Convolutional Neural Network and Its Application in Handwritten Digit and Traffic Sign Recognition

Pengxiang Jia

University of Victoria, Victoria BC Canada
Percy6995@gmail.com

Abstract
The convolution neural network (CNN) is an application of deep learning in computer vision. It has a strong feature extraction ability and has become an important component in deep learning. It has good achievement in image classification tasks. This article summarizes the background and concepts of CNN, talking about the application of CNN on the classification of MNIST handwritten numbers and traffic signs.

Keywords
Convolutional Neural Network (CNN), Deep Learning, MINIST Handwritten Digit Recognition, Traffic Sign Recognition

1. Introduction

Machine learning is utilizing computers to perform tasks to a level of human capability by searching the algorithm in the training process [1]. In traditional machine learning, feature extraction is done by engineers with domain knowledge (PCA), and makings precise and correct decisions on features is difficult [2]. Deep learning makes an alternative to learn feature extraction within the whole training process. The structure of layers enables this learning ability of its features [3].

The multilayer network's feature extractions in the convolution neural network (CNN) are learned in its training: every layer constructs its feature for the best classification success rate. Every hidden layer's features are built on the output of features from the last layer, and then the output of this layer is again used for the next layer. In this way, CNN has a strong capability on defining complicated features in objects for recognition [3, 5, 6].

Multilayer network's applications are very wide. Benefitted from its strong expressive power, it is used not only in image classification, but also in object detection, computer vision, language recognition, prediction, etc. [4].

CNN is a kind of neural network. It has three advantages that make it particularly outstanding: First, it has fewer parameters to study, because it uses less weight due to weight sharing. Secondly, its training time is much reduced. Third, it has strong expressive power – strong power for object recognition and learning capability, although there are fewer parameters used [5].

This survey introduces the usual composition and the training process of CNN. The second part describes CNN application on MINIST handwritten digit which is a well-known benchmark for image recognition algorithms [6, 7]. Starting from the third part, we expand on the application of CNN on traffic signs. Across different applications of CNN, the training and application processes are similar, but based on training on different datasets and different purposes, the CNN architectures and training process differ in various degrees [8, 9, 10]. LeNet-5, VGG-13, MTCNN, RCNN, Fast RCNN, and Faster RCNN are known reported architectures [11].

This survey emphasizes the base architectures of CNN and the differences of CNN in different application circumstances, in terms of small differences such as operations, process, and architectures, and their comparison.

2. Convolutional Neural Network

The input to convolutional neural network is a $1 \times k$ vector $X = (x_1, x_2, x_3, \ldots, x_k)$. If we are going to train the network with images, the input to the network will be matrices of pixels of the images.

We need to transform the square matrix of pixels to a vector of pixels of size $1 \times k$, so that it can be input into the network. We can connect the head of the second row to the tail of the first row of the matrix.

CNN on image classification is known to be effective, involving few parameters and fast training. Nowadays, CNN has been applied to many recognition and classification tasks in many distinctive domains. Classic CNN has three parts: the convolutional layer, the max-pooling layer, and the fully connected layer. Normal fully connected can achieve the performance of CNN, but comparatively, CNN needs fewer parameters and less training time.

Convolutional layers are those layers specifically used for pattern recognition in neural networks. Every single neuron can recognize a particular pattern, which is called the receptive field or filter. The pattern is a matrix of weights that directly represents the weights assigned on. It is 3x3 in size in Figure 1.

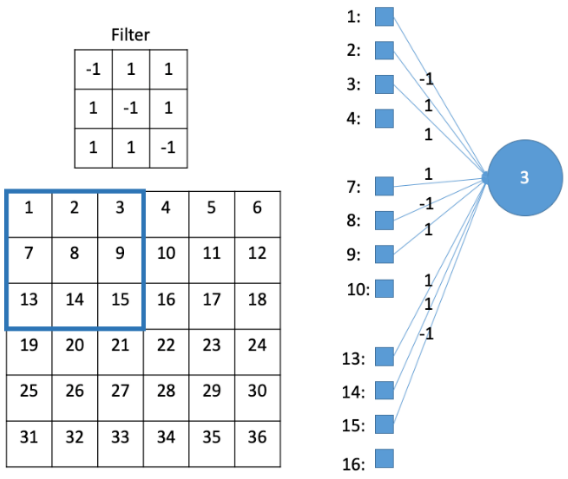

Figure 1: Convolutional Layer and Filter

The weights enclosed in the square boundary is corresponding to the 1st, 2nd, 3rd, 7th, 8th, 9th, 13th, 14th, and 15th connection from the past layer to the neuron in the next layer. The convolution process is to overlay the filter on the image. For example, in Figure 1, the filter is at the top left corner, and the convolution calculation is executed. The 9 filter values and 9 overlaid pixel values (e.g., the 9 values in the square boundary.) are timed and summed together, as in expression (1).

