Microscopy cell semantic segmentation with fully-convolutional neural networks

Alessandro Marzo
Francesco La Rosa

October 2016

Abstract

In this work we developed a deep learning cell recognition system using fully convolutional neural networks. Our approach is based on semantic segmentation to correctly classify every single pixel that belongs to a cell in an image. After a brief introduction on the biological basis of deep learning, we focus on the fundamental principles of common deep learning architectures, such as convolutional neural networks. Recent works show that the fully convolutional neural network architecture is particularly capable of semantic segmentation tasks. This particular architecture is characterized by the ability of accepting in input images of arbitrary sizes and we exploit this to improve the network computational efficiency by training on image patches. First we trained the networks on a dataset of computer generated images and then we took advantage of the transfer learning properties of neural networks and fine-tuned the model to a dataset of real microscopy images, further improving the accuracy of the network. Finally we compared the results of the two trained networks using various indicators of performance.

Contents

1 Introduction **2**

2 The biological basis of deep learning **4**
2.1 The perceptron 5
2.2 Neural network architectures 6
2.3 Convolutional neural networks 6

3 Convolutional neural networks **7**
3.1 Convolutional layer 8
3.2 Pooling layer 9
3.3 Fully-connected layer 9

4 Fully convolutional neural networks **9**

5 Semantic segmentation **10**

6 Transfer learning **10**

7 Semantic segmentation with fully convolutional neural networks **11**
7.1 Software and simulations 11
7.1.1 Caffe 12
7.1.2 SIMCEP 12
7.2 Implementation details 13
7.3 Results 14

8 Conclusions **17**

1 Introduction

Deep learning is a branch of machine learning based on various computational models that allow to learn representations of data. The models are made of multiple processing layers, composed of both linear and non-linear transformations, and attempt to discover intricate structures in large data set. In the last years various deep learning architectures, such as deep neural network and convolutional neural network (CNN), have brought about breakthroughs in speech and images recognition, drug discovery, genomics and many other fields.

Deep learning is based on the assumption that the raw data observed can be analysed separating its factors of variation, the separate sources of information. The aim of the algorithms is then to extract these useful features by studying different level of abstraction or composition of the data. This can be done changing the number of the layers and their size in the model architecture. Thus complex representations and concepts can be understood composing simpler ones.

A very basic example of a deep learning model is a multilayers perceptron. That is a feedforward artificial neural network which transforms some set of input values to output

values. Nowadays much more complicated architectures are used, such as deep neural networks. They have multiple hidden layers between the input and the output and each one of them learns appropriate features. The whole representation is then expressed composing all of them. The two main learning paradigms used to train neural networks are supervised and unsupervised learning. In the first one a large data set is collected along with its labels and both of them are provided to the network during training. A cost function is then computed, very often by the backpropagation method, and the parameters are modified in order to reduce the difference between the output and the desired labels. On the other hand, unsupervised training is a technique where the data set is the only input and the goal is to minimize the cost function by recognizing patterns in the unlabelled data and dividing it into clusters. Once the network is trained, a different set of examples, generated with the same distribution of the training set, is given as input to test its performance.

In 1965 Ivakhnenko and Lapa published the first study about an algorithm for supervised deep forward multilayer perceptron. In each layer the best features were selected through statistical methods and passed to the next layer. The network was then trained using a least squares fitting method, not the backpropagation. The first convolutional networks came in the early 1980s and they already had several convolutional and pooling layers. At this point the biggest challenge was how to train networks with multiple layers. Backpropagation of errors was derived in the 1960s, but only in a work by Yann LeCun [7], dated 1989, it was successfully applied to a deep neural network. He developed an algorithm to classify handwritten digits (MNIST) on mail and the system was widely used in the United States. However the training time, which was approximately three days, was a considerable drawback. In 1995 the support vector machine was developed by Cortes and Vapnik and in the following years there were much more studies on these models compared to neural networks. Later, with the increased computational speed of computers and the introduction of graphic processing units (GPUs), the deep neural networks started to draw attention again. They were slower than support vector machines, but could achieve better results with the same amount of data. At that point the main problem was due to the vanishing gradient, the fact that the early layers of the network couldn't learn any features because the learning signal didn't reach them. A few solutions, as for example the recurrent neural networks, were proposed in order to initialize the features of the early layers, which then just needed to be adjusted during the training. In the following years as the GPUs started to become faster, it became possible to train deep neural networks without initializing any features. Ciresan and colleagues won many competition in 2011 and 2012 with their convolutional neural network architectures and the interest in deep learning quickly increased. From 2012 Google, Microsoft and Facebook have started investing on it and the research on this field has rapidly developed.

Today deep learning models have applications in many fields, achieving often state-of-the-art results in the most difficult computer science challenges. The huge amount of data available in many areas combined with the speed of the latest GPUs allow to train neural network with many convolutional layers in a reasonable time. Automatic speech recognition has been totally improved by deep learning techniques. Initially the challenge was approached by Long Short Term Memory (LSTM) a recurrent neural network developed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber [4], obtaining fairly interesting results. In the following years this model became competitive with traditional speech recognizers,

achieving outstanding scores on certain tasks. Since then huge progress have been made, particularly in 2015 Google almost doubled its previous performance with a Connectionist Temporal Classification trained LSTM and this system is now available through Google Voice on every smartphone.

Another field where deep learning is successfully applied is image recognition. In the last years many image processing and computer vision challenges have been tackled by these techniques obtaining excellent results. There is then a wide range of applications in image and video understanding, search algorithms, mapping, medicine, biology, drones, self-driving cars and so on.

