A Survey on Object Classification using Convolutional Neural Networks

Roy Khristopher O. Bayot

Universidade de Évora, Department of Informatics,
Rua Romão Ramalho n°59, 7000-671 Évora, Portugal

Abstract. Object recognition has been one of the main tasks in computer vision. While feature detection and classification have been generally useful, an inquiry has been made to learning features suited to the task. One such method is the use of convolutional neural networks. This uses an architecture that combines elements of convolution, subsampling, and backpropagation. This paper gives an overview on the development, the use, and variations in using convolutional neural networks as an algorithm for object recognition tasks.

Keywords: object recognition, convolutional neural networks, deep learning

1 Introduction

Recent object recognition is usually a two step process. First features are extracted from images and then a classifier is trained with those features. And most of object recognition varies mainly on the type of features and classifiers used. The most common of these features are histogram of oriented gradients, scale invariant feature transform and its variants, and bag word features while classifiers can vary from naive-bayes, to support vector machines, to logistic functions, as well as many others.

Histogram of Oriented Gradients was used by Dalal and Triggs in [9]. The idea is that the shape and appearance of an object within an image can be described by a histogram of intensity gradients. So this descriptor was made by dividing the image into smaller regions. And for each region, a histogram of the edge orientations is computed. The combination of these histograms form the descriptor for the image. This method has been used for human detection in the same study in [9].

SIFT has been one of the most widely used feature descriptor in object recognition. Lowe [19] first proposed this method of turning images into a collection of local feature vectors which are invariant to translation, scaling, and rotation. It is also partially invariant to illumination changes. These features are initially detected using a Harris-Laplace corner detector in scale space. Once stable corners are detected, the scale space at which they were detected are stored. Then for each of these keypoints, the dominant gradient is also computed over a neighborhood of 16 x 16. This is also stored and after which, the descriptor for this

keypoint is computed. This keypoint descriptor is described by a 128 dimensional vector. This vector is obtained by selecting a 16 x 16 neighborhood around the keypoint. This is then partitioned into a 4 x 4 region with each region containing 4 x 4 pixels. For each region, a histogram with 8 bins is computed from gradient magnitudes in that region. Thus, the 128-dimensional vector is formed. SIFT has also many variants. The first of which is SURF or Speed Up Robust Features [2]. The key difference is that this uses integral images to speed up convolutions. Other variants include RGB-SIFT, HSV-SIFT, and Opponent-SIFT. These mainly use SIFT on different color spaces since SIFT was primarily made for grayscale images. RGB-SIFT operates on the RGB channels, HSV-SIFT on HSV channels, and C-SIFT on opponent channels. One final variant is called dense SIFT (or D-SIFT) wher descriptors are computed for every pixel. And some of these variants could be taken from the implementations of VLFeat [27].

SIFT produces multiple keypoints. And one can imagine multiple images to train a classifier. This could result in a very long list of keypoint features. One method to compress this list of features is to use bag of words. Essentially, the keypoint features undergoes a processing step before being used to train a classifier. All the keypoint features are collected and then a clustering algorithm such as k-means is used. The number of clusters is specified and this indicated the bag of words feature vector. Once the clustering is complete, a histogram is computed for each image. This indicates how many features in the image belong to a certain cluster. The classification step then operates on the histogram produced after the clustering step. An example of using such a method is that of Issolah et. al. [15] where plant images are classified using SVM. The features were taken from SIFT and k-means algorithm with 100 clusters was used to generate the bag of words features that was used in the classifier.

SIFT, SIFT variants, bag of words, and histogram of oriented gradients have all been quite useful for object classification however it is needed to be specified and selected for the task. One can imagine that ordinary SIFT would be a weak choice for the task if the image to be classified involves circles since SIFT finds keypoints based on corners. One of the recent methods proposed to aid in this problem is to learn the features detectors that are need for the task. And this paper will discuss mainly how this is done through convolutional neural networks.

