Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Deep Learning: Neural Networks for Object Detection and Tracking Tasks

Seminar Paper

Brain Modelling

Daniel Speck

Matr.Nr. 632 13 17

2speck@informatik.uni-hamburg.de

29.05.2015

Abstract

Deep learning and the interrelated deep neural networks are one of the most successful learning strategies at the moment as the computing power for creating such structures rised in the past years via GPU computing. Image and video classifcation, object detection and tracking tasks can be fulfilled with these architectures which perform far better than more classical ones. One focus will be convolutional neural networks and the latest research on (mostly supervised) deep learning architectures.

Contents

1 Introduction **2**

2 Background information: Artificial neural networks **2**

3 Deep neural networks **3**

4 Convolutional neural networks and image processing **4**

5 Research and field of application **6**
5.1 Image classification 6
5.2 Video classification 7
5.3 Object tracking 9

6 Conclusion **9**

Bibliography **10**

1 Introduction

Big data is one of the most present keywords in the last couple of years. With more and more devices being able to capture high-resolution images and videos the data and information available on the internet increases rapidly, rather exponentially but all this data is useless if this flood of information can not be searched, categorized and made easy accessible for humans. As a consequence of this situation classic, mostly static algorithms are "out-of-date" and new, intelligent systems have to be developed. Biological entities are, up to now, by far better when it comes to recognize, classify, sort etc. images or videos compared to computers. This circumstance led to researching intelligent systems that are able to adapt to their input, search for patterns and recognize, track and classify them.
Inspired by biological systems, deep learning and especially convolutional neural networks have become popular because their results in classification and tracking tasks are state-of-the-art.
In this paper, a basic, background information view about artificial neural networks will be given for understanding the underlying concepts which are important to solve complex problems with machine learning architectures.
In addition to that a comprehension of deep neural networks and deep learning will be made to point out the abilities but also the limits of "classic" neural networks. Also some information about the role of GPU-computing in such disciplines will be provided.
For solving complex problems in classification, recognition, tracking etc. domains convolutional neural networks have shown good performances so they will be focused later on. Especially the latest research on these topics and fields of application will be viewed in detail.
An important contest for classification tasks is measuring the performance of such architectures with the ImageNet LSVRC databases and for videos the YouTube 1M database by Google. Both topics will be introduced. One very interesting thing, for example, is the fact that pre-trained networks on videos showing sports did better on non-sports related video categorization than "fresh" networks.

2 Background information: Artificial neural networks

Artificial neural networks are intended to approximate certain functions for machine learning purposes. Fix/static algorithms can be calculated fast by modern computers but fail most times at disciplines requiring "intelligent" behavior. A classic example is recognizing handwritten digits because the shape, color and contrast highly vary in dependence of the used pencil and, of course, of the writing style. As a result classic, static algorithms would fail to identify those varying patterns and therefore fail to appropriately classify handwritten digits.
Such tasks can be fulfilled by artificial neural networks. A common architecture is the multilayer-perceptron as shown in figure 1.

This artificial neural network, for example, could be used to classify handwritten digits. The images information of input size 5x5, which results in 25 pixels, would be used by the 25 neurons in the input layer. The hidden layers would try to extract the features which are characteristically for each digit and the output layer with 10 neurons models the digits 0 to 9 for representing the calculated digit.
However, such architectures are insufficient for more complex tasks because they need a vast amount of resources, are not optimized and do not make use of any prior-knowledge of the problem. Non-deep structures where every layer is fully-connected turned out to be expensive in terms of computational resources and not optimal considering the results in disciplines of image and video classification, object recognition and tracking.

