Image Classification using Convolutional Neural Networks

Suchismita Padhy
MIMS, UC Berkeley

February 17, 2017

Abstract

Classifying and annotating images is an important task in machine learning. Many algorithms have been proposed for these tasks, based on features such as color, texture, and shape. Success of these algorithms is dependent on the selection of features. Deep learning models are widely used to learn abstract, high level representations from raw data. In this project, we explore Convolutional Neural Networks for the task of image classification on some standard image datasets such as MNIST handwritten digits and CIFAR-10.

1 Introduction

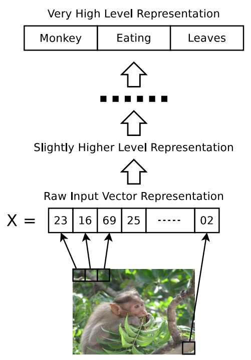
The task of grouping images into semantically meaningful categories such as concepts or classes is commonly known as image classification. Several machine learning approaches can be used for image classification. In most of these approaches low level features such as, a representation based on color, texture, shape, is extracted from an image and then the image is categorized to a unique class.

The performance of image classification systems is largely dependent on the effectiveness of the features used because different representations can express or hide the explanatory factors behind the data. Learning a representation of data that makes it easier to extract useful information is highly desirable for developing a good classification framework [1]. Deep learning techniques try to learn abstract and useful representations from raw data using multiple layers of nonlinear processing. These learned representations can be more effective for tasks such as classification and annotation.

1.1 Feature Hierarchy

Deep learning models have been shown to be suitable for learning representation from raw data. Deep learning techniques exploit the idea of hierarchical explanatory factors. For example, consider the image shown in Figure 1. The pixel intensity values of pixels in the image can be used to obtain a low level feature representation. These low level features are combined to form more abstract representations such as edges and objects that describe the image. The way humans describe an image is perceived as a very high level representation of the image.

Some of the widely used deep learning models involve artificial neural networks with many hidden layers. The deep belief networks and deep Boltzmann machines first use the principles of energy based models (EBMs) to estimate on the weights of the connections between different layers and then use the error backpropagation method for fine tuning. The deep convolutional neural networks use the convolution operation to extract features from different sub-regions of an image to learn a representation.


Very High Level Representation
Monkey
Eating
Leaves
Slightly Higher Level Representation
Raw Input Vector Representation
X =
23
16
69
25

02

Figure 1: Taking the pixel information of an image's raw input, a feature vector representation of the image is obtained. These low level features are combined in each layer of hierarchy to form higher level representations of the input image. A human description of the image, i.e., a monkey eating leaves, is a very high level representation of this image.

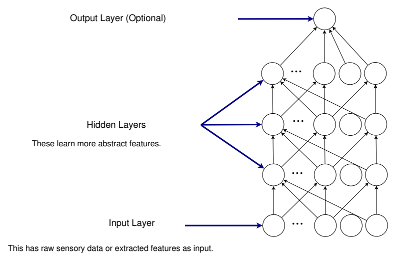

Output Layer (Optional)
Hidden Layers
These learn more abstract features.
Input Layer
This has raw sensory data or extracted features as input.

Figure 2: Raw pixel values or extracted features are given as input. The input layer is followed by multiple hidden layers that learn increasingly abstract representations.

2 Convolutional Networks

The convolutional networks is a variation of a multi-layered feed forward neural network, where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field [6]. From Hubel and Wiesel's early work on the cat's visual cortex [2], it is known that the cells within the visual cortex are sensitive to small sub-regions of the input space, known as a receptive field, and are tiled to cover the entire visual field. These filters are local in the input space and are thus better suited to exploit the strong spatial correlations present in natural images. A convolutional network is composed by stacking multiple convolutional layers and subsampling layers, followed by a set of fully connected layers.

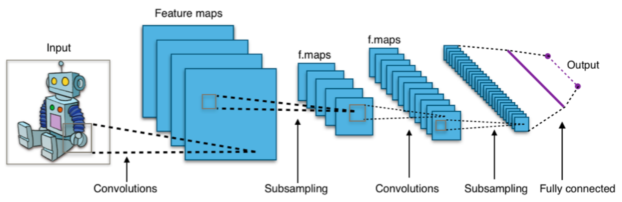


Figure 3: The structure of a deep convolutional network.

In a convolutional network, the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. The units in the m^{th} layer are connected to a local subset of units in the $(m-1)^{th}$ layer, which acts like spatially contiguous filters. The first layer is connected to the input. Figure 3 shows the typical structure of a deep convolutional network.

2.1 Convolution

A convolutional layer consists of a rectangular grid of neurons. Each neuron takes inputs from a rectangular section of the previous layer. The weights for this rectangular section are constrained to be the same for each neuron in the convolutional layer. Constraining the weights makes it work like many different copies of the same feature detector applied to different positions. This constraint also helps to keep a check on the number of parameters. The output of a neuron in the convolutional layer, l for a filter of size $(m * n)$ is given by

$$s_{ij}^{l} = f(\sum_{x=0}^{m}\sum_{y=0}^{n} w_{xy} s_{(x+i)(y+j)}^{(l-1)}) \tag{1}$$

where $f(x) = \max(0, x)$, which is known as the rectifier function. The nodes that use the rectifier function are referred to as Rectified Linear Units (ReLU). In terms of training time, a network with ReLU nodes performs significantly better.

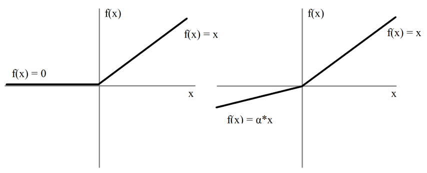


Figure 4: (A) ReLU activation function $f(x) = max(0, x)$. (B) Leaky ReLU activation function. When $x < 0$, we use a small negative slope.

