Convolutional Neural Networks in Image Recognition

Sushant Kumar Singh,

Application Developer at Oracle, Hyderabad, India

Abstract— In the last few years, we have witnessed an exponential growth in research activity into the advanced training of convolutional neural networks (CNNs), a field that has become known as deep learning. This has been triggered by a combination of the availability of significantly larger data sets, thanks in part to a corresponding growth in big data, and the arrival of new graphics-processing-unit (GPU)-based hardware that enables these large data sets to be processed in reasonable timescales. Suddenly, a wide variety of long-standing problems in machine learning, artificial intelligence, and computer vision have seen significant improvements, often sufficient to break through long-standing performance barriers.

Index Terms — Machine Learning, Deep Learning, Convolutional Neural Networks, Object Detection, Mobile Devices.

I. INTRODUCTION

To store and analyze analog signals (e.g., voice and biological signals) on a computer, we must first convert these inputs into a digital form using analog-to-digital conversion. This transforms the data from a continuous to a discrete space. Some information is lost in the process [2]. Often that information isn't vital to understanding the underlying analog signal, but in some instances, we could lose critical data, such as high-frequency information. In digital processing, we often refer to a piece of data, such as a picture, as a sample. It is convenient, but computationally expensive, to represent these samples in a high-dimensional space where each unit (or pixel, in the case of images), is considered as being located on a specific axis with the range of possible values being the size of that axis (e.g., 0–255 in the case of an 8-b colour-channel in an image). An image that is 100 × 100 pixels would be represented as a vector that is a point in a 10,000-dimensional space. We call this space the feature space. Since even the most powerful computers can have trouble with such high-dimensional space, we try to reduce the number of dimensions to just those that are critical to a task or to change the way these features are represented. In the traditional pattern recognition approach, to perform a given task, we separate our process into two steps: feature generation and feature selection. The former generates new features from the pixel space, while the latter reduces the dimensionality of the feature space. Examples of feature generation include morphological, Fourier, and wavelet transforms, which create more useful features for specific tasks [2].

II. PROBLEM FORMATION

In this world today when we speak of computers matching the capabilities of human beings, computer vision plays a very important role. How accurately a computer can identify objects within an image is something of our interest. There have been various attempts using multiple methods to address this problem of object identification .CNN is one such attempt to address this problem.

III. TRAINING THE NETWORK

The process of teaching our network is called

training. When we train a network, what we are formally doing is fitting a network to our training (data) set. The training set is the sample information that we think is sufficiently representative that our network should be able to learn from it. The thing we want our network to learn to do is called the task [2]. When training ANNs, we want the network to perform well at a given task on the basis of unseen information. When an ANN is not trained sufficiently to do this, we call it under fitting, which means that the network did not sufficiently learn the training set. The opposite of this is called overfitting, where the network learns the training set so well that it cannot effectively be applied to data that it has not seen before. These concepts can be best understood by referring to Figure 1, which illustrates a simple two-dimensional data set [1].

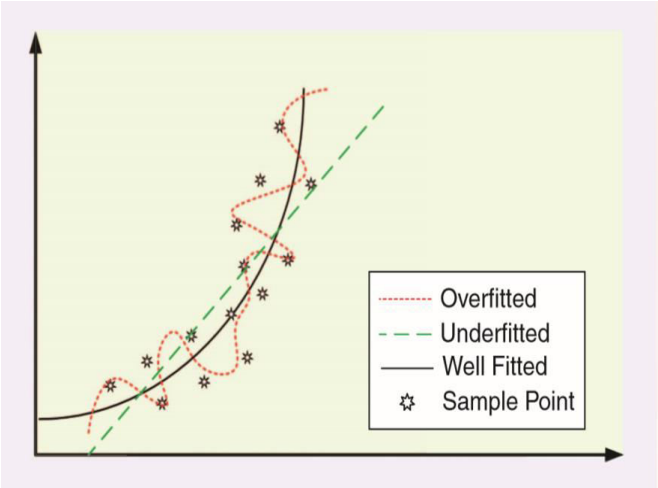

Figure 1. Training with a 2D data set

IV. CONVOLUTIONAL LAYER

In general, for an n-dimensional signal, the convolutional layer is an n- or (n + 1)-dimensional feature space mapping with n + 1 or n + 2 dimensional kernels (filters). For example, given a two-dimensional image, the convolutional layer would be 3-D with a 4-D kernel. In this case, the four dimensions of the kernel correspond to

->the width of the input
->the height of the input
->the number of channels of the input
->the number of channels of the output.

