A Tutorial on Deep Neural Network based Semantiv Segmentation

Marvin Teichmann

January 25, 2016

Abstract

In this work we evaluate the potential of Deep Convolution Neuronal Network (DCNN) for segmentation applications in Medical Informatics. We give an overview of the most important publication in the fields, an comprehensive explanation of the most common approaches used and a short review on how this techniques can be applied to tasks in the field of Medical Informatics.

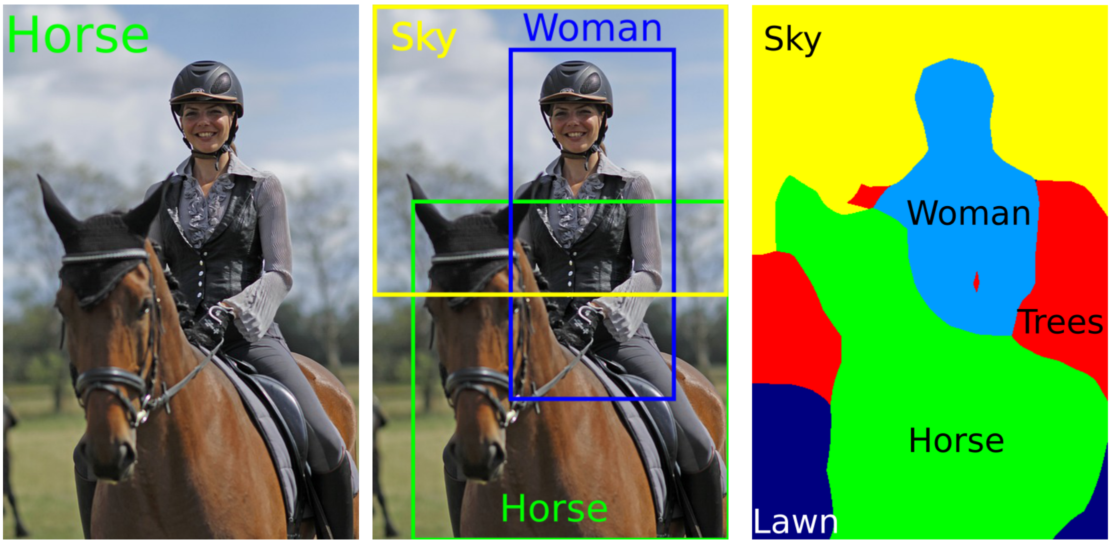
1 Introduction

This Tutoial aimes to give a comprehensive introduction to neuronal network based semantic segmentation. It is directed to an audients, who is already familier with semantic segmentation using traditional shallow learning methods but not with the novel deep learning approaches. The work is specied with examples and framed around applications of the field medical informatics, however the presented theory generalizes easily to other fields.

Deep Neuronal Networks and in particular Convolution Neuronal Networks (CNNs) have been the driving factor of innovation in Computer Vision in the past years. Since Krizhevsky et. al [KSH12] proposed a novel and award winning Network architecture, Deep CNNs have broken new records in many Computer Vision Domains including Classification [KSH12, SZ14, SLJ$^+$14], Localization and Detection [GDDM13, SEZ$^+$13] and Semantic Segmentation tasks [LSD15, ZJR$^+$15, PCMY15].

Especially Semantic Segmentation does have several interesting applications in medical informatics, including ... . A recent survey of the traditionally used methods in this area are published in [TB16]. Considering the very recent break-throughs in semantic segmentation using deep learning models [LSD15, ZJR$^+$15], the next logical step is to adapt such models to medical informatics tasks. This paper aims to help researchers in the field of medical informatics to do so.

This paper is structured as fellows: Section 2 defines classification, localization and semantic segmantation. These computer vision tasks are closely related to each other an deep learning based approaches to heavily take advantages of these relations. Section 3



(a) **Classification:** The output is a one-dimensional label of the entire image. In this case the image can be classifed as showing a cat.