$$\sum_{i=1}^{9} p_i f_i \tag{1}$$

If the convolution result is a large value, that indicates a pattern is found; otherwise, it is a small value. A pattern, for instance, a grain of a kind of material, can appear anywhere on the image. To search for the pattern, we must slide the filter in a step size of a pixel and go through all pixels in the two dimensions of the image. Each time when we slide the filter horizontally or vertically, we calculate the convolution of the filter values with the image pixels. A very good advantage of having a filter and sliding it is *weight sharing*. Weight sharing can be understood by seeing the filter represented in CNN architecture. In the actual CNN representation of the filter, each time when a filter does the convolution operation with 9 pixels on the image, we connect a neuron from the convolutional layer to 9 neurons on the previous layer, applying the weights in the filter. Every neuron in the convolutional layer should be connected to different sets of 9 neurons of the previous layer. Each neuron in the convolutional layer reflects one step of the sliding filter across the image. The filter has static weights during the sliding, and the filter's weight will not be updated until all the convolution operations are done, so in the CNN representation, the weights are shared between different convolution neurons. Compared to a fully connected layer, weight sharing has fewer different weights and fewer connections, thus reducing computation in training.

In Figure 2, we are searching for a pattern where the pixel values on diagonal are 1, and other values are -1. The first filter convolutes with the first 3x3 values and yields a 3. The second filter convolutes with the second 3x3 values yields a -1. The 3 vs -1 indicates that the first step has more similarity to the pattern than the second one, which is also called higher activation of the neuron.

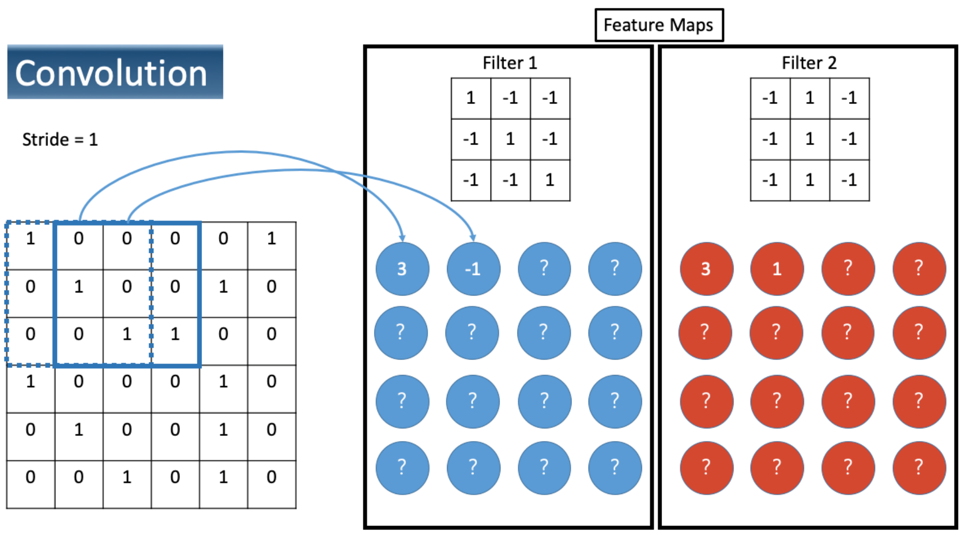

Figure 2: Convolution Operati- on

After testing every 3x3 area in the image against the filter, we get a matrix of the convolution results, which is also called the *feature map*. For one image, we usually set up not only one filter, but multiple filters. With multiple filters, we can detect different patterns on the image. The size of the feature map is one dependent on the filter size, and because we only have the same size for different filters, we get the same-size feature maps for different filters. The feature maps pile up and become the output of the convolutional layer, which is also the input for the next layer, as illustrated in Figure 3.