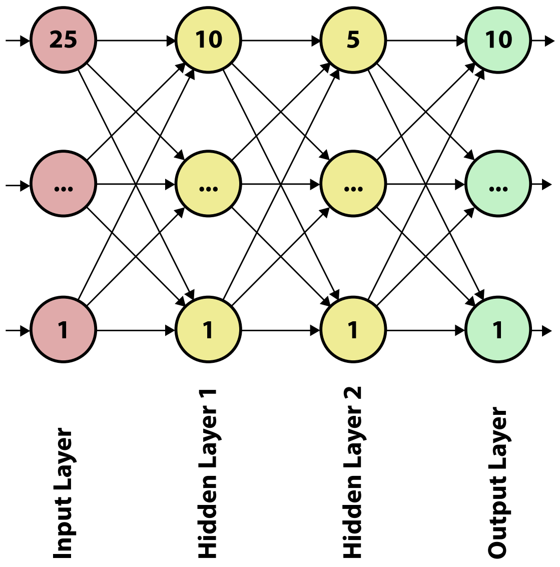

Figure 1: An example of a classic, fully-connected multilayer-perceptron. The input layer holds 20 neurons, hidden layer 1 10 neurons, hidden layer 2 5 neurons and the output layer 2 neurons. All arrows but the short ones represent randomly initialized weights.

3 Deep neural networks

Classic, small neural networks can not succeed on complex tasks such as object detection in high-resolution, real world images. The variety of features, blurry backgrounds, one or several objects in an image and other factors render detection, tracking and classification tasks challenging. The ability to detect, track and distinguish real world objects demands complex structures with a much larger capability of solving complex problems.
As computing power rises and GPU-computing became popular neural networks are no longer limited to a few layers containing just some neurons. Modern solutions (2014) can carry tens of layers with thousands of neurons and millions of

connections [4]. Complex structures like this enable computing complex problems which arise with tasks like image classification in real world images.
Deep neural networks with GPU-optimized code can be considerably faster than CPU-optimized code and hence allow larger, more accurate architectures [1].

4 Convolutional neural networks and image processing

For deep learning purposes (classic / fully-connected) multilayer perceptrons consume a sizable amount of resources for proper training when they are designed to solve complex tasks because the amount of neurons and especially weights increases rapidly with the network's size.
For example, a MLP with three layers, an input layer with 100 neurons, a hidden layer with 25 neurons and an output layer with 10 neurons for classifying images with a size of 10x10 pixels into 10 different classes would have $100 * 25 + 25 * 10 = 2,750$ weights/connections. Training this net would already result in a big time and space complexity. Moreover, as features in images capturing real world scenes are distributed in certain patterns (they cover spatially local correlation, such as shapes), there is no need to have every pixels information being processed by one neuron. Actually in most cases results would be even better, if the pixels information is pre-processed, for instance by edge detection filters but a fully connected layer of neurons is not an optimal solution for this task.
Convolutional neural networks (CNNs) are inspired by biology, instead of connecting every pixels information directly with a neuron to process its information it filters the information in the first layers [5]. This procedure is similar to the on processes happening when an biological eye receives stimuli.
The receptive field [1] has a vast amount of photoreceptor cells [2] gathering information and converging the received information on to distinctly less retinal ganglion cells [3]. This process maps several features and reduces the input dimensionality as well as distinguishes the information to separate "channels" which are then transfered to the corresponding neurons to process features such as color, motion, shapes and so on separately [6]. The idea of CNNs is based on this biological processes, the information of an input image is convolved by several filters which try to extract interesting features in the first layer and in following layers this information is pooled and subsampled [5].
Convolution itself is the applying of a function repeatedly of the output of another function and in the context of CNNs it is applying different "filters" over an image to extract the already mentioned features. A convolution layer extracts the pixel information out of an image with kernels [4].

[1]http://en.wikipedia.org/wiki/Receptive_field
[2]http://en.wikipedia.org/wiki/Photoreceptor_cell
[3]http://en.wikipedia.org/wiki/Retinal_ganglion_cell
[4]http://en.wikipedia.org/wiki/Kernel_(image_processing)

