DATA SCIENCE INSTITUTE
DUG
PRESENTS

MOVIE NIGHT

THE IMITATION GAME

Join us for a movie with popcorn and other refreshments!

FEBRUARY 24
Solomon 001

6:00 PM
DALL-E 2 prompt “a painting of deep underwater with a yellow submarine in the bottom right corner”
Recap

Filters/Kernels and Stride

Learning filters

CNNs are partially connected networks

Tensorflow conv2d function

Padding

Application to MNIST/CIFAR

Convolution

Convolution in Tensorflow

```
tf.nn.conv2d(input, filter, strides, padding)
```

Input Image (4-D Tensor)  | Filter/Kernel (4-D Tensor)  | Strides along each dimension  | Type of Padding (String “Valid” or “Same”)
Today’s goal – continue to learn about CNNs

(1) Convolutional Neural Network (CNN) architecture

(2) First successful CNN - AlexNet
   Pooling and translational invariance

(3) Deeper CNNs!
   Residual Blocks
   Batch normalization
Bias Term in Convolution Layers

Just like a fully connected layer, we can have a learnable additive bias for convolution.
Adding a Bias in Tensorflow

If you use tf.nn.conv2d, bias can be added with:

\[
\text{tf.nn.bias_add(value, bias)}
\]

Full documentation here: [https://www.tensorflow.org/api_docs/python/tf/nn/bias_add](https://www.tensorflow.org/api_docs/python/tf/nn/bias_add)
Adding a Bias in Tensorflow

If you are using keras layers, bias is included by default:

```python
tf.keras.layers.Conv2D(filters, kernel_sz, strides, padding, use_bias = True)
```

- Number of filters
- Filter Size
- Strides along each dimension
- Type of Padding (VALID or SAME)

Full documentation here: [https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D)
Our neural network so far
Convolutional Neural Network Architecture

![Diagram of Convolutional Neural Network Architecture with layers labeled as 64, 64, and 3. There is a green block with 64 units, a linear layer, a purple block labeled as Σ, and a blue block labeled as softmax, leading to an output label.]
CNN Architecture

- CNN Architecture Diagram:
  - Input: 3x3x64
  - Convolutional Layers:
    - 5x5x64 ReLU + Pool
  - Fully Connected Layer:
    - Linear Layer
    - Softmax Function
  - Output: 10 classes

- Key Components:
  - Convolutional Layer: 5x5x64
  - ReLU Activation
  - Pooling
  - Linear Layer
  - Softmax
CNN Architecture
CNN Architecture

[Diagram showing a CNN architecture with layers, filters, and activation functions such as ReLU and pooling.]
CNN Architecture

This part learns to extract **features** from the image.
CNN Architecture

A single convolutional layer

- 64x64 input
- 3 convolutional layers:
  - 3x3 ReLU
  - Pool
  - 5x5 ReLU
  - Pool
  - 3x3 ReLU
- Linear layer
- Softmax

output
CNN Architecture

Activation after filter passes over image

linear layer softmax

output
This part learns to perform a specific task (e.g. classification) using those features
CNN Architecture

Flattened data (beginning of the fully connected portion)

linear layer -> softmax

output
CNN Architecture

Fully connected layers to classify input

linear layer  softmax

output
CNN Architecture

Why multiple convolutional layers?

Input

Label=“Llama”

Output
Feature Extraction using multiple convolution layers

Hierarchy of features

Sequence of layers detect broader and broader features
Example: Network Dissection

Layer 3 active regions

Layer 4 active regions

Layer 5 active regions

“Eye Detector”

“Eyes and Nose Detector”

“Dog Face Detector”

http://netdissect.csail.mit.edu/

Any questions?
ILSVRC 2012
(ImageNet Large Scale Visual Recognition Challenge)

The classification task on ImageNet:

For each image, assign 5 labels in order of decreasing confidence. Success if one of these labels matches the ground truth.

Predictions:

1. Carpet
2. Zebra
3. Llama
4. Flower
5. Horse

https://commons.wikimedia.org/wiki/File:Common_zebra_1.jpg
ILSVRC 2012
Percentage that model fails to classify is known as *Top 5 Error Rate*

Predictions:
1. Sponge
2. Person
3. Llama
4. Flower
5. Boat

https://commons.wikimedia.org/wiki/File:Puffer_Fish_DSC01257.JPG
AlexNet: Why CNNs Are a Big Deal

Major performance boost on ImageNet at ILSVRC 2012
Top 5 error rate of 15.3% compared to 26.2% achieved by 2nd place

Note: SuperVision is the name of Alex’s team

http://image-net.org/challenges/LSVRC/2012/analysis/
AlexNet

- 60 million parameters
- 5 Convolutional Layers
- 3 Fully Connected Layers

[Alex Krizhevsky et al. 2012]

Pooling
Max Pooling

Max pooling with stride 2 and 2x2 filters

Max of pixels in window
Max Pooling

Max pooling with stride 2 and 2x2 filters

Max of pixels in window

\[
\begin{array}{cccc}
6 & 3 & 1 & -3 \\
4 & 1 & 2 & 0 \\
3 & 1 & 3 & 2 \\
7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cc}
6 \\
\end{array}
\]
Max Pooling

Max pooling with stride 2 and 2x2 filters

Max of pixels in window
Max Pooling

Max pooling with stride 2 and 2x2 filters

Max of pixels in window
Max Pooling

Max pooling with stride 2 and 2x2 filters

Max of pixels in window
Max Pooling

Max pooling with stride 2 and 2x2 filters

Why use Max Pooling?
Pooling: Motivation

Max Pooling

- Keeps track of regions with highest activations, indicating object presence
- Controllable way to lower (coarser) resolution (down sample the convolution output)
Other Pooling Techniques

Average pooling with stride 2 and 2x2 filters

Average pixel values in each window
Learning a Pooling Function

- The network can learn its own pooling function
- Implement via a strided convolution layer

Learned filter weights
So...did we achieve our goal of translational invariance?
What was Translational Invariance again?