The CNN is the most used architecture for image analysis; it performs patterns matching and aggregation operation starting at a pixel level. In 2005 Ning at al. published one of the earliest study of deep network for biological images [10]. They developed a system able to automatically detect and segment cells and nuclei in microscopy images. The system was based on a three convolutional and pooling layers CNN which classified each pixel in cell wall, cytoplasm, nucleus membrane, nucleus or outside medium. The results soon outscored traditional methods as Markov random fields in many tasks. In the following years more layers were added in the networks and in 2013 Ciresan et al [1] won the mitosis challenge at the International Conference of Pattern Recognition with a similar model. Their system was able to detect mitosis in breast histology images with a CNN made of five convolutional and pooling layers followed by two fully-connected ones. Deep learning techniques are currently used for many other tasks in biological image analysis and computational biology and they often outperform standard methods. One of the problems tackled is regression in microscopy image cell counting, a procedure which is required in multiple tasks in biology and medicine. Object counting can be a very long and time consuming process in many real world applications and various automatic systems have been developed. In medicine, for example, it is very important to know the exact number of red and white cells in the blood as it influences the health of the patient, whereas in molecular biology knowing the cell concentration can help to adjust the amount of chemicals applied in an experiments. The automatic cell counting systems follow two main methods: the first one is based on detection and it requires an initial segmentation and the second one needs a cells density estimation. Both of them are widely used in recent studies, achieving state-of-the-art cell counting.

2 The biological basis of deep learning

The human brain contains approximately 86 billion neurons, each one of them connected on average to 7000 other neurons. A neuron is a cell that can be excited by electrochemical inputs from other neurons, process these signals and then transmit them. The synapses are important connections that allow a neuron to pass an electrical or chemical signal to the following one. A neuron is composed of a main cell body called soma, many thin structures called dendrites and a long nerve fibre, the axon. Therefore an electrical impulse pass through the synapse, is received by the neuron through the dendrites and then the outgoing signal flows along the axon. However the output signal is transmitted only if the neuron is activated. This happens if the weighted sum of the input impulses exceeds a certain value, otherwise the cell remains at rest and nothing is passed.

The human brain is then composed of interconnected neurons which form a so called neural network. Our ability to accomplish very complex tasks is therefore based on a very large number of basic processing units, the neurons, which cooperate together. Artificial neural network are inspired by this biological model, but don't quite reach its complexity. Indeed, they manage to do tasks, as speech and image recognition, easy for a humans, but very difficult to solve using traditional rule-based programming.

2.1 The perceptron

The perceptron is a mathematical model of a biological neuron; it was developed in the 1950's by the scientist Frank Rosenblatt. It is a type of linear classifier and the most basic unit of artificial neural network. The electrical signals received by the neurons are represented by binary values for the perceptron. These input are multiply by a certain number, called weight, a bias is added and then the sum over all the contributes is computed. If it exceed a certain threshold, a binary output, either 0 or 1, is fired by the perceptron. As in biological neural networks this output then flows to other perceptrons. Summing up, it can be said that the perceptron makes decisions based on a sum of evidences. Varying the parameters, weights, bias and threshold, different decisions are made.

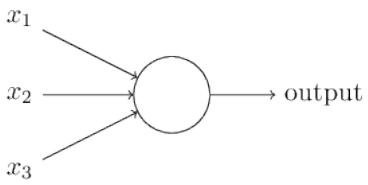

Figure 1: Perceptron unit.

Artificial neural network are made up of many layers of perceptrons. Each layer allows to make decisions at a more complex and abstract level compared to the previous ones. In this way the network can become a very sophisticated decision maker. Learning algorithms have been developed in order to automatically tune the parameters of the perceptrons. This is what is called the training of the network and it can be compared to the fact that even biological neurons connections strengthen or weaken over time, as a consequence of learning and memorizing new things. The learning algorithms are then a very powerful tool in order to make the network behave as desired. However, a small change in perceptrons parameters can cause a considerable variation of the output and this is not the best way to train the network. In order to overcome this issue, a different type of artificial neurons, called sigmoid neurons, has been introduced. These are exactly like perceptrons, with the exception that a small change in their parameters cause only a small change in the output. This is due to the fact that the output is a sigmoid function of the input, whereas in a perceptron the output is given by a step function. In this way every change of the parameters is smoother and the training happens to be more effective.

2.2 Neural network architectures

The first layer of an artificial neural network is called the input layer and it is responsible for extracting the most general features. On the other hand the last layer is called the output layer and it is the one which shows the results. In between these two there are the so-called hidden layers, which analyse in a deeper and more abstract way the data. The design of the input and output layer is usually quite simple, but the hidden layers are more complicated and often the performance of the model depends on their number and size. It is important to notice how each neuron is connected to every neuron of the next layer and contributes to its output. These are called fully-connected layers.

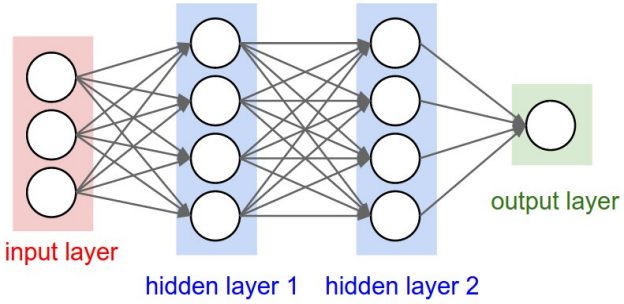

Figure 2: Example of feedforward network: the multilayer perceptron.

