Deep AutoEncoder
CSE/ECE 577

Beibin Li
Nov/1/2021
Summary

- Convolutional Neural Network
  - Learning Convolutions
  - CNN for Classification
  - CNN for Segmentation

- Autoencoder (AE)
  - Birth of Autoencoder
  - U-Net
  - PyTorch Implementation in HW 2

- Other Applications for Autoencoder

- Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision
Convolution Neural Network
CNN

- Convolution + nonlinearity
- Max pooling
- Convolution + pooling layers
- Fully connected layers
- Nx binary classification
Convolution: from fixed to learnable

Expert-Designed Convolution: SIFT, HoG, LBP, ...

Learn Flexible Parameters
FC Activation: From Step Function to Sigmoid

(a) Step Function

(b) Sigmoid Function
CNN Activation: From Sigmoid to ReLU

Sigmoid

\[ f(x) = \frac{1}{1 + e^{-x}} \]

TanH

\[ \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

ReLU

\[ f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
 x & \text{for } x \geq 0 
\end{cases} \]
Representation

Input = (3, width, height)

Input

<table>
<thead>
<tr>
<th>4</th>
<th>9</th>
<th>2</th>
<th>5</th>
<th>8</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Parameters:
- Size: \( f = 3 \)
- Stride: \( s = 1 \)
- Padding: \( p = 0 \)

\[ n_h \times n_w = 6 \times 6 \]

Filter

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Result

\[ \begin{bmatrix} 2 & 6 \end{bmatrix} \]

Meet Softmax

\[ \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K. \]

Scores (Logits)  \[ Z \]

\[ \begin{cases} 2.0 & p(y=0) = 0.7 \\ 1.0 & p(y=1) = 0.2 \\ 0.1 & p(y=2) = 0.1 \end{cases} \]

Probabilities
Inference

https://datascience.stackexchange.com/questions/27506/back-propagation-in-cnn
Learning

Details:
- [https://www.slideshare.net/EdwinEfranJimnezLepe/example-feedforward-backpropagation](https://www.slideshare.net/EdwinEfranJimnezLepe/example-feedforward-backpropagation)

\[
\begin{align*}
O_{11} &= F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22} \\
O_{12} &= F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23} \\
O_{21} &= F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32} \\
O_{22} &= F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}
\end{align*}
\]
CNN for Image Classification
Vanilla Structure for Image Classification

Feature Extraction from Image

Classification

$E_{total} = \sum \frac{1}{2} (target - output)^2$
Convolutional Unit (CU) - VGG

Basic Block in ResNet


- Residual Connection
- Element-wise addition of input and output
- Improves gradient flow and accuracy
- In ResNet-18 and ResNet-34
- Still computationally expensive
  - Hard to train very deep networks (> 100 layers)
Bottleneck in ResNet

- Used in ResNet-50, ResNet-101, ResNet-152, etc...
- Computationally Efficient

Influence:
- Bottleneck unit with Depth-wise convs
  - MobileNetv2
  - ShuffleNetv2

Other Structures and Blocks

- InceptionNet, 2014, 2016 (v4)
- DenseNet, 2016
- MobileNet, 2017
- ESPNet, 2018
- EfficientNet, 2019
- RegNet, 2021
- ...

Transformer-based (Not convolutional)
- ViT, 2020
- DERT, 2020
CNN for Semantic Segmentation
Encoding in Brain

- Brains can:
  - Encode, store and recall information
  - Learn and adapt from previous experiences
  - Build relationships of data
- **Encoding** allows a **perceived item** to be converted into a **construct** that can be stored within the brain and recalled later.
- Studied since 1880s.

(Wikipedia)
Encoding in Computer

Information Source

Message

Transmitter

Signal

Received Signal

Receiver

Message

Destination

Noise Source
CHAPTER 8

Learning Internal Representations by Error Propagation

D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS
The wake-sleep algorithm for unsupervised neural networks

Geoffrey E Hinton*, Peter Dayan, Brendan J Frey, Radford M Neal

Department of Computer Science
University of Toronto
6 King’s College Road
Toronto M5S 1A4, Canada

3rd April 1995

Autoencoder (aka, Encoder-Decoder)

- PCA (SVD)
- Boltzmann machines
- Wake-Sleep Algorithm
- Information Theory
- Unsupervised Learning

Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such “autoencoder” networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer “encoder” network.
FCN: From MLP to Convolutional

U-Net
FCN
U-Net
Applications of U-Net

DeepLab

- Spatial Pyramid Pooling, 2016
  - Learn from different receptive fields
- Nowadays, DeepLab V3 +, 2017
Extra Credits for HW 2
U-Net Model

```
# Set random seed for reproducability
torch.manual_seed(577)
random.seed(577)

# The `init_features` defines how many channels the hidden layers contain.
# Usually, people would use 32 init features. Here, because our dataset
# is small, we use 4 init features. So, the channel size is about 32 / 4 = 8
# times smaller, and the training speed is about 8 x 8 = 64 times faster.
model = torch.hub.load('mateuszbuda/brain-segmentation-pytorch', 'unet',
in_channels=1, out_channels=4, init_features=4, pretrained=False)
```