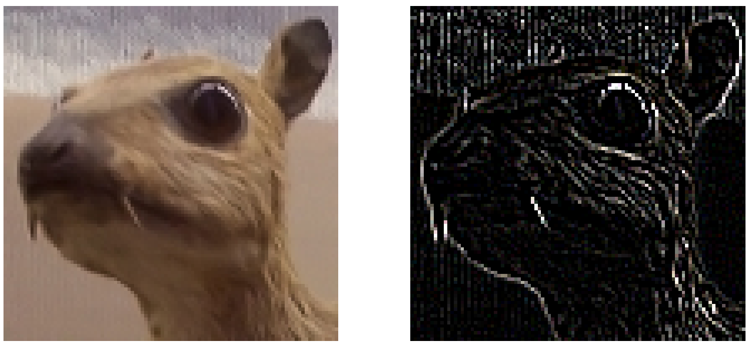

Figure 2: Left side: original image, right side: edge-detection kernel processed image. Original image by Michael Plotke, 28th of January, 2013. Open creative commons license. `http://upload.wikimedia.org/wikipedia/commons/5/50/Vd-Orig.png` and `http://upload.wikimedia.org/wikipedia/commons/6/6d/Vd-Edge3.png`

Example for a convolution layer: In figure 2 you can see an image of an animal on the left side (original image) and a kernel processed one on the right side. The used kernel matrix for filtering the left image is shown in equation (1). So basically each pixels information in the right, processed image is the result of applying the kernel matrix (1) on the same pixel (and the neighboring pixels) in the left image. With a wide variety of different kernels several different features can be extracted from an image.

$$K_M = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{1}$$

Typical CNNs use tens to hundreds of different kernel filters gathering as much features from an image as possible. The resulting pixel processed by a kernel filter is calculated via the formula:

$$I_{out}(x,y) = \sum_{a=1}^{3}\sum_{b=1}^{3} I_{in}(x+a-c_x, y+b-c_y) * K_M(a,b) \tag{2}$$

Where c_x is the coordinate of the x-center and c_y the coordinate of the y-center of the input image. For some example input image I_{in} with a dimension of 3x3 pixels the center pixel (coordinates $x = 2, y = 2$) of the kernel processed output image I_{out} using the kernel filter K_M of equation (1) would be calculated like this:

$$I_{in} = \begin{bmatrix} 46 & 42 & 50 \\ 44 & 65 & 56 \\ 41 & 52 & 58 \end{bmatrix}$$

$$I_{out}(2,2) = 46*(-1) + 42*(-1) + 50*(-1) + 44*(-1) + 55*8 + 56*(-1) \\ + 41*(-1) + 52*(-1) + 58*(-1) = 131$$

After processing this filter to the whole image edges would be highlighted and the rest nearly black, like in figure 2 so that shape features are extracted out of the original image.

Another idea of CNNs is subsampling layers following convolutional layers. Subsampling reduces the overall size of information and therefore not only saves resources for the later classification tasks but also strengthens the detected features in an image. Often max-pooling is used as a subsampling strategy in CNNs [8, 5] which determines the most distinctive pixel in a given area. A max-pooling algorithm splits the input in grids and selects the maximum value out of each grid, thus non-maximal values are deleted so that only those information continues to exist which best represents the current feature. Before applying max-pooling the input represents the presence of a feature in one or only some pixels. After this kind of dimensionality reduction the assertion is enlarged to whole area, corresponding to several pixels of the original image.
Additionally this technique provides robustness to the position of a feature in an image, as the position becomes less important.

The last layers of a CNN regularly consist of fully-connected layers. In comparison to feature extraction, strengthening and dimensionality reduction those layers are supposed to take all gathered features / information and make predictions. If the supplied input contains big shapes of ears, legs etc., maybe filtered trunks, grey-ish color maps and so on the fully-connected layers have to collect and mix all those gathered features / information and to recognize it for finally classifying it as an image of elephants.