The most common neural networks are feedforward, which means that the data flows only in one direction and there are not loops in the connections. These are similar to biological neural network because its units perform their functions collectively and in parallel, without a clear sequence of tasks to be followed. Another type of network architecture is represented by the recurrent neural network. They are based on the fact that its neurons fire only in a limited time period and there are loops between them. Thus they are quite different from how a feedforward network works and they are capable of solving different types of problems.

2.3 Convolutional neural networks

Convolutional neural networks (CNN) are a specific type of feedforward artificial neural network inspired by the animal visual cortex. From studies on the cat visual cortex it is known that the cells are arranged in a complex way, where they are sensitive only to small sub-regions of the visual field, called a receptive field. The receptive fields are tiled to cover the whole visual field. This is represented in the CNN by the characteristic of local connectivity between the neurons of adjacent layers. Indeed, each unit of a hidden layer is connected only to certain units of the previous layer, which together form its receptive field. Any variation outside its receptive field is not seen by the neuron. In addition to this, the CNN has also a shared weight architecture that allow to detect features regardless of their position in the visual field and therefore makes it a shift and space invariant artificial neural network. The human visual system is itself a pattern recognition system and it has the ability of perpetual invariance, which comes from the complex organization of the neurons connections. This ability allows the humans to recognize the same object in very different

conditions, as varying the size, contrast, rotation, orientation and so on. Therefore the CNN resembles the arrangement of information in the human visual system.

3 Convolutional neural networks

As explained in the previous chapter, convolutional neural networks are feedforward neural networks inspired by the animal visual cortex. They have the neurons arranged in such a way that they respond only to a certain sub-region of the previous layer, the receptive field. As for the common neural networks, there is an input and an output layer and in between many hidden layers, which in this case usually are convolutional or pooling layers.

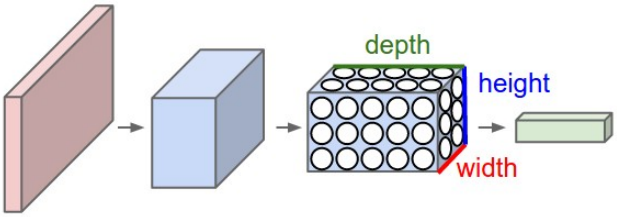

Figure 3: CNN layers arranged in 3 dimensions.

The most important difference is that CNN make the assumption that the input data are multidimensional arrays, like images. The architecture is then structured is such a way to minimize the number of free parameters of the model and therefore optimize the learning process. Indeed, the hidden layers have neurons arranged in three dimensions and so they transform an 3D input volume to a 3D output volume through a differentiable function. The architecture of a CNN is usually made up of a combination of three main types of layers: convolutional layer, pooling layer and fully-connected layer. Starting with the first one, we will explain in details what their characteristics are and what they are responsible for in the network.

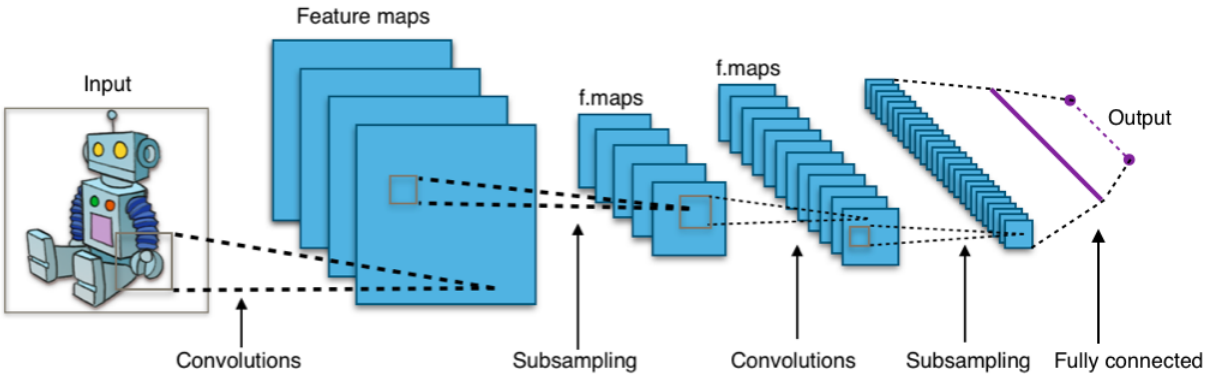

Figure 4: Typical CNN architecture.

A CNN is usually composed of a series of convolutional-pooling layers and of one or more fully-connected layers at the end. In addition to those, others layers like the rectified linear

unit (ReLU) and the deconvolutional ones can be used. More complicated architectures are constantly developed in order to tackle new challenges. The CNN adapt very well to images and compared to other classification algorithms have the advantages of shift-invariance, reduced memory requirements and of a very limited need of pre-processing of the data. For these reasons they are currently one of the most used types of artificial neural networks.

3.1 Convolutional layer

The name convolutional comes from the mathematical operation of convolution, which is very important in image and video processing. If the input data is an image, the convolution involves this image and a 2D kernel and gives as a result a 2D array. During this operation we slide the kernel over the image, each time computing the product of the overlapping values and assigning the sum of the products to the pixel where the kernel is centred.

As mentioned before, in a convolutional layer the neurons are arranged in a 3D volume. Every convolutional layer has a set of learnable parameters, which are usually called filters. Each filters has a small receptive field in the width and hight directions, but it extends to the full depth of the input volume. The connections are then local in space, but necessarily extended in depth depending on the input. This concept is know as local connectivity and it is the first reason that allows to reduce to total number of free parameters. The other reason is given by the parameters sharing scheme. In a convolutional layers volume, in fact, each depth slice of neurons share the same weight and bias. By that we mean that all the neurons in a slide are constrained to have the same parameters. The forward pass of the data is then given by the operation of convolution of a 2D filter over the whole input image, repeated for each depth slide of the layer with the different filter. In this way to total number of parameters of a convolutional layer is drastically reduced.