2 Convolutional Neural Networks

2.1 Feed Forward Artificial Neural Networks

A convolutional neural network is an architectural extension of the feed forward artificial neural networks with multiple layers. Artificial neural network typicall has an input layer, a hidden layer, and an output layer. The number of hidden layers as well as the number of units or nodes in each hidden layer is parameterizable. Each of the nodes in the hidden layer and output layer, behaves like a neuron. The output of this single neuron is given by the equation

$$a = f(\boldsymbol{\theta}\mathbf{x}) \tag{1}$$

where the function $f(x)$ is given by a nonlinearity like that in equations 6, 7, 8 and $\mathbf{x}$ is the input row vector augmented with 1 for the bias, and $\boldsymbol{\theta}$ are the weights as a column vector.

To train a neural network, the backpropagation algorithm is used and the delta rule is used to update the weights. Essentially, given say a classification task, the input vector is propagated through the hidden layers to the output layer, using equation 1 for each neuron in each layer. When it hits the output layer, the predicted class is compared to the actual class and the error is propagated to the other layers. This is given by the equations below.

Output layer error:

$$\delta_j^{(l)} = a_j^{(l)} - y_j \tag{2}$$

Hidden layer error signal:

$$\delta^{(i)} = (\boldsymbol{\theta}^{(i)})^T . * \Delta a^{(i)} \tag{3}$$

The $\boldsymbol{\theta}^{(i)}$ gives the weights to layer i in the neural network. The $\Delta a^{(i)}$ are the gradients for the activation funciton in layer i. And $\delta^{(i)}$ are the error signals backpropagated to update the activation in layer i.

For a neural network with one hidden layer, updating the weights $\boldsymbol{\theta}^{(i)}$ is as follows:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i)} - \alpha(\Delta\boldsymbol{\theta}^{(i)}) \tag{4}$$

$$\Delta\boldsymbol{\theta}^{(i)} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}^{(i+1)} a_n^{(i)} \tag{5}$$

The variable n gives the number of training examples. While $a_n^{(i)}$ gives the activation in layer i.

One can imagine using an image and reshaping the 2 dimensional matrix into a vector which will then be used in a feed forward artificial neural network with many hidden layers. However, this has not proven effective because of the number of weights to be kept updated and also because of diminishing gradient as the number of hidden layers increase. The convolutional neural networks we know have been further inspired by the study of the visual cortex as shown in the next subsection. And this will later be developed into the basis as how we know it today.

2.2 Visual Cortex Inspiration

Hubel and Wiesel conducted a study on the receptive fields of the cat [13] and that of the monkey [14]. They experimented on cats and monkeys by stimulating their retinas with spots or patterns of light. Then they observe the responses on each. They found out the receptive fields defined by different cells. The first of these are simple cells. These have receptive fields that give a distinct on and off areas separated by parallel straight lines when a spot of light is given as a stimulus. These cells are position dependent. Then, there are complex cells which respond when a correctly oriented line on shined on the field. And these

also have different responses to slits, edges, and dark bars. Hypercomplex cells on the other hand have a different response such that when the line is extended beyond the activation area of the field, the response drops off. There have also been cells lacking in orientation specificity, cells with specific color response, and complex and hypercomplex cells with color selectivity. And finally, cells with concentric fields and dual opponent system. These cells seem to be organized around a center where the center is activated by long wavelengths while the periphery is the reverse.

Another interesting finding are the architecture features of the visual cortex as composed by these cells. First, the layering on the visual cortex allows the aggregation of different cell types. This is from simple cells, to complex cells, to lower order hypercomplex cells, to higher order complex cells. Second, there is also an aggregation due to the receptive field orientation. As the the depth of the cortex increases, the orientation decreases. And finally, clustering of cells is independent depending on eye dominance.

2.3 Neocognitron

This study by Hubel and Wiesel became the basis for the Fukushima's Neocognitron [10]. It consist of a cascade connection of modular structures. These structures are made of two layers of cells in a casacade - the "S-cells" and the "C-cells". The S-cells show similarities to simple cells or lower order hypercomplex cells. The C-cells on ther hand are similar to complex cells or higher order hypercomplex cells. Each of these layers have multiple cell planes. The synapse for each receptive field in the simple cells are modifiable and changes in training. Also, all the cells in each cell plane have input synapse of the same distribution.