In Figure 2, you can see two different convolutional layers. The k channel layer on the left side is mapped to a p channel layer on the right side using a 4-D kernel. This kernel is shown using different 3-D kernels with different colors. Convolution is necessary since we can make a unique filter for every image. Hence, convolution is preferred over any other mathematical operation [3].

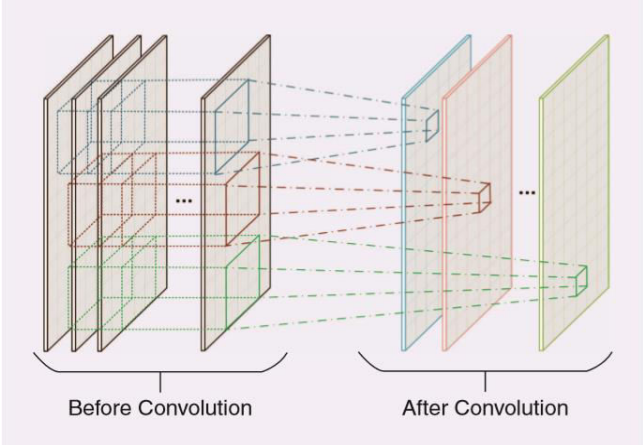

Figure 2.A 4D filter maps one 3D space to another 3D space using convolutions.

V. POOLING LAYER

Pooling is an operation that accepts a pool of data values as input and generates one value from them to be passed to the next layer. For example, this operation could be mean or maximum of the input values. There are two important purposes for pooling operations. One is reducing the size of the data space to reduce overfitting, and the other is transition invariance. A pooling layer is performing the pooling operation on its inputs. Figure 3 shows how a pooling operation is applied to a one-channel input to reduce the dimensionality of the dataspace. In Figure 3, we

have a 3 × 3 pooling operation applied to a 12 × 6 one-channel feature space. The most frequently used pooling operation is max-pooling, wherein the maximum value of the features in the pool is selected to be mapped to the next layer [2].The importance of this layer is described in the next example. In generic deep neural networks (described in the "Generic Deep Neural Networks" section), a dense, fully connected layer emerges from the convolutional layers. For example, consider a convolutional layer with ten channels and a 100 × 100 feature space. Placing a fully connected layer after it would result in computing 100,000 weights from this layer to each neuron in the dense layer, which requires significant memory and computation resources. We could use a pooling layer to reduce resource demands. Additionally, without a pooling layer, a network this big might suffer from overfitting, especially if there is not enough representative data in the training set. Pooling also helps provide transition invariance by helping each kernel cover more space [2].

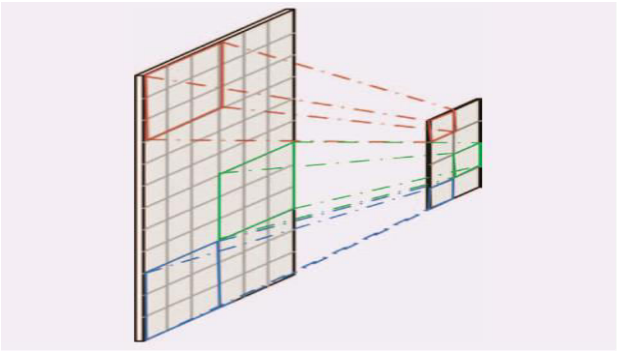

Figure 3. Pooling reduces size of feature space

VI. GENERIC DEEP NEURAL NETWORK

Generic deep neural networks are usually made of one or more convolutional layers, wherein each convolutional layer is usuallyaccompanied by a pooling (max-pooling) operation. One can also use bigger strides in the convolution layer to reduce the data dimensionality (the stride of a convolution operation is the number of the pixels the kernel window is sliding before calculating the convolution in each location). In generic deep neural networks, the convolutional layers are usually followed by one or more dense fully connected layers (Figure 4). The rectified linear unit is the most common activation function used in these networks. The last layer is typically taking advantage of other nonlinearities based on the task [3].

