Implementation of Striving for Simplicity: The All Convolutional Net

Aruna Neervannan

University of Illinois, Urbana Champaign

Abstract

This project is an implementation of The All Convolutional Net neural network built for object recognition using the same principles: Alternating convolution and max-pooling layers followed by a small number of fully connected layers. **Keywords** CIFAR-10, Neural Network, CNN, Image Processing, Strided convolutions

1 Introduction

The term convolutional network (CNN) is used to describe an architecture for applying neural networks to two-dimensional arrays (usually images), based on spatially localized neural input. They represent an interesting method for adaptive image processing, and form a link between general feedforward neural networks and adaptive filters. Two dimensional CNNs are formed by one or more layers of two dimensional filters, with possible non-linear activation functions and/or down-sampling. Conventional neural network error minimization methods may be used to optimize convolutional networks in order to implement quite powerful image transformations. Because of the low number of free parameters, training of CNN requires far less computational effort than training multilayer perceptron.

2 Purpose

The vast majority of modern convolutional neural networks (CNNs) used for object recognition are built using the same principles: They use alternating convolution and max-pooling layers followed by a small number of fully connected layers (e.g. Jarrett et al. (2009); Krizhevsky et al. (2012); Ciresan et al.). Within each of these layers piecewise-linear activation functions are used. The networks are typically parameterized to be large and regularized during training using dropout.

A considerable amount of research has over the last years focused on improving the performance of this basic pipeline. Among these two major directions can be identified. First, a plethora of extensions were recently proposed to enhance networks which follow this basic scheme. Among these the most notable directions are work on using more complex activation functions (Goodfellow et al., 2013; Lin et al., 2014; Srivastava et al., 2013) techniques for improving class inference (Stollenga et al., 2014; Srivastava & Salakhutdinov, 2013) as well as procedures for improved regularization (Zeiler & Fergus, 2013; Springenberg & Riedmiller, 2013; Wan et al., 2013) and layer-wise pre-training using label information (Lee et al., 2014).

Second, the success of CNNs for large scale object recognition in the ImageNet challenge (Krizhevsky et al., 2012) has stimulated research towards experimenting with the different architectural choices in CNNs. Most notably the top entries in the 2014 ImageNet challenge deviated from the standard de-

sign principles by either introducing multiple convolutions in between pooling layers (Simonyan & Zisserman, 2014) or by building heterogeneous modules performing convolutions and pooling at multiple scales in each layer (Szegedy et al., 2014).

3 Motivation

Convolutional neural networks (CNNs) have been applied to visual tasks since the late 1980s. However, despite a few scattered applications, they were dormant until the mid-2000s when developments in computing power and the advent of large amounts of labeled data, supplemented by improved algorithms, contributed to their advancement and brought them to the forefront of a neural network renaissance that has seen rapid progression since 2012. Image classification forms the basis for other computer vision tasks such as localization, detection, and segmentation (Karpathy, 2016). Although the task can be considered second nature for humans, it is much more challenging for an automated system. In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to overcome these challenges. In this implementation I explore a simpler implementation to handle the classification of the CIFAR-10 dataset.

4 Data

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

4.1 ZCA Whitening of Images

Whitening, or *sphering*, is a linear transformation that converts a d-dimensional random vector $x = (x_1,...,x_d)^T$ with mean $E(x) = \mu = (\mu_1,...,\mu_d)^T$ and positive definite dxd covariance matrix $var(x) = \Sigma$ into a new random vector

$$z = (z_1,.....z_d)^T = W_x$$

of the same dimension d and with unit diagonal "white" covariance $var(z) = I$. The square dxd matrix W is called the **whitening matrix**.
Whitening can be viewed as a generalization of standardizing a random variable which is carried out by

$$z = V^-(1/2)x$$

where the matrix $V = diag(\sigma_1^2,....\sigma_d^2)$ contains the variances $var(x_i) = \sigma_i^2$. This results in $var(z_i) = 1$ but it does not remove correlations. Often, standardization and whitening transformations are also accompanied by mean-centering of x or z to ensure $E(z) = 0$, but this is not actually necessary for producing unit variances or a white covariance.
Why use stride of 1 in CONV? Smaller strides work better in practice. Additionally, as already mentioned stride 1 allows us to leave all spatial down-sampling to the POOL layers, with the CONV layers only transforming the input volume depth-wise.
Why use padding? In addition to the aforementioned benefit of keeping the spatial sizes constant after CONV, doing this actually improves performance. If the CONV layers were to not zero-pad the inputs and only perform valid convolutions, then the size of the volumes would reduce by a small amount after each CONV, and the information at the borders would be washed away too quickly.