Training this network by Fukushima did not employ backpropagation algorithm. Instead it used an unsupervised competitive learning method which uses reinforcement of cell planes that were selected among representative planes. This reinforcement method was the same for Fukushima's previous work.

2.4 LeNet5

The work of LeCun et. al. which has been mainly inspired by the work of Fukushima, has been the basis for the types of convolutional neural networks we see today. The is known as LeNet5. The main difference between LeCun's work and Fukushima is the use of backpropagation. LeCun's work also revolves around three main ideas to make sure that whatever features could be captured would be invariant to shift, scale, and distortion. These three ideas would be local receptive fields, shared weights, and spatial or temporal subsampling [18]. The three ideas would be realized by implementing an architecture wherein there is an alternate between a convolutional layer and a subsampling layer as shown in figure 1.

The input image is convolved with three trainable filters and biases to produce three different feature maps at layer C1. Then maps are subsampled by taking a group of 4 pixels, where they are added, weighted, and combined with a

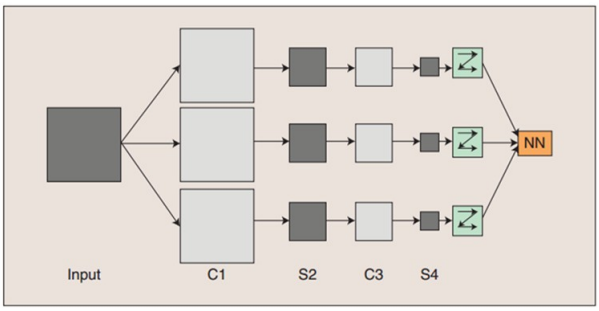

Fig. 1. Conceptual example of convolutional neural network. Image taken from [1].

bias, and passed through a non-linearity to make 3 features maps at S2. This is illustrated in figure 2. Then another convolution is performed to make C3, and doing the same subsampling to make S4. Soon, the resulting feature maps are turned into a single vector and then becomes an input to a conventional neural network.

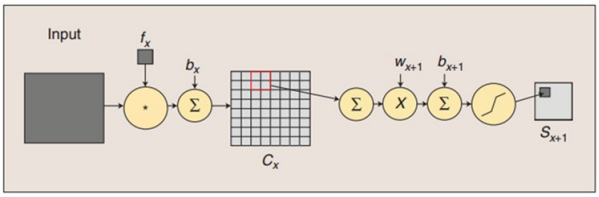

Fig. 2. More details regarding convolutional maps and subsampling. Image taken from [1].

The CNN architecture is parameterized by the number of layers, the number of maps per layer, the size of the kernel for the convolutional layer, the size of the subsampling window, and also the way things are connected - if there is a full connection between layers or not or if there was a pattern adhered. In LeCun's work [18], the task was for document recognition and the architecture was shown in figure 3.

The first layer is the input layer, or the images. There could be 1 channel or 3 channels to accommodate color. For this case we have 1 channel. The second layer is composed of 6 feature maps. To get from the input layer towards one of the feature maps, the input layer is convolved with kernel, then added with a bias, and then passed through a non-linear function such as a sigmoid function as shown in figure 2. The kernels are randomly initialized and later updated after each pass of the backpropagation algorithm. Each kernel is different from one map to the next but the same kernel is used within one feature map.

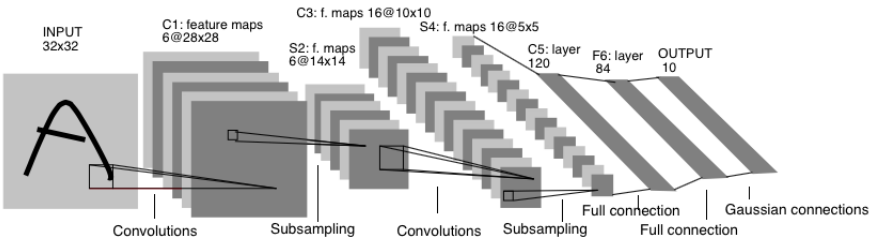

Fig. 3. Architecture of LeNet5

This method of using convolution on a kernel provides the implementation for receptive fields and shared weights. The local receptive fields are mimicked by using the kernel and the shared weights shows that not all the nodes are fully connected but that there is only a replication of the weights.

Once this layer is done, the idea is that there will be points of interest or features present in the feature maps. However, once the feature is detected, the exact location becomes less important. This is why layer 3 is composed of a subsampling layer. This means that for this implementation, an averaging around 4 pixels is made and taken. The average is then multiplied by a trainable coefficient, added with a bias and then again passed through a sigmoid function.

This alternate between convolution and subsampling is repeated for the next two layers except that the maps are increased and that there have been new combinations. Finally, towards the end, there is the fully connected network.

Obtaining the weights for the kernels on and the biases are done using back-propagation and updated after every pass.

2.5 Recent additions to the architecture

LeNet5 uses both convolution and subsampling by averaging. However, there have been new additions to what could be done to the architecture. The first one is in pooling.

Since weights are replicated in a convolutional layer, a feature can be seen in multiple parts of the image. Or the adjacent results will come close to that of the feature desired. The pooling layer essentially takes the results of the adjacent feature detectors and combines them. This is essentially a form of dimensionality reduction and preserves only the useful information [5] [26]. The typical pooling layer is average-pooling but there is another pooling function called max-pooling, which returns only the maximum value from the input. A feature of average pooling is that it lowers the effects of a higher values even and this becomes a problem if the important information is in the higher values. The feature of max-pooling is that it ignores other values in the pooled layer.

Other variations regarding the pooling layer is the use of spacing such that they would take distinct separate inputs or could be overlapping inputs. This

changes the courseness of the coding [12]. Finally, Zeiler et. al. proposed a stochastic max-pooling which randomly picks the activation value [28].

The second set of possibilities are dropout and dropconnect. Dropout by Hinton et. al. [12] works by randomly disregarding half of the neurons for each training case. This enables the working neurons to use the random combination and not rely on the architecture. This helps in preventing overfitting. Dropconnect on the other hand acts similarly to dropout but instead of omitting the half of the neurons, it sets the weights to zero.

The third set of possibilities are the choice of activation functions. Most of the common choice of activation functions are the sigmoid function and the hyperbolic tangent. However, rectified linear units have been recently used because they are bounded by their minimum value and then causes the networks to suffer less from diminished gradient flow [11]. This then helps to train deeper networks. The equations for sigmoid, hyperbolic tangent, and rectified linear are given by equations 6, 7, 8 respectively.

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{7}$$

$$rect(x) = max(0, x) \tag{8}$$

2.6 Variants

Convolutional neural networks have been applied to many things and there have been variants to the general architecture. The first variant is the choice of number of maps.

Another variant is to hardwire the values of some of the filters. This was seen in the work of Kwolek when tasked with face detection [17]. The network consisted of 6 layers, normally with convolutional and subsampling layer. They key difference is that the convolutional kernel for the first layer was not learned. Instead it was hardwired to be that of a Gabor filter. The reasoning for this is that the network would have to learn the basic edge detectors and using a Gabor filter would already give that information. This has also been similar to the work by Mutch and Lowe with object classification on Caltech 101 images [20]. And this has also been similar to the work of Serre et. al. in [22]. The use of the Gabor filters was also to mimic a primate's visual cortex.

One of the possible problems with the use of convoluntional neural networks is that it may require a large amount of examples per class. And there is a possibility that given a training set, the number of images for a class could not be enough. And this results to another variant when dealing with convolutional neural networks, which is to pretrain the network with other images. This is done in an unsupervised manner such as in the work of Ranzato et. al. [21].

The idea is to combine elements of convolutional neural networks with autoencoders. Autoencoders are a feedforward multilayer perceptrons with input,

hidden, and output layers. The key difference is that the number of nodes for the input and output layers are the same and the goal of the the autoencoder is to reconstruct the input. Learning the weights on pretraining in an unsupervised manner is such a case when the input image is fed into the convolutional neural network which is adapted as an autoencoder with the output to be the same as an input.