Create Data Loader

@title TODO: create data loader

# TODO: create a data loader with batch size 8 and shuffling.
# Note 1: check `torch.utils.data.dataloader`.
# Note 2: You will need the `dataset` variable from the above block.
dataloader = None
Gradient Descent (Adam)

@title Define loss and optimizer

criterion = torch.nn.CrossEntropyLoss()

@title TODO: create an optimizer

# TODO: create an Adam optimizer with learning rate 0.001.
# Note: check out `torch.optim.Adam`
optimizer = None
Training

# For simplicity, we only train a few epochs. Good enough~
# Note that you might need 20 minutes to run this block.

# If you have enough time (or machine power), you can increase
# the number of epochs. It can improve your results.
# Don't turn off your monitor while running the code: if the
# machine falls asleep, the Colab will stop working.

num_epochs = 5

use_cuda = torch.cuda.is_available()
if use_cuda:
    model = model.cuda()

for epoch in range(num_epochs):
    losses = []
    if dataloader is None or optimizer is None:
        break  # NotImplementedError
    for x, y in tqdm.tqdm(dataloader):
        if use_cuda:
            x, y = x.cuda(), y.cuda()
        pred = model(x.float())
        loss = criterion(pred, y.long())

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        losses.append(loss.item())

    print("Epoch:", epoch, "Mean Loss:", np.mean(losses))
Other Applications of Autoencoder
我 靠 极 大 量 的 阅 读 来 弥 补 我 有 限 的 智 力 END
Towards Automated Reporting for Chest X-ray


Disease Detection

☑️

- Pneumothorax
- Pleural Effusion
- Mass

Disease Reporting

Findings: left apical small pneumothorax and small left pleural effusion remains. Unchanged nodular opacity right mid lung field.

Impression: removal of left chest tube with tiny left apical pneumothorax and small left pleural fluid.
Applications... Multi-Modality

The encode data and decode data can be...

- Image
- Video
- 3D Image
- Audio
- Text
- Signal
- More...
Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision
ORIGINAL ARTICLE

Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision

Haiguang Wen\textsuperscript{1,2}, Junxing Shi\textsuperscript{1,2}, Yizhen Zhang\textsuperscript{1,2}, Kun-Han Lu\textsuperscript{1,2}, Jiayue Cao\textsuperscript{2,3} and Zhongming Liu\textsuperscript{1,2,3}

\textsuperscript{1}School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA, \textsuperscript{2}Purdue Institute for Integrative Neuroscience, Purdue University, West Lafayette, IN 47906, USA and \textsuperscript{3}Weldon School of Biomedical Engineering, Purdue University, West Lafayette, IN 47906, USA

Address correspondence to Zhongming Liu, Assistant Professor of Biomedical Engineering, Assistant Professor of Electrical and Computer Engineering, College of Engineering, Purdue University, 206 S. Martin Jischke Dr, West Lafayette, IN 47907, USA. Email: zmliu@purdue.edu
Overview

- Subjects were in an MRI machine watching colorful videos that contained various object categories.
- fMRI showed activated areas of the brain as the subjects watched the videos.
- Encoding and decoding models were developed and evaluated for describing the bi-directional relationships between the CNN and the brain.
Details

- Studied how the brain decodes visual information.
- Acquired 11.5 hours of fMRI data from each of 3 human subjects watching 972 different video clips including diverse scenes and actions.
- Used AlexNet to extract hierarchical visual features from the movie stimuli.
- Five convolutional layers and three fully connected layers.
- Reduced the original 1000 categories to 15.
- Compared the outputs of CNN units to the fMRI signals.
- Trained encoding and decoding models for describing the relationship between the brain and the CNN.
Figure 7.

(a) 

stimulus 

extracted FM 

CNN 

fMRI 

estimated FM 

decoder 

FM = Feature Map
Figure 8.

Reconstruction of a dynamic visual experience. For each row, the top shows the example movie frames seen by 1 subject; the bottom shows the reconstruction of those frames based on the subject’s cortical fMRI responses to the movie. See Movie 1 for the reconstructed movie.
Findings

- The CNN recognizes the same object as the subject recognizes in the video from features in its penultimate layer.
- They correlate those features with brain voxels for each object.
- Then they can reconstruct the movie from the fMRI using a decoder.
- And also determine which object category from the fMRI.
Discussions:

Do the results mean human brains and artificial neural networks work in a similar way?
Discussions:

Do the results mean autoencoder can map brain activity to stimulus?
Pros vs Cons of This Study

Pros:

Cons: