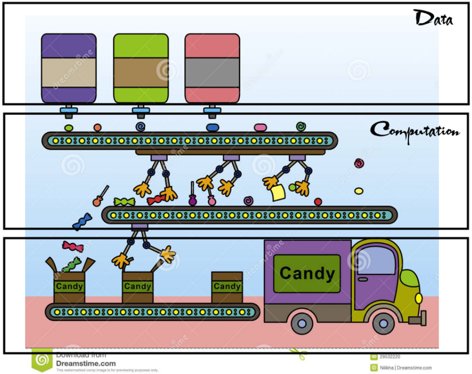
Recap: Tensorflow



- Distinguish between computational graph and data

Image credit: Dreamstime.com

Recap: Tensorflow



- Distinguish between computational graph and data
- Understand the role of a Session

Image credit: Dreamstime.com

Recap: Feed-forward NN - A Motivation

Carnegie Mellon

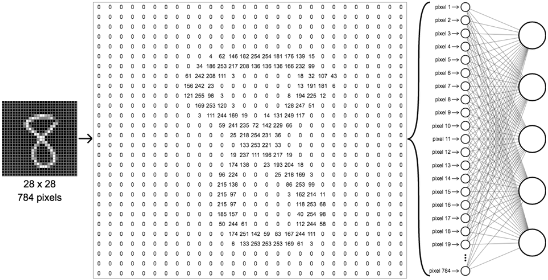
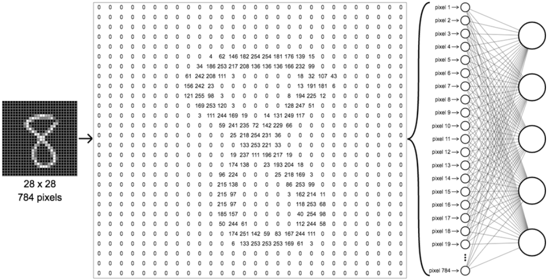


Image credit: https://ml4a.github.io/ml4a/neural_networks/

Recap: Feed-forward NN - A Motivation

CarnegieMellon

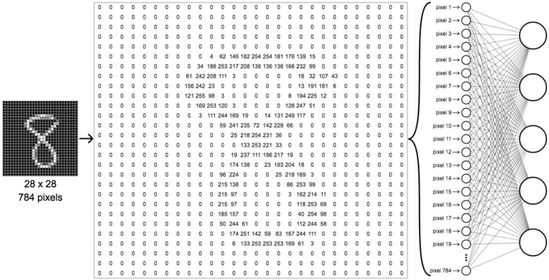


- The first layer of a NN
- Number of params?

Image credit: https://ml4a.github.io/ml4a/neural_networks/

Recap: Feed-forward NN - A Motivation

Carnegie Mellon

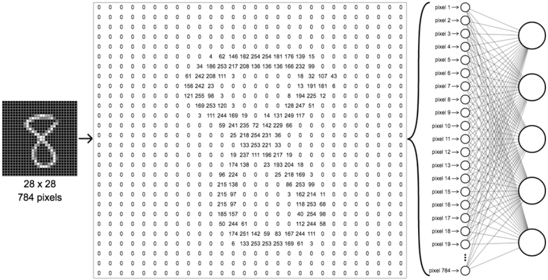


- The first layer of a NN
- Number of params? $28 * 28 * n$

Image credit: https://ml4a.github.io/ml4a/neural_networks/

Recap: Feed-forward NN - A Motivation

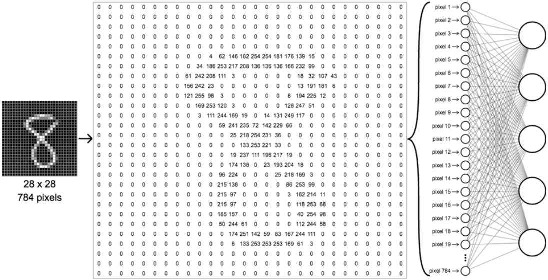
Carnegie Mellon



- The first layer of a NN
- Number of params? $28 * 28 * n$
- Real world: $200 * 200 * 3 * 1000 \approx 1e8$ for the first layer

Image credit: https://ml4a.github.io/ml4a/neural_networks/

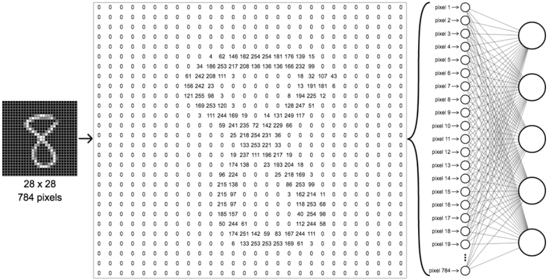
Recap: Feed-forward NN - A Motivation



- The first layer of a NN
- Number of params? $28 * 28 * n$
- Real world: $200 * 200 * 3 * 1000 \approx 1e8$ for the first layer
- What's more?

Image credit: https://ml4a.github.io/ml4a/neural_networks/

Recap: Feed-forward NN - A Motivation



- The first layer of a NN
- Number of params? $28 * 28 * n$
- Real world: $200 * 200 * 3 * 1000 \approx 1e8$ for the first layer
- What's more? How about spatial correlations?

Image credit: https://ml4a.github.io/ml4a/neural_networks/

Another Motivation

- Local spatial features at different locations are the similar

Image credit: Dreamstime.com

Another Motivation

- Local spatial features at different locations are the similar
- Model parts of the image instead of the whole

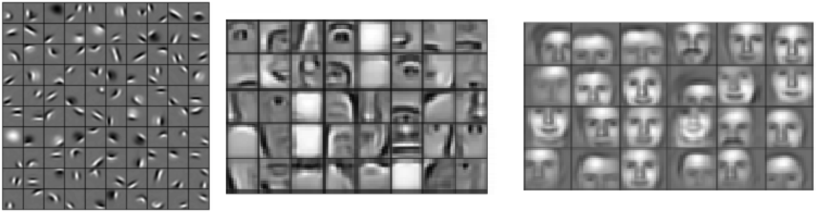
Image credit: Dreamstime.com

Another Motivation

- Local spatial features at different locations are the similar
- Model parts of the image instead of the whole
- Exploit image's redundancy: e.g. edges

Image credit: Dreamstime.com

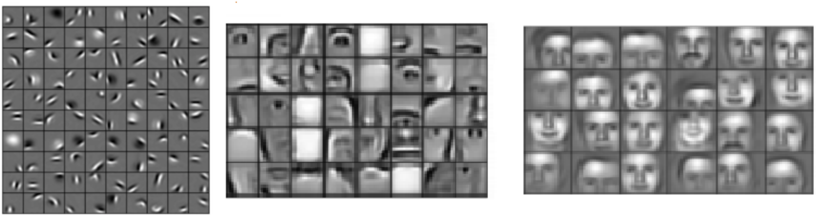
Even More Motivation



- Human cognition works similarly
  - Eyes detect edges
  - Visual cortex uses Gabor-like filter to recognize objects