The convolutional layers have also some important hyperparameters that determine the size of the output volume. By hyperparameters we mean values that are decided initially and then remain fixed during the training process. These are the depth, the stride and the zero-padding. The depth refers to the number of filters that the layer applies to the input data. So this is exactly the depth of the volume of neurons in the layer. The stride indicates how we slide the filter, if it is equal to 1 then the filter is moved one pixel at a time. Finally the zero-padding corresponds to the number of zeroes added around the border of the input volume and it helps us controlling the size of the output. These three hyperparameters determine the width and hight of the output volume by the following formula:

$$\frac{W - F + 2P}{S + 1} \tag{1}$$

where W is the input volume size, F is the receptive field size of the layer, P is the zero-padding and S the stride.

To conclude, the convolutional layers are core building block of the CNN and they have the role of detecting and extracting meaningful features.

3.2 Pooling layer

The pooling layers are often inserted between convolutional layers in a CNN architecture and their main function is to reduce the spatial dimensions of the data. They also allow to decrease the number of parameters and hence to extract features at a different scale in the following convolutional layers.

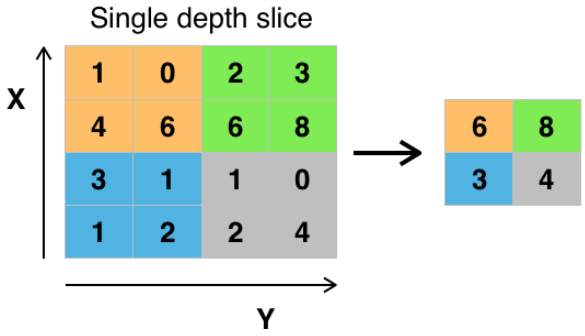

Figure 5: Max pooling with a 2x2 filter and stride = 2

The pooling layers work independently on each depth slice of the input volume and so they reduce the width and hight dimensions. There are many operations used to implement the pooling and the most common one is the max pooling, a non-linear function. The two hyperparameters important in this case are the spatial extend of the filter and the stride; usually the former is 2x2 and the latter is equal to 2 along the width and hight directions. In this case the pooling max operation would simply give the highest values of every four pixel analysed. Therefore the 75% of the pixels would be discarded and the high and width dimensions of the input volume reduced by half.

3.3 Fully-connected layer

The fully-connected layer has the typical architecture of a layer in a multilayer perceptron 2. Each neuron is connected to every neuron of the previous layer and therefore the number of parameters is much higher than for a convolutional layer. Neurons are arranged in just one dimensions and are not spatially located. For this reason fully-connected layers cannot be followed by convolutional layers and are generally used at the end of the architecture of a CNN. They can make non-linear combinations of the features previously extracted and therefore output probabilities of class predictions.

4 Fully convolutional neural networks

Convolutional layers are very well adapted to work with images because they can work with input images of any size and still keep spatial relationship between pixels. On the other hand, fully connected layers have fixed dimensions and throw away spatial coordinates. Fully convolutional indicates that the neural network is composed of convolutional layers without any fully connected layers at the end of the network. In a fully convolutional network every

parameter learned during training is a filter, even the decision-making layers at the end of the network. This allows the network to learn representations and make decisions based on local spatial input, while instead fully connected layers enable the network to learn something using global information where the spatial arrangement is not needed. It is possible to obtain a Fully convolutional network [9] by reinterpreting the final fully connected layers of a CNN as convolutions with kernels that cover their entire input regions.

The main advantage of a fully convolutional network is that it can accept images of virtually any size, since only the fully connected layer expects inputs of a certain size. These networks also benefit from a reduced computational cost since convolutional layers are more computationally efficient than fully connected layers while still achieving a great representation power.

5 Semantic segmentation

The term semantic segmentation is used to identify the process of semantically understanding the role of each pixel in the image. In general, the segmentation of an object in an image is a partition of the image into several "coherent" parts, but there isn't any attempt at understanding what these parts represent. On the other hand semantic segmentation attempts to partition the image into semantically meaningful parts, and to classify each part into one of the pre-determined classes. This is usually achieved by classifying each pixel (rather than the entire image/segment), which is also called pixel-wise classification. To this purpose recently [9] fully convolutional networks trained end-to-end proved themselves to be very effective and they achieved state of the art accuracy in semantic segmentation on some of the most popular databases (PASCAL VOC, NYUDv2 and SIFT Flow). This result was achieved by adapting popular contemporary classification networks (AlexNet [6], the VGG net [11], and GoogLeNet [12] into fully convolutional networks and transfer their learned representations by fine-tuning [2] to the segmentation task.

6 Transfer learning

Deep learning models are indisputably the state of the art for many problems in pattern recognition. Using neural networks with many hidden layers of artificial neurons and millions of parameters, deep learning algorithms exploit both hardware capabilities and the abundance of gigantic datasets. In comparison, linear models saturate quickly and under-fit. But what if you don't have big data? In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size required by the depth of the network. Instead, it is common to pre-train a convolutional network on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the convolutional network to generate representations for novel tasks either as an initialization or a fixed feature extractor for the task of interest, where the dataset may not be large enough to train an entire CNN from scratch. Another common tactic is to take the pre-trained ImageNet network and then to fine-tune the entire network to the novel task. This is typically done by training end-to-end with backpropagation and stochastic gradient descent.

