Convolutional Neural Networks for Deep Learning: An Introduction

Abhishek Jain
PhD Candidate,
Electrical & Computer Engineering,
North Carolina State University

October 4, 2017
Table of Contents

1. Introduction
   - Artificial Neural Networks
   - Deep Learning
   - Convolutional Neural Networks

2. Data Sets

3. Convolutional Neural Networks Algorithm
   - Convolution
   - Thresholding
   - Pooling
   - Classification
   - Training and Testing

4. Online Demo

5. Useful References
Introduction: Artificial Neural Networks (ANN)

- **ANNs**: Algorithms inspired by biological neuron networks in the brain.
- **ANN** is comprised of *neurons* and *synapses* in sequential layers.
- Neurons contain *activation functions* to change their internal states, and produce an output.
- Synapses use *back propagation* to update weights, and hence *learn* the input-output model.
- Data is traversed multiple times to improve learning.
**Deep Learning**: Name given to types of machine learning algorithms that use feature extractions to learn a model of the input-output data.

Multiple interacting layers can be used to extract features, corresponding to different layers of abstraction (arrangement of complexity).

Each successive layer uses the output of the previous layer. Also, these layers (and their outputs) are usually hidden from the user, thus the name “Deep Learning”.

Extracted features can then be used for classification or simply feature detection.

Applications include:
- Image recognition (handwriting recognition, defect classification in manufacturing)
- Speech recognition (Siri voice search)
- Advertising (the recommendations you see on your Amazon page)
**Introduction: Convolutional Neural Networks (CNN)**

- **Convolutional Neural Networks**: A deep learning algorithm specifically designed for image classification.

- CNNs perform a **convolution** operation on the input images, where the **neuron connections** are inspired by the visual cortex of the brain, hence the name.

- CNNs can use various types of hidden layers:
  - **Convolution Layer**: Uses filters (patches of pixels) to extract image features.
  - **Thresholding Layer**: Introduces nonlinearities in features using a ReLU operation.
  - **Pooling Layer**: Reduces the feature dimensions.
  - **Fully Connected Layer**: Performs classification using features and class labels.

- Multiple convolution layers (along with their relu, pooling and fully-connected layers) are generally used to extract increasingly complex features.

- Rest of the talk will focus on explaining these above layers in detail.
Data Sets: MNIST and ImageNet

- **MNIST**: Modified National Institute of Standards and Technology.
  - 1000 images for each of the handwritten digits (also the classes) 0-9.
  - Size of each digit image is 28x28 pixels.
  - *Training* data contains 75% of images from each class, rest is treated as *testing* data.

- **ImageNet**
  - A million color images of 1000 different types of objects (hence 1000 classes).
  - Size of each image is 227x227x3 pixels.
  - The CNN *AlexNet* is pre-trained on ImageNet.

```matlab
% Load and Explore the Image Data
digitDatasetPath = fullfile(matlabroot,'toolbox','mnset',...  'nndemos','mnndatasets','DigitDataset');  
digitData = imageDatastore(digitDatasetPath,...  'IncludeSubfolders',true,'LabelSource','folderNames');
% Specify Training and Test Sets
trainingNumFiles = 750;  
rng(1)  
[trainDigitData,testDigitData] = splitEachLabel(digitData,...  trainingNumFiles,'randomize');
```
**CNN Algorithm Step 1: Convolution**

- The convolution layer applies a convolution operation to the input image.
- Different types of filters can be used to extract different features.
- These filters are *learned* by the CNN using the given input-output data.
- **Arguments**: Filter size, stride, number of filters.
CNN Algorithm Step 2: Thresholding

- Thresholding layer applies ReLU (Rectified Linear Units) to introduce nonlinearity in the feature maps generated by the convolution layer.

- ReLU function: $f(x) = \max(0, x)$

- Arguments: None.

- Other nonlinear functions such as the hyperbolic tangent ($f(x) = \tanh(x)$) can be used to increase nonlinearity.
CNN Algorithm Step 3: Pooling

- The pooling layer reduces the dimensionality of the rectified feature maps by downsampling the maps.
- The intuition is that the exact location of a feature is less important than its rough location relative to other features.
- Another advantage of pooling is that it makes the CNN invariant to small transformations and distortions.
- Pooling functions: max-pooling, average-pooling, sum-pooling.
- Arguments: Pool size, stride.

```
% Define the Network Layers
layers = [imageInputLayer([28 28 1], 'Name', 'input')
  convolution2dLayer(5, 20, 'Name', 'conv')
  reluLayer('Name', 'relu')
  maxPooling2dLayer(2, 'Stride', 2, 'Name', 'pool')
```

Rectified Feature Map
CNN Algorithm Step 4: Classification

- A fully connected (FC) layer consists of a ‘regular’ ANN with multiple hidden layers inside. Each node in the hidden layers is connected to every other node.

- The layer uses a separate softmax layer to classify the test data into one of the class labels.

- Aside from ANNs, other classification algorithms can be used such as Bayes classifier, SVM, Random forests etc.

- Arguments: Number of neurons.

```matlab
%% Define the Network Layers
layers = [imageInputLayer([28 28 1], 'Name', 'input')
    convolution2dLayer(5,20,'Name','conv')
    reluLayer('Name','relu')
    maxPooling2dLayer(2,'Stride',2,'Name','pool')
    fullyConnectedLayer(10,'Name','fc')
    softmaxLayer('Name','soft')
    classificationLayer()];
```
CNN Algorithm: Training and Testing

- **Training Step 0**: Divide the training set into smaller subsets of images, and randomly choose one of the subsets. This is called a **batch**.

- **Training Step 1**: Forward propagation with convolution, thresholding, pooling and classification operations to get some probability values for each class.

- **Training Step 2**: Back propagation to update filter weights, and FC layer weights with a gradient descent method. This completes one **iteration**.

- **Training Step 3**: Repeat Steps 1-2 for all **batches** to exhaust the training data set. This completes one **epoch**.

- **Training Step 4**: Do multiple **epochs** until the training accuracy stabilizes to get a trained CNN model.

- **Testing Step**: Each test image is passed through the entire network once, with the trained filter weights and FC layer weights.
CNN: Results for MNIST data

- Training and testing results on a single CPU.
- 98% accuracy is achieved with a single CNN layer.
- More layers can be used to improve results.

```matlab
% Specify the Training Options
options = trainingOptions('sgdm',... 'MaxEpochs',15,... 'InitialLearnRate',0.0001);

convnet = trainNetwork(trainDigitData, layers, options);

% Classify the Images in the Test Data and Compute Accuracy
YTest = classify(convnet, testData);
TTest = testData.Labels;

accuracy = sum(YTest == TTest)/numel(TTest);
```

Accuracy: 0.9908
In real-world applications, CNNs are generally trained on GPUs to obtain a reasonable execution time.

Various programming languages have developed packages for CNNs such as:
- Matlab: In Neural Network Toolbox
- Python: Theano, TensorFlow
- R: deepnet

Two of the most famous (and widely used) CNNs are AlexNet and VGG16. Their architectures are shown below.
CNN: Online Demo

Cool CNN Demo!
An Intuitive Explanation of Convolutional Neural Networks:  
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition:  
http://cs231n.github.io/convolutional-networks/

Convolutional Neural Networks (CNNs): An Illustrated Explanation:  
Thank You

I, for one, welcome our new cyber overlords
(because I’m going to program them!)