[1] Bengio J. et al. Representation Learning: A Review and New Perspectives Image credit: Nvidia Developer Blog

Even More Motivation



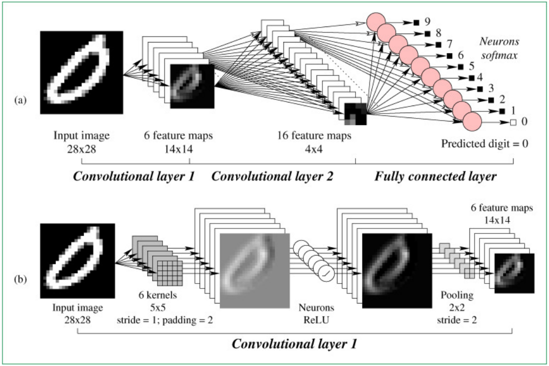
- Human cognition works similarly
  - Eyes detect edges
  - Visual cortex uses Gabor-like filter to recognize objects
- Intention to build hierarchical abstract representation[1]

[1] Bengio J. et al. Representation Learning: A Review and New Perspectives

Image credit: Nvidia Developer Blog

Convolutional Neural Network

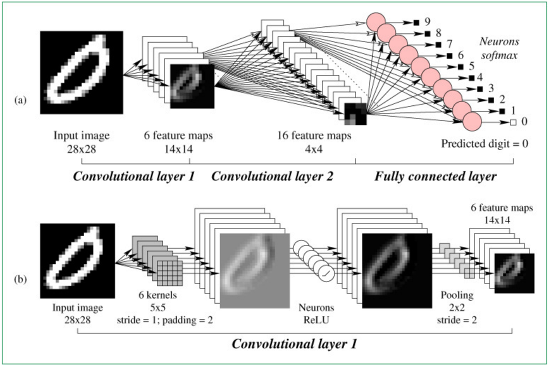
Carnegie Mellon



- Yet another type of feed-forward NN

Image credit: ScienceDirect.com

Convolutional Neural Network

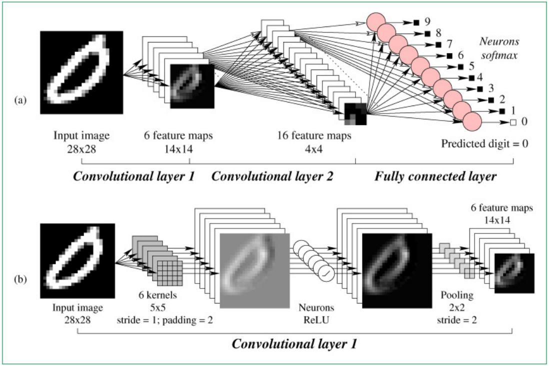


- Yet another type of feed-forward NN
- With additions of:
  - Convolutional Layers
  - Pooling Layers

Image credit: ScienceDirect.com

Convolutional Neural Network

CarnegieMellon

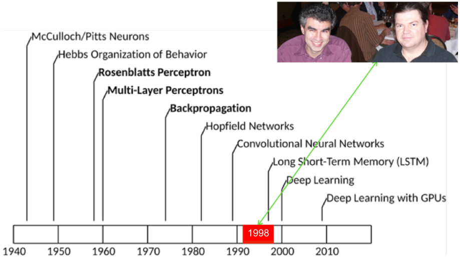


- Yet another type of feed-forward NN
- With additions of:
  - Convolutional Layers
  - Pooling Layers
- Renaming MLPs into the so-call Fully Connected

Image credit: ScienceDirect.com

Recap: History of (C)NN

Carnegie Mellon

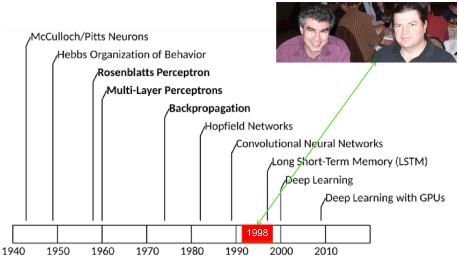


- Be a go-to feature extraction solution for images/videos

[2] Yin W. et al Comparative Study of CNN and RNN for NLP

Image credit: IBM and Recode.net

Recap: History of (C)NN

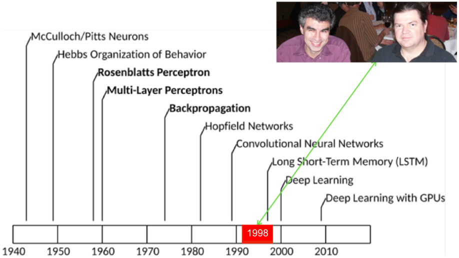


- Be a go-to feature extraction solution for images/videos
- Even outperform over the advantages of RNN in some language tasks ?!! [2]

[2] Yin W. et al Comparative Study of CNN and RNN for NLP

Image credit: IBM and Recode.net

Recap: History of (C)NN



- Be a go-to feature extraction solution for images/videos
- Even outperform over the advantages of RNN in some language tasks ?!! [2]
- Be an essential part in many SoTA solutions for classifications, detections, recognition, segmentation, OCR, motion, pose, etc.

[2] Yin W. et al Comparative Study of CNN and RNN for NLP

Image credit: IBM and Recode.net

Outline

1. Recap and motivation
2. Convolution
3. Convolution in Neural Network
4. Pooling
5. Case Studies

Convolution

Carnegie Mellon

- Yet another mathematics operation ▸ demo

$$(f * g)(t) = \int_{-\infty}^{\infty} f(u)g(t-u)du = \int_{-\infty}^{\infty} f(t-u)g(u)du$$

Convolution

CarnegieMellon

- Yet another mathematics operation demo

$$(f * g)(t) = \int_{-\infty}^{\infty} f(u)g(t-u)du = \int_{-\infty}^{\infty} f(t-u)g(u)du$$

- A linear time invariant (LTI) operation, means that no new frequency components are created, and so output is the pointwise product of input and a transfer function (Wikipedia)

Convolution

CarnegieMellon

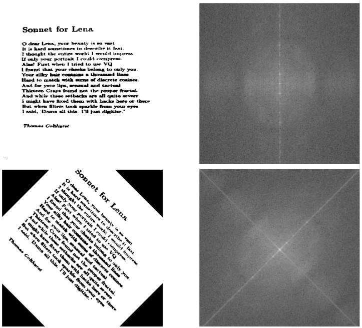
- Yet another mathematics operation demo

$$(f * g)(t) = \int_{-\infty}^{\infty} f(u)g(t-u)du = \int_{-\infty}^{\infty} f(t-u)g(u)du$$

- A linear time invariant (LTI) operation, means that no new frequency components are created, and so output is the pointwise product of input and a transfer function (Wikipedia)
- Relation to Fourier Transformation demo

$$(f * g)(t) = \mathcal{F}^{-1}(\sqrt{2\pi} \quad \mathcal{F}|f| \cdot \mathcal{F}|g|)$$