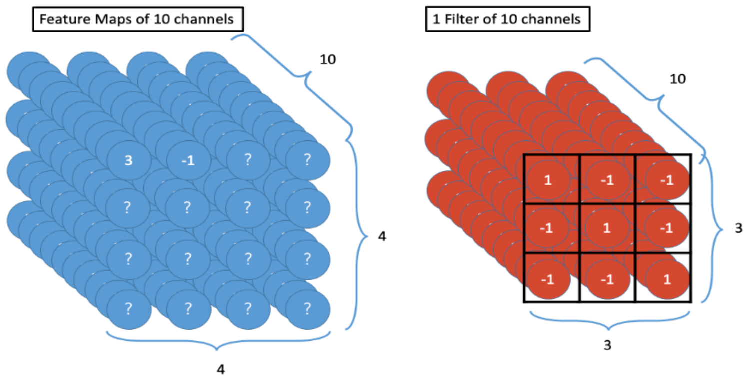

Figure 3: Feature Maps

The multiple convolutional layers architecture allows detecting complicate features. Taking feature maps input to another convolutional layer, the filters will do convolution operations with feature maps, which is the same as it has been done with the original image. The features detected on the feature map are a combination of simple features presented on the original images, thus they are complicated features. For example, a 3x3 filter can only detect a pattern as simple as a line on the diagonal of the filter, while the feature map is a matrix of indicators of the degree of activation of simpler features from the last convolutional layer, a pattern in the feature map is a pattern about the combination of the previous features with their positional information and existence information defined in the filter. For example, a simple wave-shape feature can be combined to be the texture of the rubber wheel or the actual sea waves.

If an image or feature map is input, the filter will yield another feature map, which is explained above. The number of filters determines the number of output feature maps. A filter is not only 2D, but can also be 3D. A 3D filter processes multiple layers of feature maps together. For example, a 3x3x3 filter processes 3 layers of feature maps, and 9 points in a square matrix from a layer are processed each time.

The max pooling layer is another important component of CNN. By inspection, the human inception of a picture is only a little affected by the resolution of the picture, so if we drop out every next row and column of the image, we are still able to identify the object(s) in the image. This drop-out operation is called *max-pooling*.

In CNN, the max-pooling in the max-pooling layer is a little different. The picture is first divided into squares of the same size, then the largest value or the average value of each square will be found. The nominated values are combined to be the output while preserving relative positions.

A complete CNN comprises convolutional layers and max-pooling layers, appearing alternatively. Lastly, it is connected by a normal fully connected network.

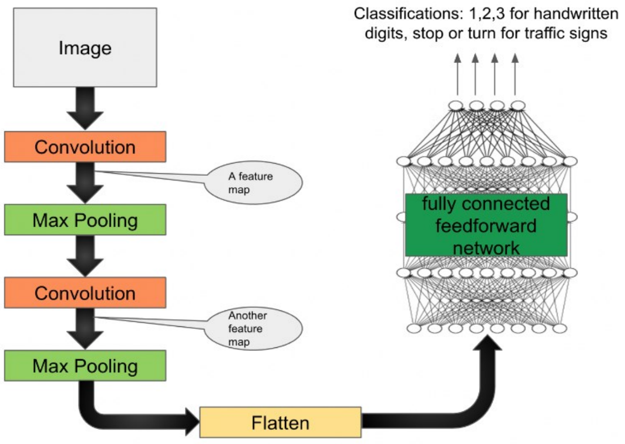
Image
Convolution
A feature map
Max Pooling
Convolution
Another feature map
Max Pooling
Flatten
Classifications: 1,2,3 for handwritten digits, stop or turn for traffic signs
fully connected feedforward network

Figure 4: Full CNN Structure

3. Application of CNN on Handwritten Digits

MINIST is a handwritten digit collection. It has tens of thousands of training sets and testing sets of 28x28 pictures. The application of handwritten digits is a good example for learning CNN’s characteristics and methodology. The LeNet-5 is a CNN architecture that is good for handwritten digits, the activation function is tanh(). It has 5 convolutional layers and 5 max-pooling layers, and is connected with a 3-layer fully connected network. The fully connected network outputs 10 signals, each representing the numbers 0 to 9.

Take LeNet-5 as an example for analysing performance. The number of convolutional layers, the number of filters, and the filter size will directly affect the training results. Other relevant factors are what activation function is used, whether it is connected to fully connected layers, and the quantity of training set; in particular, the quantity of training set has a large influence.