(b) **Localization and Detection:** The output are one or several bounding boxes together with a class label for each box.

(c) **Segmentation:** The output is a class label for each pixel. The task is hence to regognize objects on a pixel-level.

Figure 1: A visual comparision of computer vision task. The input in all three cases is the same image. The expected output is drawn on top of the input image.

gives a short introduction to neuronal networks and CNNs in general. Section 4 than explaines how deep CNNs can be applied to Semantic Segmantation tasks. Finally Section 5 discusses how medical informatics can benefit from neural network based semantic segmentation.

2 Computer Vision Tasks and Datasets

Three very important Computer Vision Tasks are *Classification*, *Detection* and *Semantic Segmentation*. In all three tasks the algorithms receive a single image as input. However the output differs. In *classification* a one-dimensional label is expected, representing the contained of the entire image in a single nominal value. The classes are usually choosen such that these values can be interpreted by humans (i.e. image of a car). In *Localization* and *Detection* the output of the algorithm is a set of bounding boxes, each assigned with a class label. In the Large Scale Visual Recognition Challenge 2015 (ILSVRC2015) taxonomy the only difference between localization and detection is the evaluation. In localization false positives are not punished, but output is restricted to five bounding boxes. In Detection on the other Hand, arbitrary many bounding boxes are excepted as output, but wrong output gets punished. One challenge in Detection is, that the data-set contains images without any bounding-boxes. Hence, the algorithms need to learn an additional background class. In segmentation the output is dense. The task is to predict one label to each pixel of the image. A comparision between the three task is visualized in Figure 1.

2.1 Relation between Tasks

There is a hierarchie between the descripted tasks. In the order Classification, Localization, Detection and Segmentation each task can be viewed as subtask of the previous one. Where Localization and Detection introduce a spatial component to Classfication and Segmentation is a fine-grine version of Detection/Localization. Additionally a segmentation output can be easily transfered to a Detection output by computing the bounding boxes of the segmented area. Detection to Localization by selecting the five non-background bounding boxes with highest propability and segmentation to classification by choosing the class label of the largest bounding box as image label.

Conversly it is also possible to build a segmentation algorithm given a perfekt classifikation oracle. On order to archive that, one just traines a classifier for each pixel individually and combines the result. In practise, this insight is usually utilised in a sliding window approach [LZW14, BKTT15]. A classifier is trained to classifier the center pixel of quadratic image section (e.g. of size 50 × 50 pixel). The classifier is than applied on each spatial location of the image, like a convolution operation.

3 Convolution Neuronal Networks

In order to reasonable train deep models on the high dimensional image data models which contain strong prior knowledge are required. Having prior knowledge about image data allows us to dramatically reduce the capacity (i.e. amount of parameters) without sacrificing much accuracy. CNN's rely on the following two strong assumptions.

1. translation invariance and stationarity of statistics
2. locality of pixel dependencies

Stationarity of statistics is archived by applying a translation invariance functions on each layer. Locality of pixel dependencies is accomplish by making the kernel of this function only depend on a relative small and dense reception field.

3.1 Definitions and Notation

Multi-Layer perceptrons operate in layer, each of shape $h \times w \times d$, where h and w are spatial coordinates and d is the channel size. Let $x_{ij} \in R^d$ be the data of Layer l, than Layer $l+1$, given by $y_{ij} \in R^{d'}$ is given by computing a layer function

$$y_{nm} := F_{nm}(\{x_{ij}\}_{0 \leq i \leq h, 0 \leq j \leq w})$$

In CNN's F_{ij} is chosen to be translation invariant. F_{ij} can therefore be described by a kernel operation $f : R^k \to R$. The meta-parameter k is called *kernel size*, it usually has shape $k = n \times n$ with $n << h, w$. The function f is than applied to every location in a sliding-window fashion. Sometimes s pixel are skipped in each dimension resulting in a down-sampling of factor s. The meta-parameter s is called *stride*. Hence we obtain

$$\begin{aligned} y_{ij} &:= F_{nm}(\{x_{ij}\}_{0 \leq i \leq h, 0 \leq j \leq w}) \\ &= f_{ks}(\{x_{s \cdot n+i, s \cdot m+j}\}_{0 \leq i,j \leq k}). \end{aligned}$$

Layer $l + 1$ has hence shape $(h - k)/s \times (w - k)/s \times d'$, where d' correspond to the number of filter applied. *Padding* can be applied in order to avoid the loss of information at the border of the feature map in each layer. The output shape than is $h/s \times w/s \times d'$.