This opens up problems with the depth of convolutional neural network as mentioned in [3]. It is essentially difficult training really deep neural networks because of diminishing gradients flows. Basically for really deep neural networks, the layers closer to the input have the diffused effect of the error that was back-propagated from the output. Fortunately, one way work around this is to train a deep neural network by doing a greedy layer-wise training [3] [4].

Finally, one of the problems with using convolutional neural networks is that the as the number of layers increases, it also takes sufficient amount of time to train the network. And usually this is worked through with the network architecture, size of the filters, and the choice for connectivity. However, another approach is to use graphics processing units to speedup the calculations such as in [24] [8] [25]. But one of the interesting work was done by Ciresan et. al. in [7] where they trained multiple deep neural networks with GPUs to classify images. Each of those deep neural networks have a hundred maps per layer unlike that of LeCun in [18]. And finally, each of those networks are combined into a multi-column deep neural network by averaging the predictions.

3 Applications

Convolutional Neural Networks have been used to classify a variety of things. The following sections lists some of the applications and the datasets that were used.

MNIST digit dataset has been one of the main datasets used to benchmark the performance of convolutional neural networks in object recognition. In the work of LeCun in [18] used 7 layers (excluding the input) to perform classification. In [23], MNIST was also used to demonstrate best practices for convolutional neural networks. And in [7], multi-column deep neural networks on GPUs were used to classify the digits from the same dataset.

The work of Ciresan et. al. in [6] was classification of german traffic signs. The main contribution of this work aside from classifying traffic signs is that it uses a fully parameterizable GPU implementation of the convolutional neural network. This work also has a convolutional layer that could be parameterized to skip convolution between a set number of units. This work resulted a recognition rate of 99.15% which is better than the human recognition rate of 98.98%.

The work of Kang et. al. in [16] uses convolutional neural network to classify documents. The datasets used were tobacco litigation dataset and NIST tax-form. The architecture is similar to that of LeCun in [18] but the input images were resized from 2000x2000 pixels to 150x150 pixels.

4 Conclusion

This paper explored the use of convolutional neural networks for object classification where feature detectors were learned. While it seems that this approach proves so many advantages, more work still needs to be done. For one, there will always be a question with the choice of architecture and the speed at which it is being trained. Another interesting direction is from object recognition towards image understanding.

References

1. Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.
2. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.
3. Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
4. Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.
5. Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118, 2010.
6. Dan Ciresan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE, 2011.
7. Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.
8. Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.
9. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
10. Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.
11. Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323, 2011.
12. Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
13. David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

14. David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.
15. Mohamed Issolah, Diane Lingrand, and Frederic Precioso. Sift, bow architecture and one-against-all support vector machines. In *Working notes of CLEF 2013 conference*, 2013.
16. Le Kang, Jayant Kumar, Peng Ye, Yi Li, and David Doermann. Convolutional neural networks for document image classification. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 3168–3172. IEEE, 2014.
17. Bogdan Kwolek. Face detection using convolutional neural networks and gabor filters. In *Artificial Neural Networks: Biological Inspirations–ICANN 2005*, pages 551–556. Springer, 2005.
18. Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
19. David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
20. Jim Mutch and David G Lowe. Multiclass object recognition with sparse, localized features. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 11–18. IEEE, 2006.
21. M Ranzato, Fu Jie Huang, Y-L Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
22. Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994–1000. IEEE, 2005.
23. Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, pages 958–958. IEEE Computer Society, 2003.
24. Daniel Strigl, Klaus Kofler, and Stefan Podlipnig. Performance and scalability of gpu-based convolutional neural networks. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 317–324. IEEE, 2010.
25. Rafael Uetz and Sven Behnke. Large-scale object recognition with cuda-accelerated hierarchical neural networks. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 536–541. IEEE, 2009.
26. Joost van Doorn. Analysis of deep convolutional neural network architectures. 2014.
27. Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472. ACM, 2010.
28. Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.