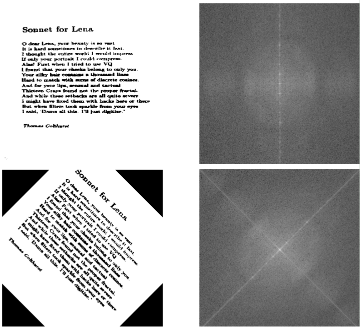
Convolution (cont'd)



- Convolution is an operation in Fourier domain

Image credit: Timdettmers.com

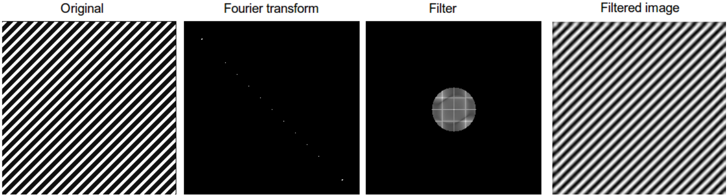
Convolution (cont'd)



- Convolution is an operation in Fourier domain
- It can capture orientation change

Image credit: Timdettmers.com

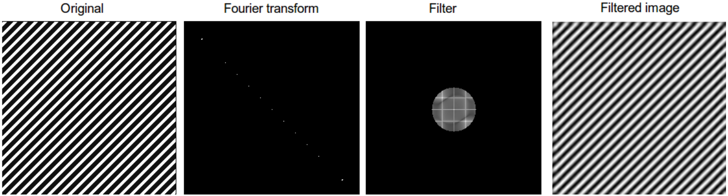
Convolution (cont'd)



- Convolution is often referred to as filtering

Image credit: Timdettmers.com

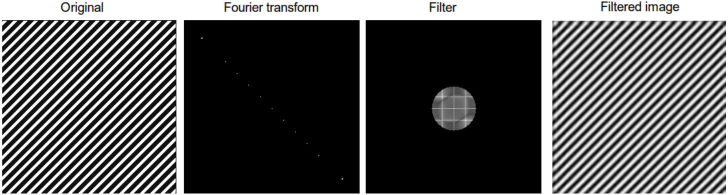
Convolution (cont'd)



- Convolution is often referred to as filtering
- Other names: kernel, receptive field

Image credit: Timdettmers.com

Convolution (cont'd)



- Convolution is often referred to as filtering
- Other names: kernel, receptive field
- And now:

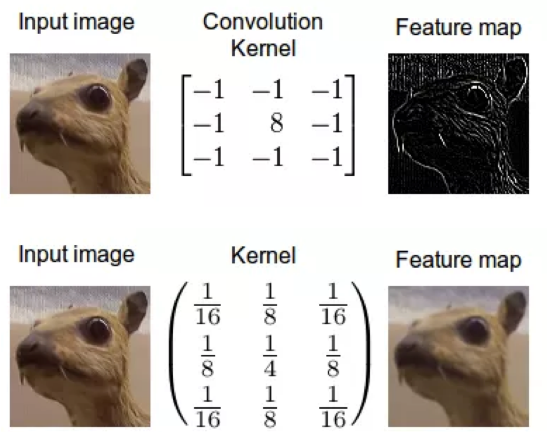
$$feature\_map = filter * input$$

$$= \sum_{y=1}^{n\_columns} \left(\sum_{x=1}^{n\_rows} input(x-a, y-b) filter(x, y) \right)$$

$$= \mathcal{F}^{-1}(\sqrt{2\pi} \quad \mathcal{F}|input| \cdot \mathcal{F}|filter|)$$

Image credit: Timdettmers.com

Convolution (cont'd)



- Similarity: Kernel SVM, PCA?

Image credit: Timdettmers.com

Break?

Recap: Fun with Python

CarnegieMellon

fun_break.py

```python
# what is the difference?
x = [10 * i for i in range(100)]
type(x)

y = (10 * i for i in range(100))
type(y)

# does this work?
for i in range(10): yield i**2

# does this work?
def f():
  for i in xrange(10):
    yield i**2
type(f)
for j in f():
  print(j)

# does this work?
f = lambda: [(yield i**2) for i in range(10)]
type(f)
for j in f():
  print(j)
```

End of Break!

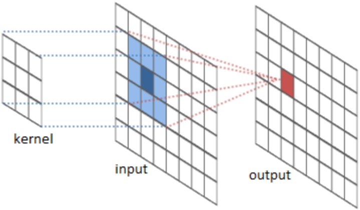
Outline

Carnegie Mellon

1. Recap and motivation
2. Convolution
3. Convolution in Neural Network
4. Pooling
5. Case Studies

Convolution In Action

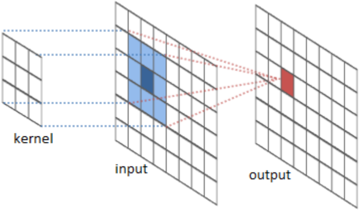
Carnegie Mellon



- Given a 200x200 image
- A fully-connected layer with n hidden neurons: $200 * 200 * n$ params

Image credit: ricardodeazambuja.com

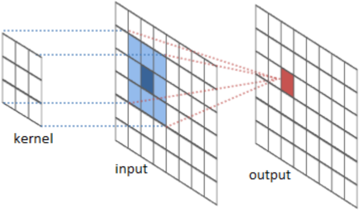
Convolution In Action



- Given a 200x200 image
- A fully-connected layer with n hidden neurons: $200 * 200 * n$ params
- With convolution with filter size 3x3, then how many params?

Image credit: ricardodeazambuja.com

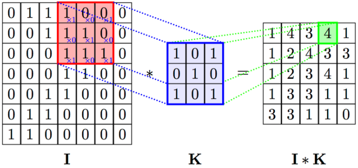
Convolution In Action



- Given a 200x200 image
- A fully-connected layer with n hidden neurons: $200 * 200 * n$ params
- With convolution with filter size 3x3, then how many params? $3 * 3 * n$

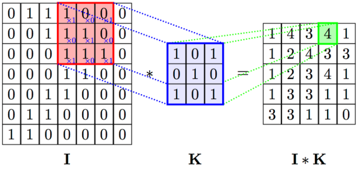
Image credit: ricardodeazambuja.com

Convolution In Action (cont'd)



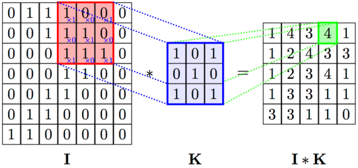
- How it works?