3.2 Layer Types

CNN's are build using three different layer types. Namely these are *convolutional*, *pooling* and *activation* layers.

3.2.1 Convolutional Layer

Convolutional layer implement a learnable convolution operation inside the neural network model. To archive that f_{ks} is chosen to be linear function (i.e. a matrix). Observe that this can be viewed as a special case of an MLP, where curtained weights are enforced to be equal or fixed to be zero. The parameters can therefore be learned using a back-propagation approach. In computer graphics convolutions are a very important tool. They can be used for a variety of tasks including edge and area detection, contrast sharpening and image blurring. Having learnable convolution kernels is therefore a very powerful tool. In convolutional layers stride is usually choose to be $s = 1$, unless the kernel-size is relatively big ($k \geq 7$). [KSH12, SZ14, SLJ+14].

3.2.2 Pooling Layer

The pooling layer applies non-learnable function, which collects a summary statistic about a region of the feature map. Typical choices are max- or mean-pooling, computing the corresponding function on its input region.

For the pooling layer typically s is choose to be k (VGG, GoogLeNet, OverFeat), although overlapping pooling has been successfully applied (Alexnet). Typical choices for the kernel size include 2×2 or 3×3. ([KSH12, SZ14, SLJ+14]).

Applying pooling has two advantages: Firstly it naturally reduces the spatial dimension enabling the network to learn more compact representation if the data and decreasing the amount of parameters in the succeeding layers. Secondly it introduces robust translation invariant. Minor shifts in the input data will not result in the same activation after pooling. The drawback of pooling however is, that fine-grained spatial information are lost in the process. This is a negligible disadvantage for non-spatial tasks such as classification but comes severe in segmentation.

3.2.3 Activation Layers

To enable the CNN to learn nonlinear function it is crucial, that some kind of nonlinearity is applied between layers. Otherwise the Network is equivalent to the concatenation of linear functions, hence a linear function itself which can be equally represented using a single layer. Nonlinearities are usually one-dimensional functions $f : R \rightarrow R$, applied to each coordinate individually, hence they can be viewed as an kernel operation with size $k = 1$ and stride $s = 1$ in the above notation.

Classical choices for nonlinearities are the hyperbolic tangent *tanh* and the sigmoid function $f(x) = (1 - exp(-x))^{-1}$. Recently ReLU Nonlinearities [KSH12](AlexNet, Bolzmann) have gained a lot of popularity [KSH12, SZ14, SLJ^{+}14]. ReLU Nonlinearities have several advantages over traditional nonlinearities. Firstly they are very fast and efficient to compute on GPU. Secondly it does not suffer from the gradient vanishing problem and lastly empirical results show, that training converges several times faster than with other Nonlinearities.

3.2.4 Fully Connected Layers

Todo: descripe fully connected layers and their relation to convolution layers

3.3 Fine-Tuning and Transfer Learning

One of the main disadvantages of deep learning based approaches is the large amount of labeled training data required to train a deep model. However this can be counterbalanced be utilizing the large amount of freely avaible image data. To archive this *fine-tuning* and *transfer learning* are two very important deep learning training paradigmens.