If the activation function of LeNet-5 is changed from tanh() to sigmoid, even cutting off the full connected layers, the performance of CNN is not significantly reduced. However, in the training process, the error curve will be even more stable and smooth. This probably indicates that sigmoid is a better activation function for tasks of handwritten digits.

If we decrease the number of convolutional filters by a small amount, the error plateau will be a little higher, but CNN will reach the plateau much faster and the training time is much reduced. If we increase the number of filters, CNN will not converge to a satisfactory error rate, reflecting an underfitting scenario.

If the training set size is small, CNN will not converge; at this point, if we increase the training set size, it is likely to see it converge.

If the size of the filter is smaller than the average size that will work for a CNN, even when the training set size is small, we can get an overfitting result – because the CNN model is trained on the training set for too many times, it starts to catch noise in the training set. When an overfitted model is set to classify handwritten digits, it can achieve high accuracy on the training set, but only low accuracy on the testing set. To solve this problem, we need to increase the size of the filter, making it not too big or too small. A big filter will make CNN have too many parameters, then the error rate will not reach the plateau, because larger filters generally need more training data. When the filter size is not too big or too small, larger filters generally have better results, but the effect of the larger filter decreases with the increase of the filter size.

The number of convolutional layer also influences the model's capability. Its effect is similar to the effect of different sizes of filters. In an appropriate range, more convolutional layers will have better classification/prediction capability, but too many or too few convolutional layers will diminish CNN's capability.

In general, we can view the classification capability as a function of the filter's size, quantity, and convolutional layers, then the capability function will be of 3 variables and is concave. There is the best combination of the filter size, filter quantity, and the number of convolutional layers that maximizes the CNN performance.

There is a sweet point at training set size. If having too much, it causes overfitting, then the test error increases drastically, but before the overfitting, increasing training sets will stably decrease the test error.

4. Application of CNN on Traffic Sign Recognition

With the increase of GPU's performance, training a deep neural network and recognizing a traffic sign are possible. Traffic sign recognition is an important part of automatic driving.

Current traffic sign recognition technology includes LeNet-5, VGG-13, MTCNN, RCNN, Fast RCNN, and Faster RCNN [11]. LetNet-5 is the first one to be invented. It has 3 convolutional layers, 2 max-pooling layers, a fully connected layer, and an output layer. The training of LeNet-5 is similar to other CNN.

When applying LeNet-5 to the task of traffic sign recognition, we need to prepare the images for the input. Because traffic signs are the photos of real life, due to different distances, lighting effect, and weather's effect while taking the picture, there will be influences on the pictures and the pictures cannot look the same. The pictures are not in high resolution, and the size and scale are not the same. With preprocessing, the final pictures are suitable for CNN training. The picture's preprocessing is to convert colored image to grayscale image, wipe out noise, and cut out the traffic signs with the sign positioned in the middle and the sign is the only object in the image, lastly, enhance the features such as edges and resize the pictures to same image size, for example, 512x512. On the other hand, if the preprocessing caused the images to be too uniform, the training is easy fall into overfitting. To avoid overfitting, we can introduce a little noise, maybe Gaussian noise, to the image, and use tricks like *early stopping.*