Convolution In Action (cont'd)



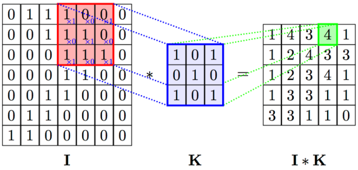
- How it works? Simply, it's a dot product

Convolution In Action (cont'd)



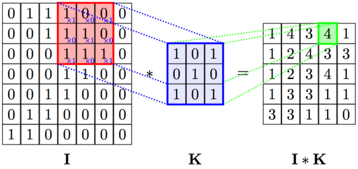
- How it works? Simply, it's a dot product
- A filter scan through the whole image which is convolved by it
- *Important*: a filter has a fixed weight → output is similar if the region is similar

Convolution In Action (cont'd)



- How it works? Simply, it's a dot product
- A filter scan through the whole image which is convolved by it
- *Important*: a filter has a fixed weight → output is similar if the region is similar
- Practice: $n * n$ image, $k * k$ filter, output?

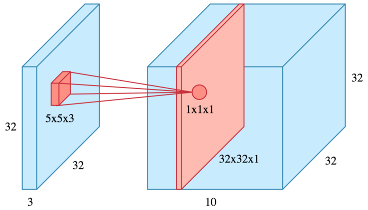
Convolution In Action (cont'd)



- How it works? Simply, it's a dot product
- A filter scan through the whole image which is convolved by it
- *Important*: a filter has a fixed weight $\rightarrow$ output is similar if the region is similar
- Practice: $n * n$ image, $k * k$ filter, output? $(n-k+1)*(n-k+1)$

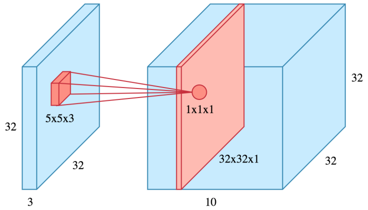
Depth of Convolution

Carnegie Mellon



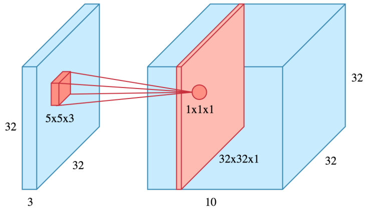
- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$

Depth of Convolution



- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation?

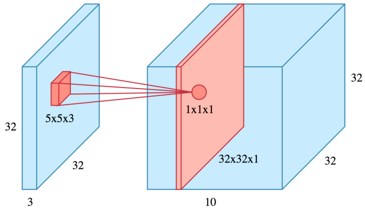
Depth of Convolution



- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.

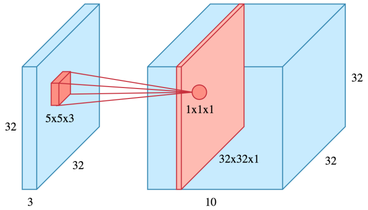
Depth of Convolution

CarnegieMellon



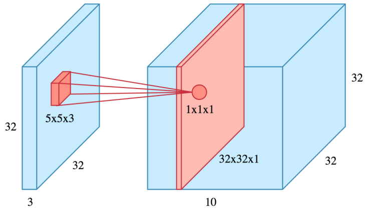
- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.
- Recall, what if we want many feature maps?

Depth of Convolution



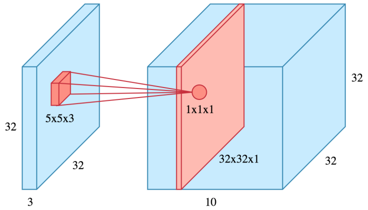
- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.
- Recall, what if we want many feature maps? Use many filters.
- Practice: $n * n * 3$ image, $k * k * 3$ filter, output?

Depth of Convolution



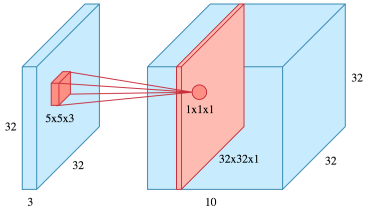
- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.
- Recall, what if we want many feature maps? Use many filters.
- Practice: $n * n * 3$ image, $k * k * 3$ filter, output? $(n - k + 1) * (n - k + 1) *$

Depth of Convolution



- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.
- Recall, what if we want many feature maps? Use many filters.
- Practice: $n * n * 3$ image, $k * k * 3$ filter, output? $(n - k + 1) * (n - k + 1)*1$

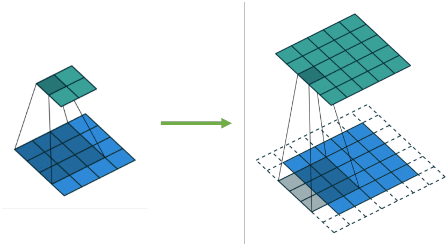
Depth of Convolution



- Filter shares the *same depth* with input
- If input image has 3 channels: $200 * 200 * 3 \rightarrow$ the filter should be $n * n * 3$
- Dimension of each convolution operation? Simply, a scalar.
- Recall, what if we want many feature maps? Use many filters.
- Practice: $n * n * 3$ image, $k * k * 3$ filter, output? $(n - k + 1) * (n - k + 1) * 1$
- Practice: If there are 10 filters?

Padding

Carnegie Mellon

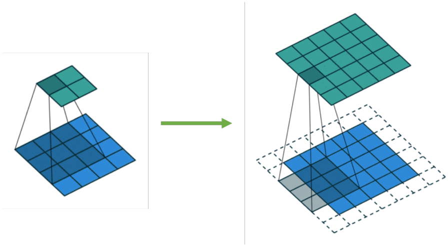


- Change the size of output, normally with zero Demo

Image credit: deeplearning.net

Padding

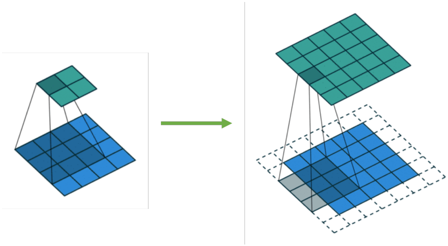
Carnegie Mellon



- Change the size of output, normally with zero ▸ Demo
- There are 2 types of padding
    - *Valid*: no padding

Image credit: deeplearning.net

Padding

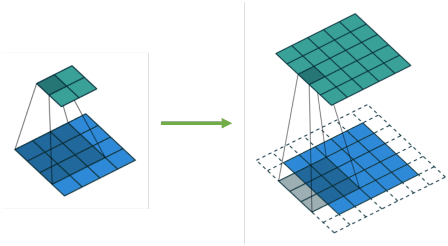


- Change the size of output, normally with zero ▸ Demo
- There are 2 types of padding
  - *Valid*: no padding
  - *Same*: pad so that output size is the same as input's

Image credit: deeplearning.net

Padding

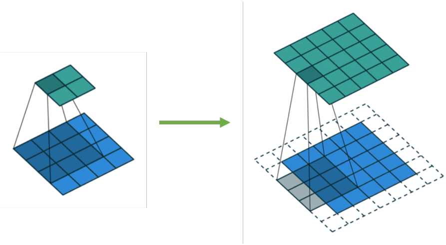
Carnegie Mellon



- Change the size of output, normally with zero Demo
- There are 2 types of padding
  - *Valid*: no padding
  - *Same*: pad so that output size is the same as input's
- Practice: $n * n$ convolves $k * k$, padding p, output?