• To make a neural net $f$ robust in this same way, it should ideally satisfy \textit{translational invariance}: $f(T(x)) = f(x)$, where
  • $x$ is the input image
  • $T$ is a translation (i.e. a horizontal and/or vertical shift)
Are CNNs Translation Invariant?

• Convolution is *translation equivariant*

  • A translated input results in an output translated by the same amount

  \[ f(T(I)) = T(f(I)) \]

  \[ (T(I) \odot K)(x, y) = T(I \odot K)(x, y) \]

\[ f \uparrow T \]

\[ f \downarrow T \]

*Here, \((l \odot K)(x, y) = \sum_{m} \sum_{n} I(x + m, y + n)K(m, n)\)*
Are CNNs Translation Invariant?

- Max pooling is intended to give invariance to small translations
  - The highest activation pixel can shift around within the pooling window, and the output does not change

\[
\begin{align*}
  f(\begin{bmatrix} 6 & 3 \\ 4 & 1 \end{bmatrix}) &= 6 \\
  f(\begin{bmatrix} 1 & 5 \\ 6 & 3 \end{bmatrix}) &= 6 \\
  f(\begin{bmatrix} 2 & 6 \\ 2 & 4 \end{bmatrix}) &= 6
\end{align*}
\]
So how does it all come together?

Convolution is \textit{translation equivariant}

Max pooling gives invariance to small translations
Are CNNs Translation Invariant?

• Answer: CNNs are “sort of” translation invariant
  • Shifting the content of the image around tends not to drastically effect the output classification probabilities...
Are CNNs Translation Invariant?

- Answer: CNNs are “sort of” translation invariant
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...

Are CNNs Translation Invariant?

- Answer: CNNs are “sort of” translation invariant
  - Shifting the content of the image around tends not to drastically effect the output classification probabilities...
  - ...but they are **not**, strictly speaking, translation invariant

Are CNNs Translation Invariant?

• Is it possible to build a truly translation invariant CNN?
  • Yes!
  • Have to properly “pre-filter” images before pooling them
  • Comes from signal processing theory (The Sampling Theorem)
  • Take CS 1230 (Computer Graphics) if you want to learn about this!

• One effort to make a translation-invariant CNN:
Other Invariances

Rotation/Viewpoint Invariance
Other Invariances

Rotation/Viewpoint Invariance

Size Invariance
Other Invariances

Rotation/Viewpoint Invariance

Size Invariance

Illumination Invariance

• All of these are desirable

• How do CNNs fare?
  • Max pooling gives some small amount of size invariance...
  • ...but in general, CNNs don’t do well with big changes in size, pose, or lighting

• Consequence of not having these invariances?
  • Need lots of training data
  • Have to show the network examples of everything under different poses, lighting, etc.
  • Data Augmentation

What should we do we do?
More Complicated Networks

AlexNet:

VGG:

224x224x3

224x224x64

112x112x128

56x56x256

28x28x512

14x14x512

7x7x512

28x28x512

1x4096

1x4096

1x1000

σ

output
Can you guess what was the biggest bottleneck to adding more layers?

Go to www.menti.com and use the code 4671 9420

https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33
Revolution of Depth

Vanishing gradients!

More Complicated Networks

ResNet:
Lots of layers, tons of learnable parameters
Avoids Vanishing Gradient problem
but how?

More Complicated Networks

ResNet:
Lots of layers, tons of learnable parameters
Avoids Vanishing Gradient problem

Residual Blocks

• In very deep nets, each layer often needs to learn just a small transformation of the preceding layer (identity + change)

• Idea: explicitly design the network such that the output of each layer is the identity + some deviation from it
  • Deviation is known as a residual
Residual Blocks

- In very deep nets, each layer often needs to learn just a small transformation of the preceding layer (identity + change).
  - Idea: explicitly design the network such that the output of each layer is the identity + some deviation from it.
    - Deviation is known as a residual.
- Allows gradient to flow through two pathways.
- Significantly stabilizes training of very deep networks.

Batch Normalization (stabilizing training)

Idea: normalize the activations for each feature at each layer

Why might we want to do this?
Batch Normalization: Motivation

More stable inputs = faster training

MNIST test accuracy vs number of training steps

Batch Normalization: Implementation

For each feature $x$, start by calculating the batch mean and standard deviation for each feature:

\[
\mu_{\text{batch}} = \frac{\sum_{i=0}^{\text{batch\_size}} x_i}{\text{batch\_size}}
\]

\[
\sigma_{\text{batch}} = \sqrt{\frac{\sum_{i=0}^{\text{batch\_size}} (x_i - \mu_{\text{batch}})^2}{\text{batch\_size}}}
\]
Batch Normalization: Implementation

Normalize by subtracting feature $x$’s batch mean, then divide by batch standard deviation.

$$x' = \frac{x - \mu_{\text{batch}}}{\sigma_{\text{batch}}}$$

Feature $x$ now has mean 0 and variance 1 along the batch
Batch Normalization in Tensorflow

tf.keras.layers.BatchNormalization(input)

Documentation: https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization
Recap

CNNs

Architecture

AlexNet + Pooling

CNNs are “sort of” translationally invariant

Many layers = vanishing gradient

Deeper CNNs

ResNet + Residual blocks

Batch normalization

Revolution of Depth

ImageNet Classification top-5 error (%)