7 Semantic segmentation with fully convolutional neural networks

Object counting can be a very long and time consuming process in many real world applications and various automatic systems have been developed. These automatic counting systems follow different methods, one of which is one is based on detection and it requires an initial segmentation. Semantic segmentation can achieve both detection and segmentation of objects in an image at the same time while benefiting from all the advantages of fully connected neural networks, which can accept input of arbitrary size and are computationally efficient. To this purpose we used a fully convolutional network architecture already proposed in [13] and re-adapted it to a two class classification problem.

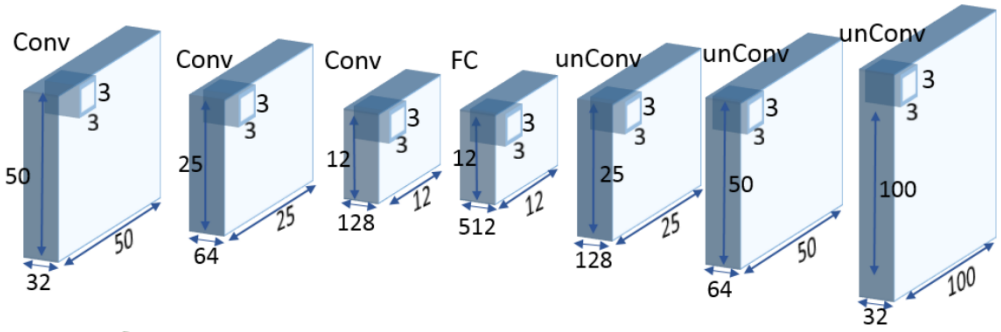

Figure 6: Fully convolutional neural network architecture.

The fully convolutional network architecture used contains in total about 1.3 million parameters and it is shown in Figure 6. All of the kernels are of size 3×3 pixels and are learnt during end-to-end training. The first several layers of the network contains regular convolution-ReLU-pooling layers (referred as Conv in figure 6), then we undo the spatial reduction by performing upsampling-ReLU-convolution operations (referred as unConv in figure 6), mapping the feature maps of dense representation back to the original resolution. The number of feature maps in the higher layers is increased to compensate for the loss of spatial information caused by max pooling. In total three max-poolings are used to aggregate spatial information leading to an effective receptive field of size 38×38 pixels.

7.1 Software and simulations

Initially we trained our network on a computer with Intel Core i3-4130 CPU 3.4 GHz, which resulted in a training time of about 30 minutes, running Caffe only on 1 CPU core (multi threading not enabled). Since training would be faster on the GPU we trained the same architecture using an experimental Caffe branch based on OpenCL developed by AMD, which is compatible with our Radeon R9 270 GPU. This reduced to training time to about 5 minutes.

7.1.1 Caffe

Caffe is a fast and fully open-source framework developed and maintained by the Berkeley Vision and Learning Center (BVLC) and by community contributors [5]. It gives access to deep architectures and it is currently one of the most popular deep learning packages available. Its code is written in efficient C++ and supports Python and MATLAB bindings, with CUDA implemented for GPU computation. Caffe is well-suited for software engineering best practices and also for research use, providing a clear modularity of the code and a good separation of network definition from actual implementation. Its speed is also another reason that makes Caffe widely used for research experiments and industry deployment. Indeed, Caffe can process over 60M images per day with a single NVIDIA K40 GPU, making it likely the fastest available deep-learning framework. Although Caffe was developed for vision at first, it has been used and improved by users in speech recognition, robotics, neuroscience, astronomy and many other fields. In the following, the main highlights of Caffe are described.

- Modularity. The software is designed to be as modular as possible. Many data formats are accepted and various layers and loss functions are already implemented, but an easy extension to new types is allowed.
- Separation of representation and implementation. The Caffe model definitions, written as config files, are well separated from its implementation. Caffe supports network architectures in the form of arbitrary directed acyclic graphs. It takes exactly as much memory as needed for the network. It is possible to switch between CPU and GPU implementation by setting a single flag.
- Python and MATLAB bindings. Caffe supports Python and MATLAB bindings and these languages can be used to construct networks and classify data.
- Pre-trained reference model. Many reference models for different tasks are provided for academic use by the Caffe community, including the famous AlexNet ImageNet model. That helps reproducing the results already obtained and pushing research forward.

There are usually four steps in training a deep-learning model in Caffe. First of all, the data preparation which includes cleaning and storing the data in a format that can be used by Caffe. Secondly, the model definition where the model's architecture is chosen and its parameters are saved in a configuration file with extension .prototxt. Then the solver, responsible for model optimization, is defined and its parameter are saved as well. Finally, the model is trained by executing a Caffe command from the terminal window. Following that the trained model is saved in a file with extension .caffemodel, which can be used later for the test.

7.1.2 SIMCEP

SIMCEP [8] is a computational framework for simulating fluorescence microscope images of cell populations with realistic properties. The tool is implemented with Matlab and all source codes are freely available under terms of GNU General Public License.

The simulation process follows successive steps. First, a synthetic ideal image is generated. This ideal image excludes all error sources originating from the measurement system

and serves as the ground-truth when validating many image processing algorithms for automated image cytometry. Therefore, the ideal image is degraded by the errors due to the measurement system. Finally, as an output of the simulation, an image of a synthetic experiment is obtained, sharing similar properties with the real fluorescence microscope images. With versatile control over simulation parameters, simulated images with varying characteristics can be generated.

With simulated images, the validation can be carried out more efficiently due to the available ground-truth information.