VII.

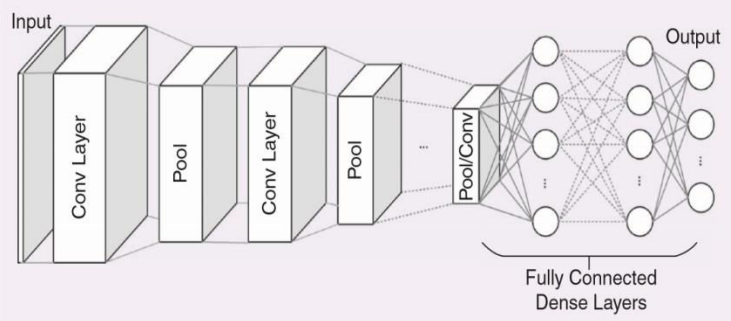

Figure 4.A generic deep neural network with fully connected dense layers

VII. FULLY CONVOLUTIONAL NETWORKS

Fully convolutional networks are deep neural networks in which all of the layers are convolutional, pooling, and unpooling layers, wherein the pooling and unpooling layers are usually placed between two convolutional layers. The output of a fully convolutional network is the same type as its input. For example, if the input is a k-channel image, the output of the network could be a p-channel image but not something else entirely [4].

VIII. RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) are designed to operate on a sequence of the data as input, such as a sequence of frames from a video. This can be considered as a system with memory that can remember the input at the previous stage and make decisions for the present input based on the sequence of previous data. The memory of an RNN network is known as the hidden state of the

network. The hidden state of the network is updated based on the current state. The input to the network and the output of the RNN is calculated based on the state of the network at each sequence. RNNs have had great success in speech and video scene recognition, where they are often combined with convolutional layers [3].

IX. RESULTS

We have explained how CNN can be used for object detection within images. We have used an open source API named Tensorflow for testing the accuracy of computer vision. Surprisingly, today computers are at par with humans in object recognition within images. Humans have an error rate of 5% whereas computers have an error rate of 3% [4].

X. CONCLUSION

Deep learning is being used today in our cell phones, cars, and tablets and computers. It has pushed the boundaries of what is possible for tasks such as image segmentation , object detection , face recognition , voice analyzing , emotion detection , and gender recognition [4]. Why has deep learning suddenly catalyzed research across so many fields? It is a combination of many factors: the recent emergence of highly affordable high-density, GPU-based computational hardware has provided the engines to process very large data sets and implement the advanced training methodologies required to develop accurate CNNs; the widespread availability of GPUs in today's devices, coupled with cloud-based data processing services, provides the means to apply these CNN architectures to everyday applications such as voice or image processing; big data provides the fuel to drive research activity and refine results to the point where deep learning solutions typically outperform even the best of human-designed pattern recognition tools [4].

XI. REFERENCES

[1] Y. Chauvin, "Generalization performance of overtrained backpropagation networks," in *Neural Networks*, New York: Springer, 1990, pp. 45–55.

[2] S. Theodoridis and K. Koutroumbas, "*Pattern recognition and neural networks*," in *Machine Learning and Its Applications*, New York: Springer, 2001, pp. 169–195.

[3] X. L. Zhang and J. Wu, "Deep belief networks based voice activity detection," *IEEE Trans. Audio.Speech.Lang. Processing*, vol. 21, no. 4, pp. 697–710, 2013.

[4] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, "Deep Learning for facial expression recognition: A step closer to a smartphone that knows your moods," in Proc. *IEEE Int. Conf. Consumer Electronics*, 2017, pp. 236–239.