2.2 Pooling

A convolutional layer is optionally followed by a pooling layer. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples each block to produce a single output from that block. The pooling layer can compute the average, or maximum, or a linear combination of the outputs of neurons in the block. Pooling helps the network achieve small amount of translational invariance at each level. Also, it reduces the number of inputs to the next layer.

2.3 Fully connected layers

After several sets of convolutional and pooling layers, multiple fully connected layers are included in a DCNN. Every neuron in a fully connected layer is connected to all the neurons in its previous layer.

2.4 Regularization

Regularization is used in the fully connected layers to prevent overfitting and reduce the training time. The two most commonly used regularization techniques are as follows:

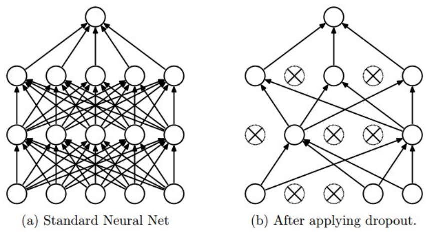


Figure 5: Dropout regularization applied to fully connected layers with a keep probability 0.5)

2.4.1 Dropout Regularization

- The output of each hidden neuron in the fully connected layers is set to zero with a particular probability.
- The 'dropped out' neurons do not participate in backpropagation, ensuring that each time only a subset of the neurons are active.

2.4.2 Dropconnect Regularization

- A randomly selected subset of weights within the fully connected layers is set to zero.
- Each unit receives input from a subset of units in the previous layer.

2.5 Softmax classifier

The final layer of this network is a softmax layer that predicts the class score for each of the classes. The softmax function is defined as follows:

$$softmax(x_j) = \frac{e^{x_j}}{\sum_j e^{x_j}}$$

3 Experiments and Results

For these experiments, we use a NVIDIA GPU with CUDA. The GPUsare capable of fast and parallel operations on matrices and double precision floating numbers. This gave me a lot of improvement in terms of running time. However, the amount of memory available on the GPU is a bottleneck.

For the experiments, I used a network with two convolutional layers, which were each followed by a pooling layer. Finally, two fully connected layers were used. Figure 6 shows the architecture of the network used.

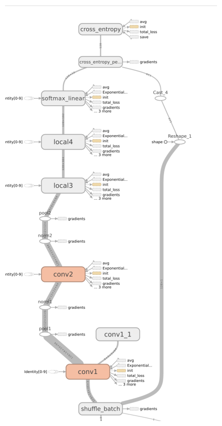


Figure 6: A part of the computation graph used by Tensorflow for the classification task on CIFAR 10 images.

3.1 Datasets used

MNIST [5] is a set of handwritten digits having 60,000 training examples and 10,000 test examples, belonging to ten classes from 0 to 9. The digits are size-normalized and centered in a fixed size (28*28) image. Figure 7 shows some randomly selected samples from MNIST dataset. Each image contains only one digit.

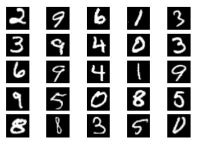


Figure 7: Randomly selected examples from MNIST dataset

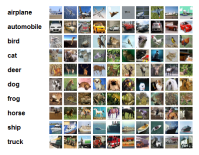


Figure 8: Randomly selected examples from CIFAR-10 dataset

CIFAR-10 [3] is a labeled subset of the 80 million tiny images dataset. It has 50,000 training and 10,000 test images divided into 10 classes, where each image is of dimension 32*32. This dataset is widely used for several deep learning tasks owing to its huge number of training examples and small size of individual images. Figure 8 shows some randomly selected samples from CIFAR-10 dataset.

3.2 Results

Figure 9 shows some randomly selected predictions by the deep convolutional network with Dropout on a random subset of test data taken from CIFAR-10 dataset. The label on the bottom of each image shows the correct label for that image, and the five labels on the right to the bar chart show the five top most labels predicted by the model arranged in sorted order of posterior probability.

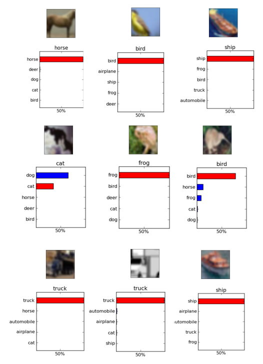


Figure 9: Randomly selected examples from MNIST dataset

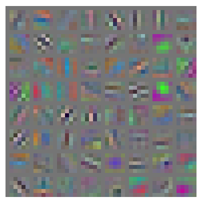


Figure 10: Features captured at the first convolutional layer for CIFAR-10.

Dataset	Regularization	Accuracy
MNIST	None	99.19 %
MNIST	Dropout	99.22 %
CIFAR-10	None	87.5 %
CIFAR-10	Dropout	89.1 %

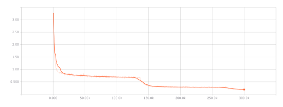


Figure 11: Total loss for CIFAR10 over iterations.

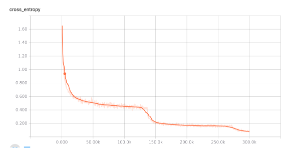


Figure 12: Crossentropy for CIFAR-10 over iterations.

4 Future Works

I plan to use leaky ReLu and pReLu activation functions to see how they compare to the standard ReLu units. For this project, I was unable to train the model on large image datasets because of the computational complexity. I want to train a deeper network on large images to observe how it performs on a more complex task.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[3] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Neural Information Processing System*, volume 22, pages 1106–1114, 2012.

[5] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. `http://yann.lecun.com/exdb/mnist/`, 2010.

[6] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.