Image credit: deeplearning.net

Padding

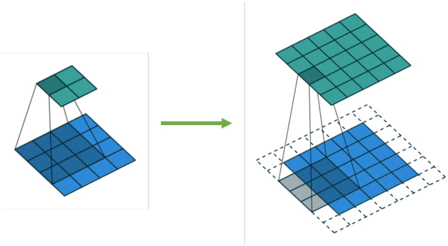


- Change the size of output, normally with zero ▸ Demo
- There are 2 types of padding
  - *Valid*: no padding
  - *Same*: pad so that output size is the same as input's
- Practice: $n * n$ convolves $k * k$, padding p, output?

  $(n - k + 1 + 2p) * (n - k + 1 + 2p)$

Image credit: deeplearning.net

Padding



- Change the size of output, normally with zero Demo
- There are 2 types of padding
  - *Valid*: no padding
  - *Same*: pad so that output size is the same as input's
- Practice: $n * n$ convolves $k * k$, padding p, output?
  $(n - k + 1 + 2p) * (n - k + 1 + 2p)$
- Assumption of this formula (and the ones above)?

Image credit: deeplearning.net

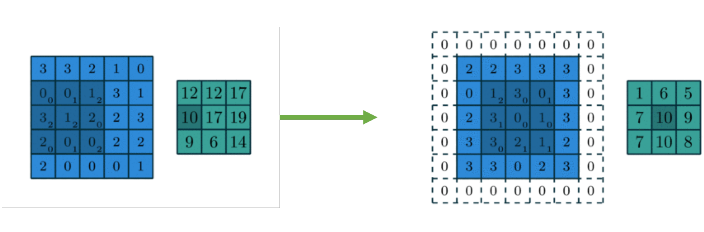
Stride

Carnegie Mellon

- Velocity of convolution ▸ Demo
- Practice: 5x5 convolves 3x3
  - Valid padding, stride 1, ouput?

Image credit: deeplearning.net

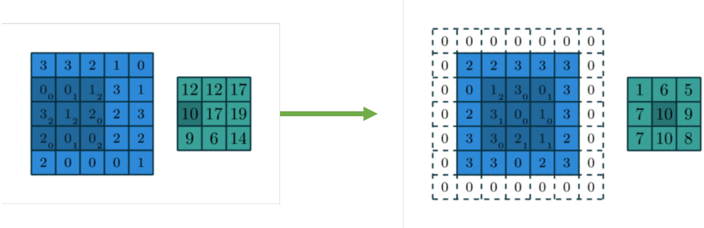
Stride



- Velocity of convolution ▸ Demo
- Practice: 5x5 convolves 3x3
  - Valid padding, stride 1, ouput? $(5-3+1)(5-3+1)$
  - Pad 1, stride 1, output?

Image credit: deeplearning.net

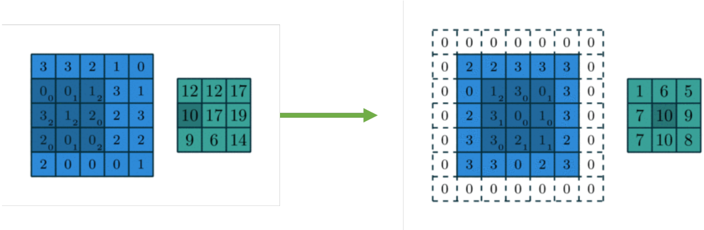
Stride



- Velocity of convolution ▸ Demo
- Practice: 5x5 convolves 3x3
  - Valid padding, stride 1, ouput? $(5-3+1)(5-3+1)$
  - Pad 1, stride 1, output? $(5-3+1+2)(5-3+1+2)$
  - Pad 1, stride 2, output?

Image credit: deeplearning.net

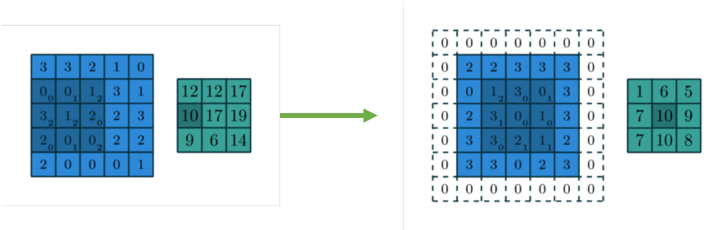
Stride



- Velocity of convolution ▸ Demo
- Practice: 5x5 convolves 3x3
  - Valid padding, stride 1, ouput? $(5 - 3 + 1)(5 - 3 + 1)$
  - Pad 1, stride 1, output? $(5 - 3 + 1 + 2)(5 - 3 + 1 + 2)$
  - Pad 1, stride 2, output? $((5 - 3 + 2)/2 + 1)(((5 - 3 + 2)/2 + 1))$

Image credit: deeplearning.net

Stride



- Velocity of convolution [Demo]
- Practice: 5x5 convolves 3x3
  - Valid padding, stride 1, ouput? $(5-3+1)(5-3+1)$
  - Pad 1, stride 1, output? $(5-3+1+2)(5-3+1+2)$
  - Pad 1, stride 2, output? $((5-3+2)/2+1)(((5-3+2)/2+1))$
  - Which one is Same padding?
- So the formula is: $(\frac{n-k+2p}{2}+1)$, Note: n can vary by dimensions
- Practice: *Same* convolution, how p relates to k?

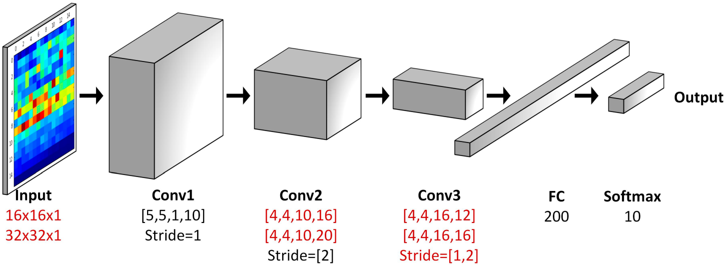
Image credit: deeplearning.net

Break?

Outline

Motivation of Pooling

Carnegie Mellon

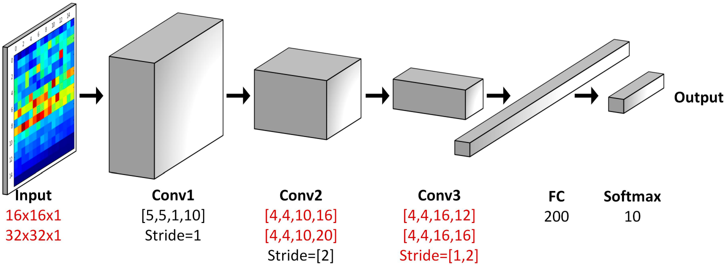


- Practice: given valid conv and zero padding, identify filter size at each step? Then calculate number of parameters at each step?