7.2 Implementation details

The implementations of the networks are based on Caffe [5], which is a fast and open deep learning framework developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Back-propagation and stochastic gradient descent are used for optimization. We first pre-trained the neural network on a dataset of synthetic images generated using SIMCEP [8]. We initialized the learning rate as 0.01 and decrease it by a factor of 10 every 20 epochs. The momentum is set to 0.9, weight decay is 0.001, and no dropout is used in the network. The parameters of the convolution kernels are initialized using the Xavier algorithm [3] that automatically determines the scale of initialization based on the number of input and output neurons and helps signals reach deep into the network.

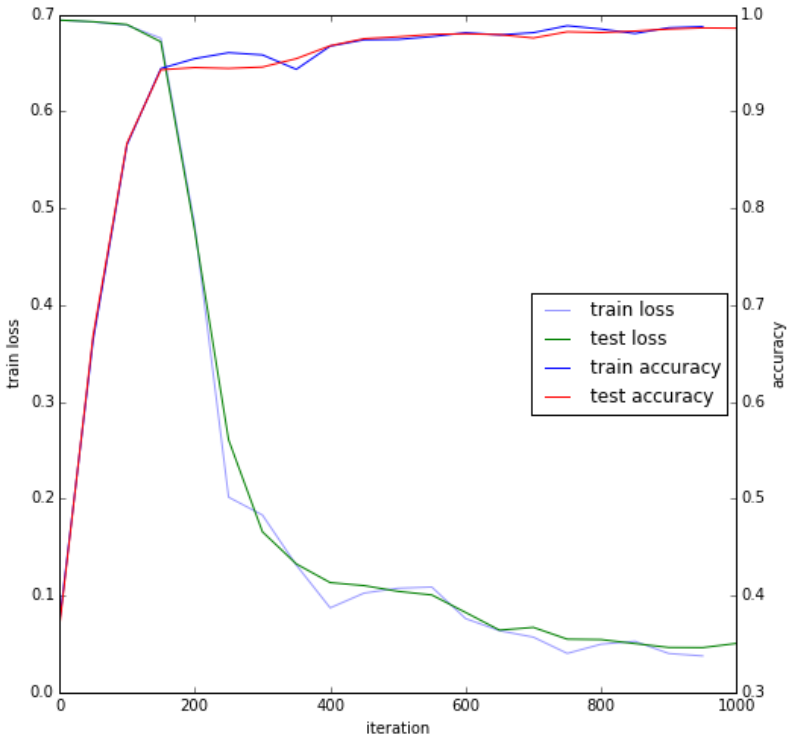

Figure 7: Training logs on the synthetic dataset.

The initial synthetic training set is composed of 200 128×128 patches randomly sampled from computer generated images using SIMCEP. Before the random sampling of these patches, we added simplex noise in the background to half of the synthetic images to simulate a non uniformly illuminated background. Some synthetic image examples are showed in Figure 8.

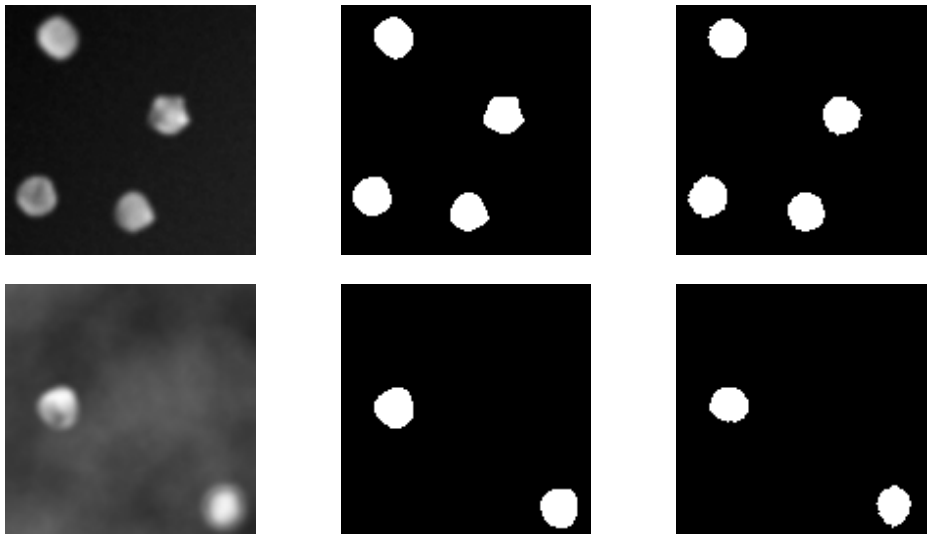

Figure 8: Example of synthetic images of the test set. Left: Randomly sampled patches. Center: computer generated ground truth. Right: output of the network.

After pre-training we fine-tuned the network on the dataset of real images, which is composed of 55 128×128 images and their respective manually annotated ground truths. Given the small number of images in the dataset, we used data augmentation techniques such as rotations (of 90°, 180° and 270°) and mirroring on the y-axis to improve the size of the dataset multiplying the number of images by 8.

7.3 Results

The output of the network is a 2-feature map of the same size as the input image and each feature map contains the probability that that corresponding pixel in the original input image belongs to the first or second class of the classification problem: in the first case the pixel is classified as background while in the second case it's classified as belonging to a cell. Since it's a two class classification problem the probabilities in the 2-feature map are complementary to each other and the sum is equal to 1. By thresholding the output probabilities we can obtain binary images which are then compared to the corresponding ground truths to measure the performance of the network architecture.