5 Research and field of application

5.1 Image classification

Alex Krizhevsky et al. have shown in "ImageNet Classification with Deep Convolutional Neural Networks" [5] the good performance of CNNs in image classification tasks. Their architecture achieved top-1 and top-5 error rates of 37.5% and 17.0% on the ImageNet LSVRC-2010 contest which was the best result in 2012.
The CNN consists of about $650,000$ neurons with 60 million parameters and nine layers total.
The only preproduction made were resizing the images to a resolution of 256x256 because the images in ImageNet vary in resolution. Out of those images 224x224x3 dimensional patches are extracted. 224x224 in horizontal/vertical dimension and 3 different colors (RBG images). The first layer is a convolution layer which filters the input of 224x224x3 to a kernel map of 96 kernels with the size of 11x11x3 each. This slightly reduces the dimensionality of the input image and extracts 96 different features out of the input.
The second layer is a max-pooling layer which filters the information via 256 5x5x48

kernels. Therefore the dimensionality is further reduced and already extracted features strengthened.
Afterwards convolutional and max-pooling layers follow the the fifth layer until three fully-connected layers with 4096 neurons each try to collect and mix the information and then classify the images.

One key problem with deep learning structures and especially CNNs is overfitting. Because of the size of those networks a huge amount of training data is needed. An interesting method used by Krizhevsky is to design first layer (224x224x3 for Krizhevsky's architecture) slightly smaller than input data (256x256x3) and then not just apply kernel filters on those input data but first extract random 224x224x3 patches out of the original data. Additionally they the horizontal reflections of those patches and altered the intensities of the RGB channels.
This technique can boost the training data set by hundreds or even thousands of samples [5].

Another procedure to reduce overfitting and test errors is called "dropout" [3]. The idea is to set the output of hidden neurons to zero with a certain probability, mostly 0.5 [5]. This prevents neurons of co-adapting each other which means that neurons can no longer distort its output because they "familiarize" with some neurons of their parent layer and ignore others. With "deactiviting" random neurons in each learning step the presence of an other neuron is not guaranteed so neurons are more likely to learn and focus on robust features rather than on single attractive neurons.
This has shown especially effective when applied in fully-connected layers [5].

Normally the transfer function for a neurons output is modeled a hyperbolic tangent or sigmoid function but those functions can increase the training time [5]. Using Rectified Linear Units (ReLUs) [2] with a function like $f(x) = max(0, x)$ not only makes CNNs train a lot faster but also slightly reduces the training error, because low training error rates are reached faster (in less epochs) the whole network concentrates on robust features faster and this has an additional effect of convergence.
In Krizhevsky's architecture the 25% error rate on the CIFAR-10 training dataset is reached six times faster with ReLUs than with classic hyperbolic tangent transfer functions [5].

5.2 Video classification

A more challenging and likewise interesting task is the classification of videos because instead of images where the input remains static the input of videos changes over time.
CNNs already showed "state-of-the-art" results in image recognition and classification tasks so experiments with videos were just a matter of time. A huge problem was the changing input of videos over time in combination with the long periods of

training for CNNs. One solution figured out by a group of researchers of Stanford and Google was to improve the runtime performance. They split up the input data in two different streams, the "context" and the "fovea" stream and created a multiresolutional architecture [4].
The original input dimension was 178x178. The fovea stream keeps track of the centered 89x89 pixels of a frame while the context stream analyzes the whole frame but downsampled to a size of 89x89 total. Therefore the total input dimensionality is halfed [4]. An illustration of this architecture is shown in figure 3.
Four different solutions have been introduced by Karpathy et al. to create CNNs capable of classifying videos properly: Single Frame, Early Fusion, Late Fusion and Slow Fusion.
Single Frame classification is an adoption already used for classifying images: only one image is being classified at a time.
Early Fusion combines the pixel input information for a specific time window.
Late Fusion calculates two different single frames which are combined at the last fully-connected layers.
Slow Fusion is a mix of Early and Late Fusion: Several input images pixel information is combined two several streams and those streams are additionally combined every few layers.
According to Karpathy et al. [4] Slow Fusion had the best results with top-1 and top-5 error rates of 60.9% and 80.2%.