Image credit: mdpi.com

Motivation of Pooling

Carnegie Mellon

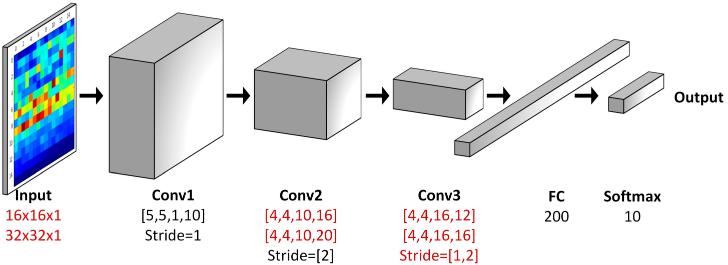


- Practice: given valid conv and zero padding, identify filter size at each step? Then calculate number of parameters at each step?
- Can we go deeper with this? Imagine the real world cases of 200x200, or even 1024x1024.

Image credit: mdpi.com

Motivation of Pooling

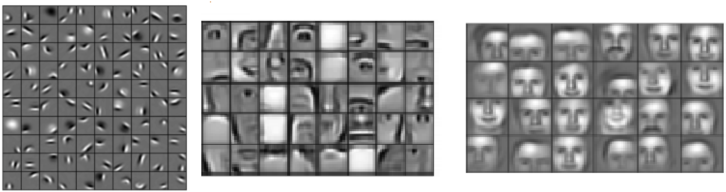
Carnegie Mellon



- Practice: given valid conv and zero padding, identify filter size at each step? Then calculate number of parameters at each step?
- Can we go deeper with this? Imagine the real world cases of 200x200, or even 1024x1024.
- Too many params $\rightarrow$ overfitting $\rightarrow$ reduce number of params

Image credit: mdpi.com

More Motivation of Pooling

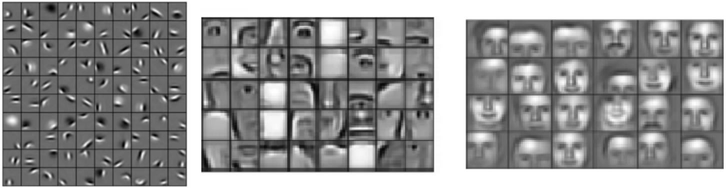


- We want to reduce size of feature maps to easily control

Image credit: mdpi.com

More Motivation of Pooling

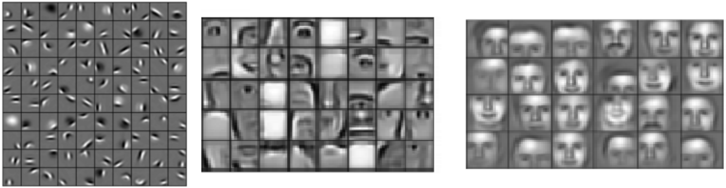
Carnegie Mellon



- We want to reduce size of feature maps to easily control
- To prevent loss, we summarize features:
  - Each node (scalar) in a feature map represents features of a kernel-size region

Image credit: mdpi.com

More Motivation of Pooling

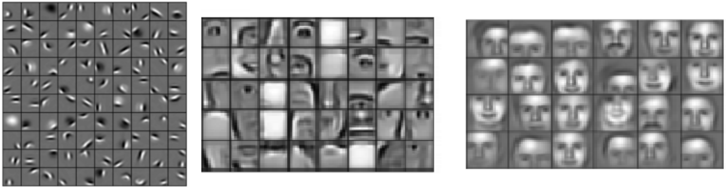


- We want to reduce size of feature maps to easily control
- To prevent loss, we summarize features:
    - Each node (scalar) in a feature map represents features of a kernel-size region
    - Adjacent nodes represent a local region's features

Image credit: mdpi.com

More Motivation of Pooling

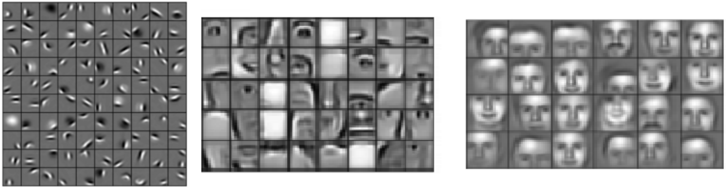
Carnegie Mellon



- We want to reduce size of feature maps to easily control
- To prevent loss, we summarize features:
  - Each node (scalar) in a feature map represents features of a kernel-size region
  - Adjacent nodes represent a local region's features
  - Summarize adjacent nodes, we have features of a 'big' local region

Image credit: mdpi.com

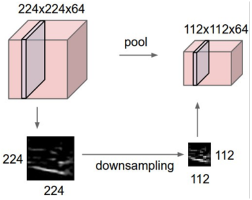
More Motivation of Pooling

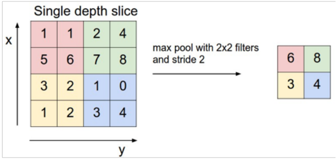


- We want to reduce size of feature maps to easily control
- To prevent loss, we summarize features:
  - Each node (scalar) in a feature map represents features of a kernel-size region
  - Adjacent nodes represent a local region's features
  - Summarize adjacent nodes, we have features of a 'big' local region $\rightarrow$ *hierarchy*
- To get *translation invariance* for higher object levels

Image credit: mdpi.com

How Pooling Works?

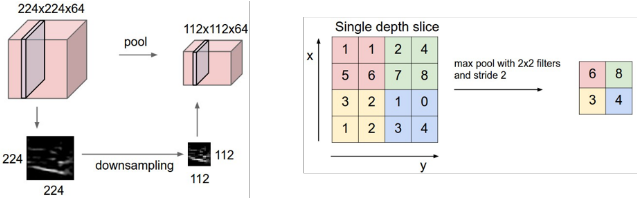




- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.

Image credit: http://cs231n.github.io/convolutional-networks

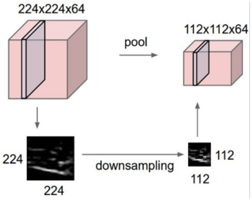
How Pooling Works?

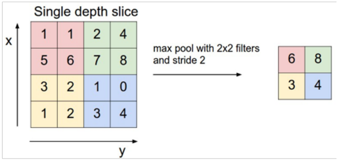


- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.
- The window is also called a kernel/filter
- Popular types: MAX, AVG
- Practice: 5x5 with filter 3x3, stride 1, output?

Image credit: http://cs231n.github.io/convolutional-networks

How Pooling Works?



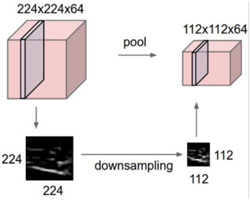


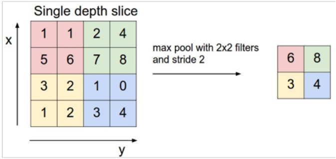
- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.
- The window is also called a kernel/filter
- Popular types: MAX, AVG
- Practice: 5x5 with filter 3x3, stride 1, output? 2x2
- Practice: And how many params?