5 Model Architecture

5.1 Convolutional Layer

The primary purpose of Convolution is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

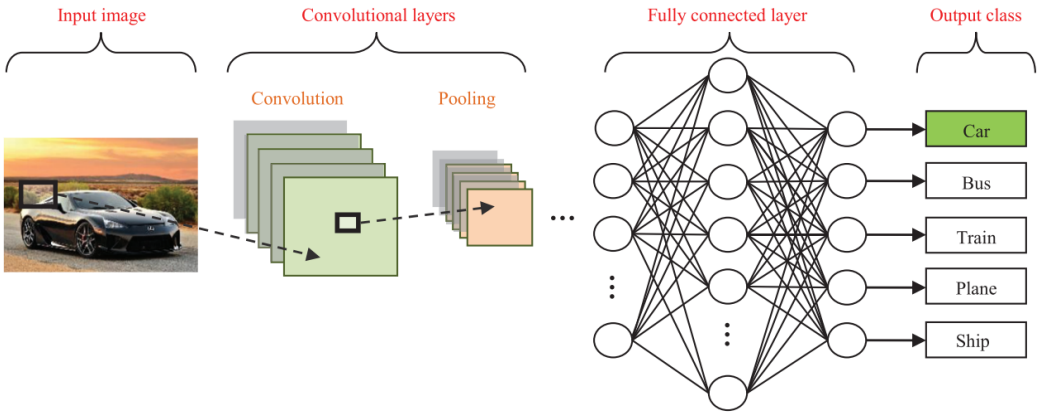

Figure 1: Convolution Neural Network for image processing

Table 1: The three base networks used for classification on CIFAR-10 and CIFAR-100.

Model		
A	B	C
Input 32 × 32 RGB image		
5 × 5 conv. 96 ReLU	5 × 5 conv. 96 ReLU 1 × 1 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU
3 × 3 max-pooling stride 2		
5 × 5 conv. 192 ReLU	5 × 5 conv. 192 ReLU 1 × 1 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
3 × 3 max-pooling stride 2		
3 × 3 conv. 192 ReLU		
1 × 1 conv. 192 ReLU		
1 × 1 conv. 10 ReLU		
global averaging over 6 × 6 spatial dimensions		
10 or 100-way softmax		

- **Stride**: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

- **Zero-padding**: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution

- **Convolutional with stride 1** Smaller strides work better in practice. Additionally, as already mentioned stride 1 allows us to leave all spatial down-sampling to the POOL layers,

with the CONV layers only transforming the input volume depth-wise.

- **Padding** In addition to the aforementioned benefit of keeping the spatial sizes constant after CONV, doing this actually improves performance. If the CONV layers were to not zero-pad the inputs and only perform valid convolutions, then the size of the volumes would reduce by a small amount after each CONV, and the information at the borders would be washed away too quickly.

- **Pooling** Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

- **Max Pooling** In case of Max Pooling, we define a spatial neighborhood (for example, a 22 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

5.2 Fully Connected Layer

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term Fully Connected implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

6 Results

The table below shows the results from running the CNN models for 100,000 iterations.

Model	Train Acc	Test Acc	Iterations
Model A	98.44	92.19	100,000
Model B	97.66	88.28	100,000
Model C	96.09	93.75	100,000

7 Summary and Future work

The results clearly show that the simpler CNN with strided convolutional achieve a high accuracy on the CIFAR-10 dataset. The accuracy is considerably higher than the Tensorflow CIFAR-10 tutorial accuracy.
Future work in this area would involve hyperparameter tuning on regularizers, optimizers, learning rates, adjusting the dense dimensions of ReLU layers and calibrating the CNN layers.

References

[1] STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET (https://arxiv.org/pdf/1412.6806.pdf)

[2] Goodfellow, Ian J., Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. In ICML, 2013

[3] Gulcehre, C aglar, Cho, KyungHyun, Pascanu, Razvan, and Bengio, Yoshua. Learned-norm pooling for deep feedforward and recurrent neural networks. In ECML, 2014.

[4] Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. In ICLR: Conference Track, 2014.

[5] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In arxiv:cs/arXiv:1409.1556, 2014

[6] Kingma, D., Ba, J. (2014). Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR)

[7] WZeiler, Matthew D. and Fergus, Rob. Stochastic pooling for regularization of deep convolutional neural networks. In ICLR, 2013.

[8] Wan, Li, Zeiler, Matthew D., Zhang, Sixin, LeCun, Yann, and Fergus, Rob. Regularization of neural networks using dropconnect. In International Conference on Machine Learning (ICML), 2013.

[9] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In NIPS, pp. 11061114, 2012.