3.3.1 Fine-Tuning

In fine-tuning a model is first trained with different image data. A model witch is supposed to classify organs, for example might first be trained using a subset of ImageNet of Cifar10 data. Although the data is quite different deep layers are still initialized usefully. Deep Layers are usually responsible for detection low level features such as edges and other simple patterns. When training with the task related data, the network is than initialized with the weights obtained by the first training. In order to benefit from the feautes learned on the lower layers, the weights of this layers are sometimes fixed or learned with a lower rate. Additionally higher layers are sometimes discarded, if features of high abstractation are not usefull.

3.3.2 Transfer Learning

As discussed in Section 2.1 there is a strong relation between different computer vision tasks. Deep Models can diretly benefit from these relations using *transfer learning.* In *transfer learning* one of the tasks is learned using data labeled for a different task. This can be done as deep features are shared between architectures.

In practise usually classification data is used to train segmentation of localization networks. This is done, as labeling classification data is much cheaper and therefore avaible in much larger amounts.

4 Neural Networks for Segmentation

After the overwhelming successes of DCNNs in image classification, there as been a lot of effort to apply this models to further computer vision tasks. Early ideas include the use of CNNs based classifiers in combination with traditional classifiers [GDDM13]. Other authors used the idea described in Section 2 to tackle segmentation as pixel-wise classification problem using a sliding-window approach together with a classification network [GCM$^+$13], [SEZ$^+$13], [BKTT15] [LZW14]. These authors profit from the inherent sliding window efficiency of CNNs, described in Section 4.1. A recent break-trough has been achieved with the novel Fully Convolutional Networks (FCN) [LSD15] architectures. FCN are an architecture specifically designed for Semantic Segmentation. FCN combine the sliding window efficiency with a deconvolution architecture for upsampling and a transfer learning approach. FCNs and there deeper successors [ZJR$^+$15], [NHH15] [PCMY15] are currently the state-of-the art in several semantic segmentation benchmarks. We will descripe the mechanics behind FCNs in detail in Section 4.2.

4.1 Sliding Window efficiency in CNN's

Opposed to other classification approaches ConvNets are inherently efficient when applied in sliding window fashion. Their translation invariant structure allows to benefit from computation on overlapping patches on the images. Of high practical relevant is also, that the result itself will be a ConvNet, that means any ConvNet C can easily be transformed in a ConvNet C', whose output is equal to applying C in sliding window fashion. This idea can hence be very efficiently implemented in any ConvNet framework, without much effort. The only downside is, that C' will have a stride s equal to the product of all strides in C.

The reason for the efficiency is, that translation invariant computing is computational traceable. Let F, G function, computing a layer as defined in Section 3.1. Let f, g kernels corresponding with sizes k, k' and stride s, s', corresponding to F, G respectively. Than $F \circ G$ is obtained by the kernel $f \circ g$, which has kernel size $k + (k-1)s'$ using stride $s \cdot s'$. For networks only consisting of convolution and pooling layers one can therefore simple increase the size of the input layer at evaluation time. The output will be equivalent to apply the original network in sliding-window fashion. Common ConvNet architectures typically have one or more fully-connected layer producing the final classification output. This layers can be replaced by convolutional layers using the trick described in Section 3.3.2.

The main downside of this approach is the stride of the overall output stride. The output image of the transformed ConvNet C' will be an low resolution image. The input image will be down sampled by a factor of s corresponding to the product of all strides being applied in C'. On most network architectures s becomes quite large, e.g. 32 on VGG16, a network using pooling very cautiously.

To avoid downsampling while still profiting from the sliding window efficiently of CNNs it is possible to use shift-and-stitch. Each layer which is associated with an stride s is feeded with s^2 shifted versions of the input. (Each with a different shift in x or y dimension). The output can than be stitched together in order to obtain an image of original resolution. The result is than equivalent in applying sliding-window with stride 1. However the computational advantage of applying strided pooling is lost in the procedure (while the model advantage remains). Fast-scanning [GCM+13] describes a trick to efficiently perform this computation. This idea is used in several publications [SEZ+13, HWT+15].