Image credit: http://cs231n.github.io/convolutional-networks

How Pooling Works?

Carnegie Mellon

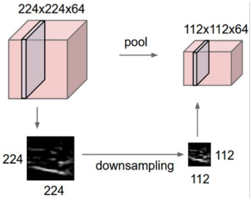


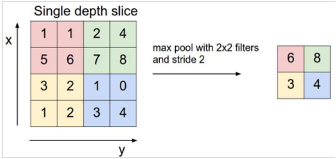


- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.
- The window is also called a kernel/filter
- Popular types: MAX, AVG
- Practice: 5x5 with filter 3x3, stride 1, output? 2x2
- Practice: And how many params? Zero

Image credit: http://cs231n.github.io/convolutional-networks

How Pooling Works?



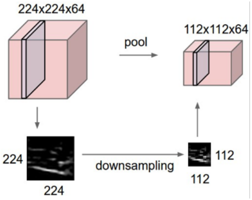


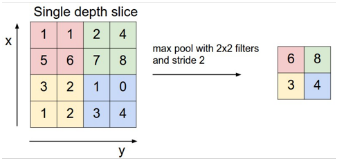
- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.
- The window is also called a kernel/filter
- Popular types: MAX, AVG
- Practice: 5x5 with filter 3x3, stride 1, output? 2x2
- Practice: And how many params? Zero, and None padding.
- Output formula?

Image credit: http://cs231n.github.io/convolutional-networks

How Pooling Works?

Carnegie Mellon





- Process each feature map *independently*
- *a.k.a* subsampling or downsampling process.
- The window is also called a kernel/filter
- Popular types: MAX, AVG
- Practice: 5x5 with filter 3x3, stride 1, output? 2x2
- Practice: And how many params? Zero, and None padding.
- Output formula? $(n - k)/s + 1$

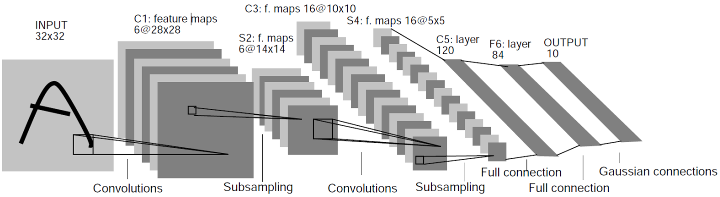
Image credit: http://cs231n.github.io/convolutional-networks

Outline

Carnegie Mellon

1. Recap and motivation
2. Convolution
3. Convolution in Neural Network
4. Pooling
5. Case Studies

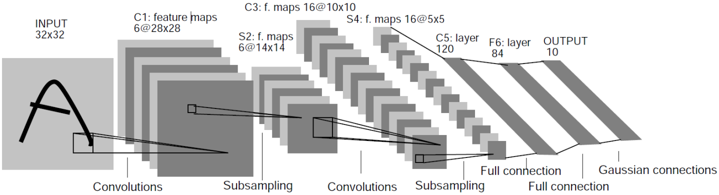
LeNet-5



- Practice: what are the sizes of kernels and strides for Conv, and for Pooling?

Image credit: pythonmachinelearning.pro

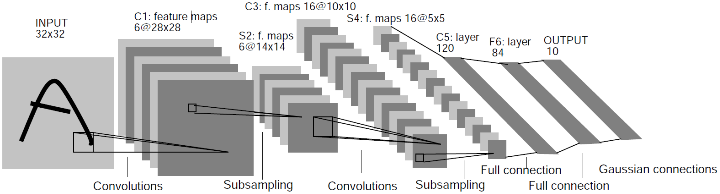
LeNet-5



- Practice: what are the sizes of kernels and strides for Conv, and for Pooling? conv: 5x5 and 1, pool: 2x2 and 2
- Practice: how many params at each step?

Image credit: pythonmachinelearning.pro

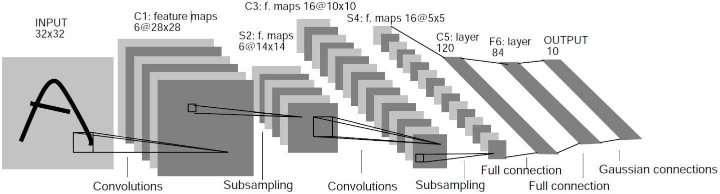
LeNet-5



- Arguably the first CNN that *really* works ▸ Demo
- Practice: what are the sizes of kernels and strides for Conv, and for Pooling?

Image credit: Yann Lecun et. al 1998

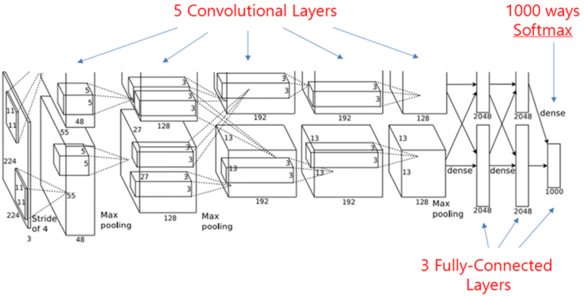
LeNet-5



- Arguably the first CNN that *really* works ▸ Demo
- Practice: what are the sizes of kernels and strides for Conv, and for Pooling? conv: 5x5 and 1, pool: 2x2 and 2
- Practice: how many params at each step?

Image credit: Yann Lecun et. al 1998

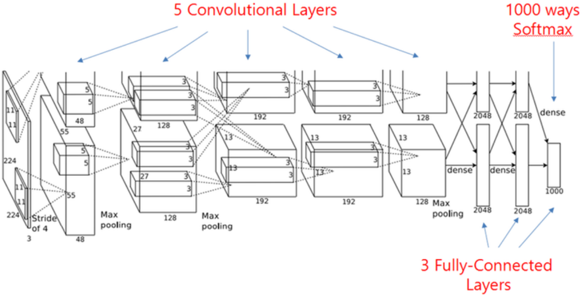
AlexNet: A Revolution



- First use of ReLU, Norm layers, heavy augmentation, dropout, momentum
- Input: 227x227x3
- Conv1: 96 of $11 * 11$ filters, stride 4, output?

Image credit: Krizhevsky et. al 2012

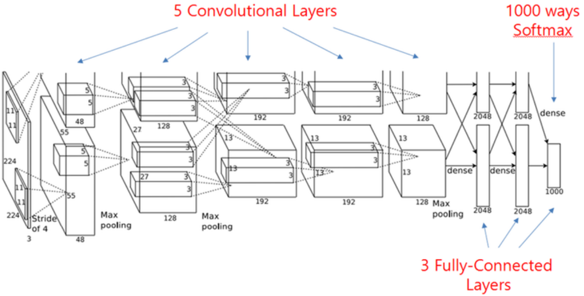
AlexNet: A Revolution



- First use of ReLU, Norm layers, heavy augmentation, dropout, momentum
- Input: 227x227x3
- Conv1: 96 of $11 * 11$ filters, stride 4, output? $55 * 55 * 96$,

Image credit: Krizhevsky et. al 2012

AlexNet: A Revolution

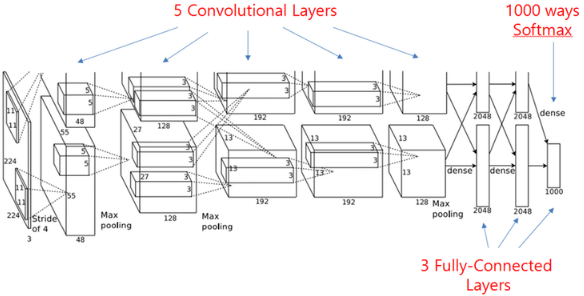


- First use of ReLU, Norm layers, heavy augmentation, dropout, momentum
- Input: 227x227x3
- Conv1: 96 of $11 * 11$ filters, stride 4, output? $55 * 55 * 96$, params?