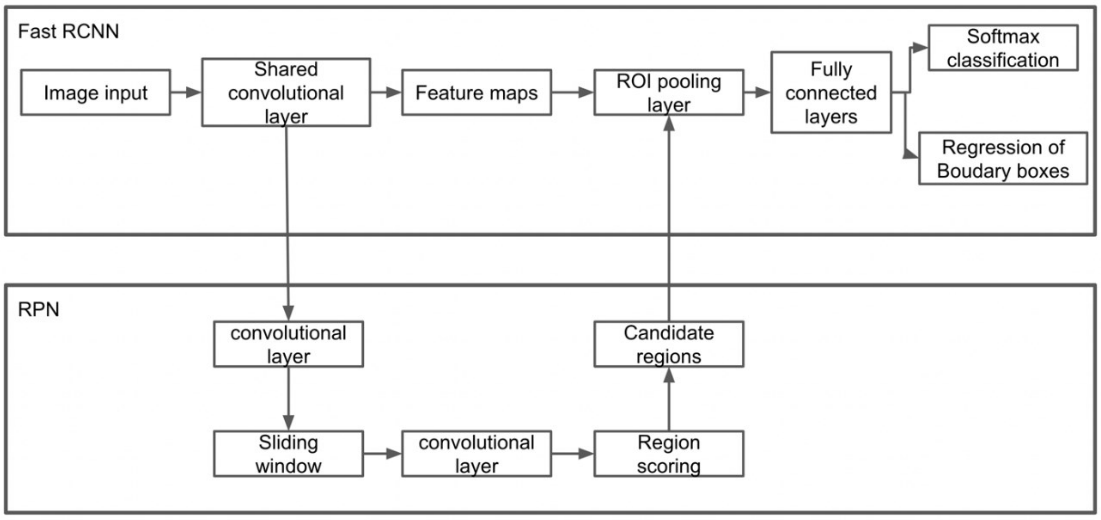

Figure 5: Fast RCNN Network Structure [11]

MTCNN includes P-Net, R-Net, and O-Net. The typical image sizes are 12x12, 24x24, and 48x48. P-Net and R-Net are relatively shallow that allow the classification process to be faster. P-Net, compared to R-Net, doesn't have a fully connected layer, it only has a convolutional layer. The image input to P-Net will be delivered to an ROI Pooling layer first, which normalizes the input images, so that MTCNN can process images of any size. The output of P-Net is some detected bounding boxes. The boxes are input into R-Net to be selected with regression. The selected boxes are then input to the fully connected layers of R-Net for further classification of the boxes. The O-Net is deeper than P-Net and R-Net, so it has stronger recognition power. The other aspects are similar to P-Net and R-Net. The final recognition results need to match with the output bounding boxes.

The difference of Fast RCNN from the category of LeNet-5 and VGG-13 (similar to LeNet-5, but more complex) that are capable of classification is that they are capable of *detecting* traffic signs. Fast RCNN is suitable for the traffic sign detection task under noisy background and unstable images data set, e.g., real-time data set.

RCNN's selective search algorithm is a process of similarity estimate, division, and composition. This approach, compared to the process of searching pixels by pixels, is quicker to find the positions of traffic signs. In the end, it generates the regions of the image that contains only the traffic signs. Comparatively, Fast RCNN's method is instead of generation of candidate region but reaching for traffic sign's feature right on the feature map. In this way, it is faster for fewer pixels to process.

RCNN and Fast RCNN's method of candidate region are both selective search algorithms, but this algorithm has problems of big computation, taking a long time, and excessive candidate regions. However, the RPN uses the RCNN's CNN to directly generate candidate regions, because RPN incorporates a selective search algorithm into the deep neural network, and the convolution computations are shared, so the overall computation is more efficient. Additionally, because RPN replaced selective search algorithm, that allows the computation of generating candidate boundary box to be moved to GPU, which accelerates the training. This type of network that includes RPN is also called Faster RCNN.

5. Conclusion

This article mainly describes the development background of convolutional neural network, and describes the CNN's basics, theories, composition, principles, and effects in every component. Lastly, it introduces CNN's application in handwritten digit recognition and traffic sign recognition and detection.

6. References

[1] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

[2] Khalid, S., Khalil, T., & Nasreen, S. (2014, August). A survey of feature selection and feature extraction techniques in machine learning. In 2014 science and information conference (pp. 372-378). IEEE.

[3] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 53(8), 5455-5516.

[4] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11), e00938.

[5] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). IEEE.

[6] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[7] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6), 141-142.

[8] Shamsuddin, M. R., Abdul-Rahman, S., & Mohamed, A. (2018, August). Exploratory analysis of MNIST handwritten digit for machine learning modelling. In International Conference on Soft Computing in Data Science (pp. 134-145). Springer, Singapore.

[9] Shustanov, A., & Yakimov, P. (2017). CNN design for real-time traffic sign recognition. Procedia engineering, 201, 718-725.

[10] Luo, H., Yang, Y., Tong, B., Wu, F., & Fan, B. (2017). Traffic sign recognition using a multi-task convolutional neural network. IEEE Transactions on Intelligent Transportation Systems, 19(4), 1100-1111.

[11] Zhu, S. (2019). Research on Traffic Sign Recognition Based on Deep Learning, Master’s thesis (pp. 46-57), Anhui University of Science and Technology.