In this sense, we used various and different popular parameters and indicators to measure the performance of the network such as accuracy, F-score, recall, precision and the Matthews correlation coefficient. In our case in particular, accuracy and F-score are not very good indicators since the distributions of background and cells in the images are not equal but on average the cells occupy only the 10% of the area in one image. This means that if the

network always classified each input pixel as background it would still achieve about 90% accuracy on average. Instead, the Matthews correlation coefficient (MCC), introduced by biochemist Brian W. Matthews in 1975, is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. It is often used in machine learning as a measure of the quality of binary (two-class) classifications since it takes also into account true and false positives and negatives rates. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction and observation.

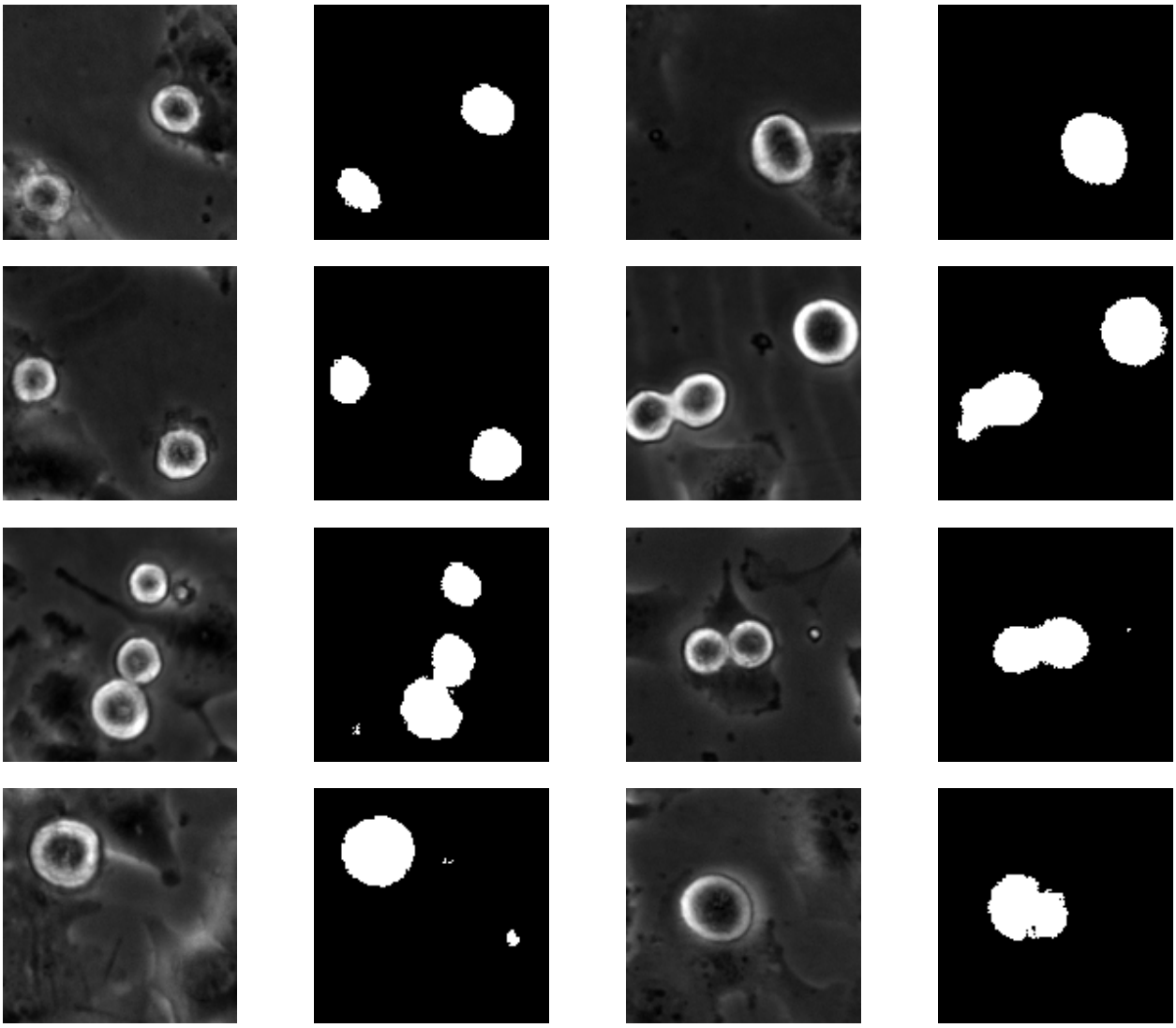

Figure 9: Example of real images of the test set properly classified by the network.

We used the MCC as the main indicator of the performance of the network and also to find the best threshold of the output of the network by selecting the value which resulted in the maximum value of the MCC. Other various indicators of the performance of the network are shown in Table 1 and represented in Figure 10.

Fine-tuning	specificity	precision	recall (TPR)	accuracy	F-score	MCC
Before	0.96	0.58	0.52	0.91	0.55	0.50
After	0.98	0.81	0.73	0.95	0.77	0.74

Table 1: Various indicators of the performance of the network before and after the fine-tuning and corresponding to a threshold value of 0.25 (before) and 0.67 (after).