Image credit: Krizhevsky et. al 2012

AlexNet: A Revolution

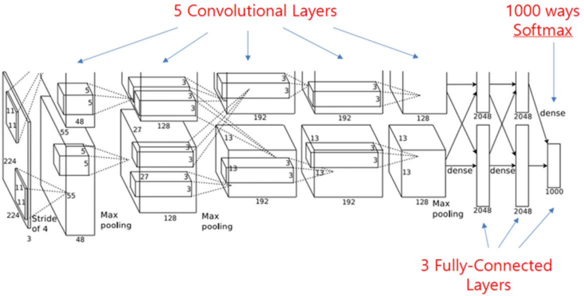
CarnegieMellon



- First use of ReLU, Norm layers, heavy augmentation, dropout, momentum
- Input: 227x227x3
- Conv1: 96 of $11 * 11$ filters, stride 4, output? $55 * 55 * 96$, params? $(11 * 11 * 3) * 96$
- Pool1: $3 * 3$ filter, stride 2, output?

Image credit: Krizhevsky et. al 2012

AlexNet: A Revolution



- First use of ReLU, Norm layers, heavy augmentation, dropout, momentum
- Input: 227x227x3
- Conv1: 96 of $11 * 11$ filters, stride 4, output? $55 * 55 * 96$, params? $(11 * 11 * 3) * 96$
- Pool1: $3 * 3$ filter, stride 2, output? $27 * 27 * 96$

Image credit: Krizhevsky et. al 2012

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output?
  $27 * 27 * 256$ (same padding),

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output?
  $27 * 27 * 256$ (same padding), params?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output?
  $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding),

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding),

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding),

AlexNet (cont'd)

- Conv2: 256 of $5*5$ filters, stride 1, padding 2, output? $27*27*256$ (same padding), params? $(5*5*96)*256$
- Pool2: $3*3$ filter, stride 2, output? $13*13*256$
- Conv3: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*256)*384$
- Conv4: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*384)*384$
- Conv5: 256 of $3*3$ filters, stride 1, padding 1, output? $13*13*256$ (same padding), params?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output?

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,

AlexNet (cont'd)

- Conv2: 256 of $5*5$ filters, stride 1, padding 2, output? $27*27*256$ (same padding), params? $(5*5*96)*256$
- Pool2: $3*3$ filter, stride 2, output? $13*13*256$
- Conv3: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*256)*384$
- Conv4: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*384)*384$
- Conv5: 256 of $3*3$ filters, stride 1, padding 1, output? $13*13*256$ (same padding), params? $(3*3*384)*256$
- Pool3: $3*3$ filter, stride 2, output? $6*6*256$
- FC6: 4096 neurons, output? 4096,params?

AlexNet (cont'd)

- Conv2: 256 of $5*5$ filters, stride 1, padding 2, output? $27*27*256$ (same padding), params? $(5*5*96)*256$
- Pool2: $3*3$ filter, stride 2, output? $13*13*256$
- Conv3: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*256)*384$
- Conv4: 384 of $3*3$ filters, stride 1, padding 1, output? $13*13*384$ (same padding), params? $(3*3*384)*384$
- Conv5: 256 of $3*3$ filters, stride 1, padding 1, output? $13*13*256$ (same padding), params? $(3*3*384)*256$
- Pool3: $3*3$ filter, stride 2, output? $6*6*256$
- FC6: 4096 neurons, output? 4096,params? $6*6*256*4096$
- FC7: 4096 neurons, output?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,

AlexNet (cont'd)

Carnegie Mellon

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,params?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,params? $4096 * 4096$
- FC8: 1000 neurons, output?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,params? $4096 * 4096$
- FC8: 1000 neurons, output? 1000,

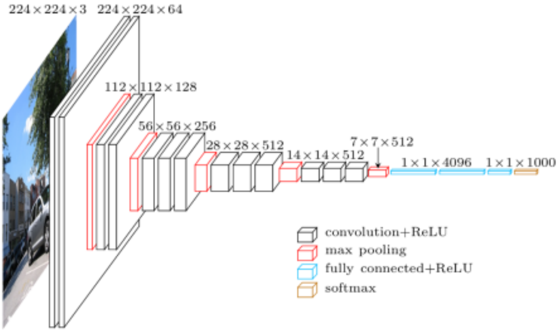
AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,params? $4096 * 4096$
- FC8: 1000 neurons, output? 1000,params?

AlexNet (cont'd)

- Conv2: 256 of $5 * 5$ filters, stride 1, padding 2, output? $27 * 27 * 256$ (same padding), params? $(5 * 5 * 96) * 256$
- Pool2: $3 * 3$ filter, stride 2, output? $13 * 13 * 256$
- Conv3: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 256) * 384$
- Conv4: 384 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 384$ (same padding), params? $(3 * 3 * 384) * 384$
- Conv5: 256 of $3 * 3$ filters, stride 1, padding 1, output? $13 * 13 * 256$ (same padding), params? $(3 * 3 * 384) * 256$
- Pool3: $3 * 3$ filter, stride 2, output? $6 * 6 * 256$
- FC6: 4096 neurons, output? 4096,params? $6 * 6 * 256 * 4096$
- FC7: 4096 neurons, output? 4096,params? $4096 * 4096$
- FC8: 1000 neurons, output? 1000,params? $4096 * 1000$

VGG



- Got 11.2% top 5 error at ILSVRC 2013

Image credit: David Frossard

Stay Tuned!

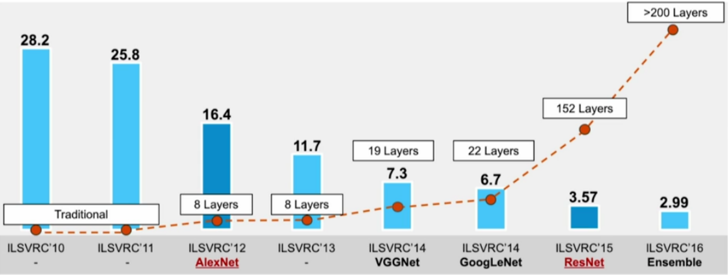


Image credit: SQLML