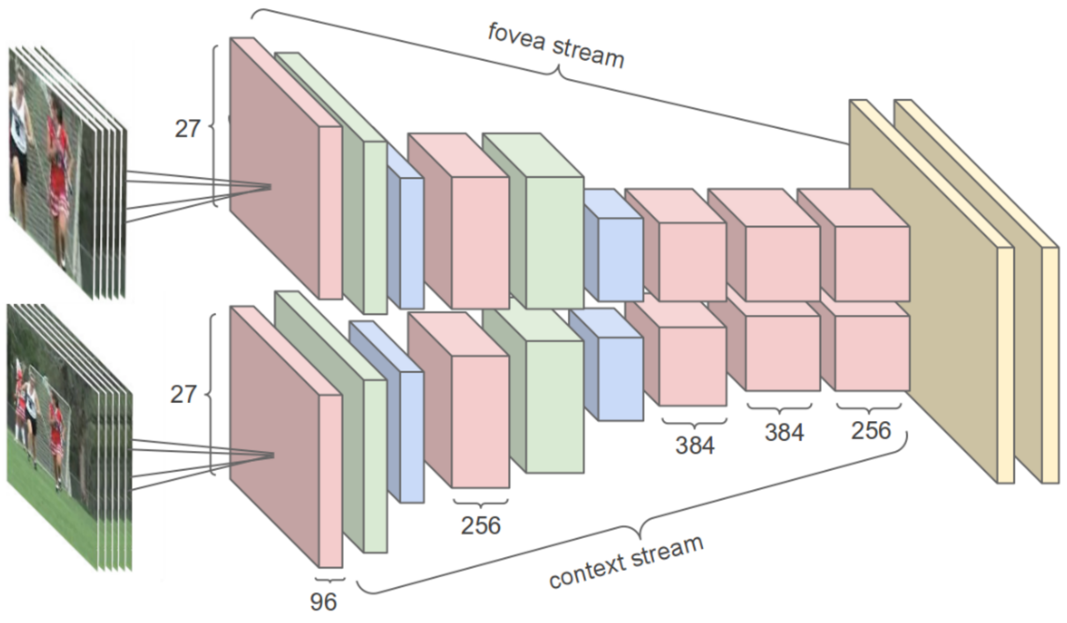

Figure 3: CNN architecture by Karpathy et al. The input information is processed in parallel by the two streams. Graphics retrieved from "Large-scale Video Classification with Convolutional Neural Networks" by Andrej Karpathy et al. [4]

5.3 Object tracking

Object tracking in videos is an even more complex scenario because some object in a video has to be recognized and followed by its position over time. Varying background, changing in the lighting, unexpected motions etc. render this a challenging task.
One supposed solution is the use of stacked denoising autoencoders to learn common features of a dataset and then have an additional deep neural network for tracking those learned patterns over time [9].
Also some kind of preprocessing with particle filters showed good results for tracking purposes [9].
(More to come - complex stuff has not be shortened and summarized enough until now)

6 Conclusion

In the beginning of 2012 the best top-5 error rate on ImageNet was 16.4% by Team "SuperVision" and in 2014 the best top-5 error rate was 6.67% by Team GoogLeNet [7]. This shows the rapid development of deep learning and CNNs but also that there is enough potential for even further improvement. At least for the next years the performance of humans in classifying images will be unmatched which indicates that todays models have to be improved.
In many papers [7, 8, 5, 9, 1] researchers say improving the computational resources also improves machine learning architectures because CNNs and other neural network architectures could deliver better results if they would be enlarged (which is not possible with current resources) but also new concepts, filters and architectures can improve machine learning.
On the one hand CNNs are one or maybe the most promising architecture at the moment [7, 5] with astounding results in classification tasks and larger CNNs, more training data, enhanced precision and perhaps some architectural / mathematical improvements are likely to boost their performance even further. On the other hand (although they are less resource demanding than pure, fully-connected neural networks for example) their resource consumption is relatively high and as shown in "Deep Hierarchies In The Primate Visual Cortex" [6] they are not equivalent nor equally effective as biological neural architectures. This displays that CNNs can only be a temporal model but not the ultimate solution and new architectures should be researched.

References

[1] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642 – 3649, June 2012.

[2] Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. In *Proc. 27th International Conference on Machine Learning*, 2010.

[3] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[4] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[6] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio Jose Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1847–1871, 2013.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[8] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013.

[9] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013.