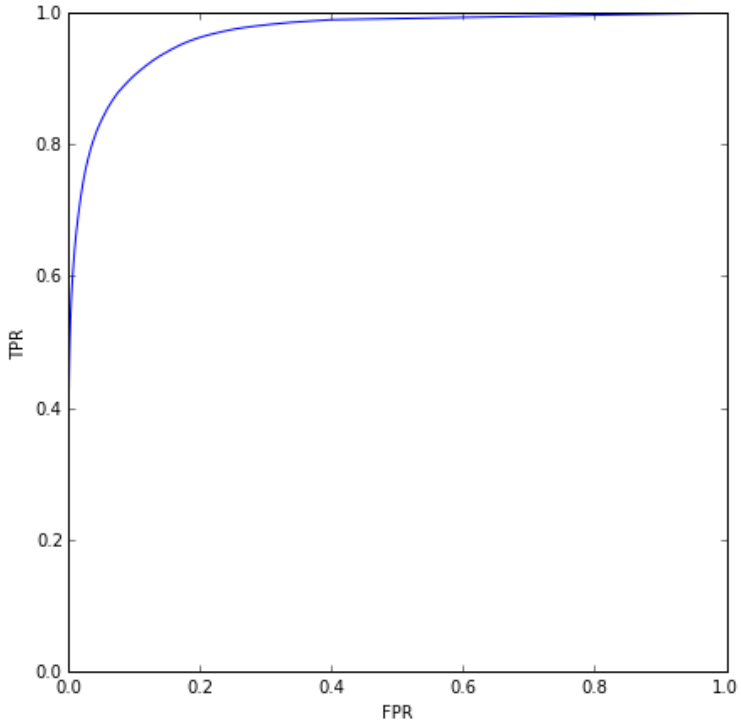

Figure 10: TPR: True Positive Rate. FPR: False Positive Rate. Every point of the plot corresponds to a threshold value.

The high value of specificity (1-FPR: False Positive Rate) indicates that the network recognizes very well the background as such and has few false positive pixels identified as cells. The precision refers to the rate of cell pixels correctly classified over the total number of cell pixels found by the network, while the recall (True Positive Rate) indicates the rate of cell pixels classified correctly over the total number of cell pixels in the test dataset. The accuracy is the rate of the number of pixels properly classified by the network. The F-score is the harmonic mean of precision and recall.

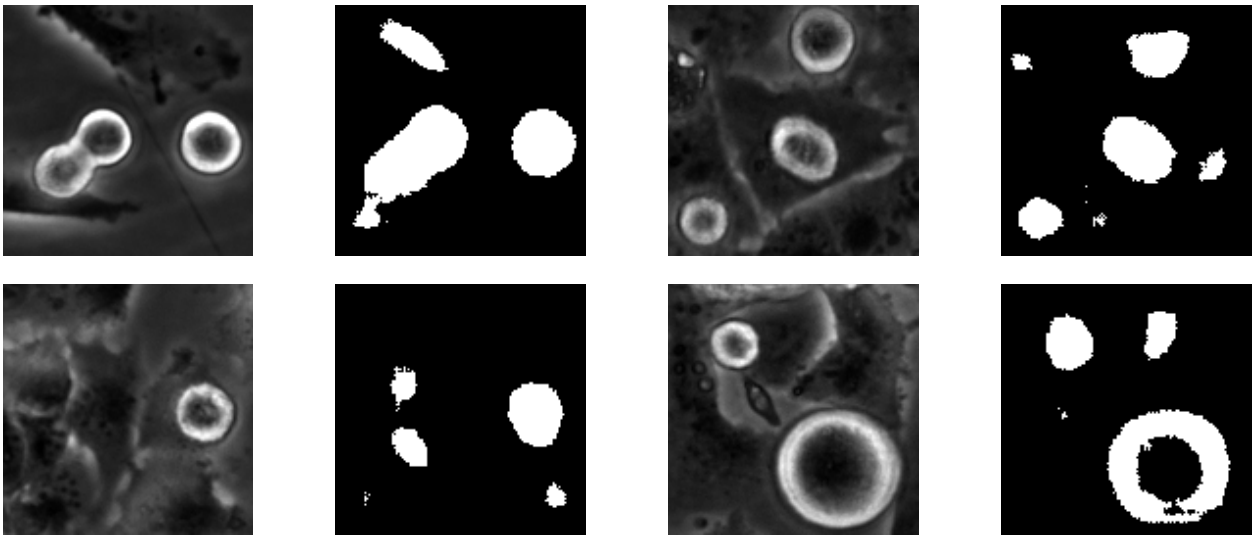

Figure 11: Example of real images of the test set improperly classified by the network.

In Figure 11 we show some examples of images of the test set improperly classified by the network. As we can see, the network struggles to classify correctly high-contrast background pixels which results often in false positive pixels, although these false identified cells are typically of smaller sizes than the other real cells and could be easily removed by post-processing of the images (using for example morphology operations). Alternatively, we could increase the total number of parameters of the network (but paying attention to avoid overfitting, since our dataset of real images is relatively small) and reduce the number of pooling layers in order to avoid too much spatial information loss. We also see that the network has problems identifying larger cells. This could be avoid increasing the receptive field size, which is the portion of image seen by each neuron in the deep layers of the network, by increasing the size of the kernels in the convolutional layers.

8 Conclusions

We developed a deep learning cells recognitions system using fully convolutional neural networks and semantic segmentation. The results on the dataset of synthetic images were improved by fine-tuning the model on the dataset of real images. We showed some examples of is images resulting from the network output and measured the performance of the model using various indicators typical of binary classification. Finally we suggested some ways to improve the network architecture by analyzing its results.

References

[1] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer, 2013.

[2] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[8] Antti Lehmussola, Pekka Ruusuvuori, Jyrki Selinummi, Heikki Huttunen, and Olli Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE Transactions on Medical Imaging*, 26(7):1010–1016, 2007.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[10] Feng Ning, Damien Delhomme, Yann LeCun, Fabio Piano, Léon Bottou, and Paolo Emilio Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, 2005.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[13] Weidi Xie, J Alison Noble, and Andrew Zisserman. Microscopy cell counting with fully convolutional regression networks.