4.2 FCN

The FCNs [LSD15] Architecture builds up on the ideas presented of Section 4.1. Similar to earlier approaches they are using existing classification networks and transform them into segmentation networks using the inherent sliding-window efficiency of ConvNets. However opposed to earlier approach they are not trying to avoid downsampling as part of the progress, but they are using a trainable upsampling layer to archive high resolution output. Further ingredients of their approach is a skip-architecture to preserve fine-grained information and a transfer-learning approach making it possible to train a very deep net.

4.2.1 Deconvolution

Todo: detailed explanation of deconvolution

4.2.2 Skip-Architecture

In the higher layer of the encoder network fine spatial information is lost due to pooling stride, which cannot be reconstructed with upsampling. To overcome this problem a skip architeture is introduced. The robust and more accurate predictions from the high level layer are combined with predictions from lower layers, whcih contain more detail. The upsamling operation is than trained on the combined feature maps enabling it to learn accurate and fine segmentation.

4.2.3 Transfer Learning

One of the strengths of the FCN approach is the use of transfer learning. Training is done in two steps. First the classification network is done using ImageNet Classification Data. Afterwards the last layer of the architecture is replaced by a deconvolution layer. Only than the Network is fine-tuned on segmentation data.

In the publication several the FCN approach was applied to AlexNet, VGG16 and GoogLeNet. The best results where achieved on VGG16. A possible explanation for this is, that VGG16 uses stride and pooling most cautiously preserving spatial information best.

4.3 Extensions of FCN

Several extensions of FCN have been proposed. All of them are using FCN on VGG16 bases to obtain a low resolution spatial encoding of the image. The main differences between the approaches are how the how upsampling is performed.

There have been quite a lot of success on building Conditional Random Fields (CRFs) on top of the FCN structure [ZJR+15]. These architectures are currently providing the best results in the Pascal VOC challenge [CPK+14], [LSRvdH15].

[NHH15] and [BKC15] proposed a slightly different approach. The designed a deep decoding network to perform the upsampling. Where each layer of the decoding network corresponds to a pooling layer of the VGG network. The upsampling itself is not trained but computed directly using the max pooling indices. Trained convolution layers are used between the upsampling operations to refine the results. The main downside of this approaches is, that the networks need a large amount of strong labeled data as they are fully trained end-to-end. This problem is relaxed in [HNH15], by introducing transfer-learning to deep deconvolution networks.

5 Application in Medical Informatics

This section is structured as fellows: Section 5.1 described gives a brief overview where CNN based semantic segmentation is already used. Section 5.2 is a case study giving a detailed description fcn can be applied to a common task in Medical Informatics.

5.1 Applications in Literature

5.2 FCNs for computed aided surgery

References

[BKC15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[BKTT15] S. Bittel, V. Kaiser, M. Teichmann, and M. Thoma, "Pixel-wise segmentation of street with neural networks," *CoRR*, vol. abs/1511.00513, 2015. [Online]. Available: http://arxiv.org/abs/1511.00513

[CPK$^+$14] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*, vol. abs/1412.7062, 2014. [Online]. Available: http://arxiv.org/abs/1412.7062

[GCM$^+$13] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," *CoRR*, vol. abs/1302.1700, 2013. [Online]. Available: http://arxiv.org/abs/1302.1700

[GDDM13] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[HNH15] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," *CoRR*, vol. abs/1506.04924, 2015. [Online]. Available: http://arxiv.org/abs/1506.04924

[HWT$^+$15] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving," *CoRR*, vol. abs/1504.01716, 2015. [Online]. Available: http://arxiv.org/abs/1504.01716

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[LSD15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR (to appear)*, Nov. 2015.

[LSRvdH15] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel, "Efficient piecewise training of deep structured models for semantic segmentation," *CoRR*, vol. abs/1504.01013, 2015. [Online]. Available: http://arxiv.org/abs/1504.01013

[LZW14] H. Li, R. Zhao, and X. Wang, "Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification," *CoRR*, vol. abs/1412.4526, 2014. [Online]. Available: http://arxiv.org/abs/1412.4526

[NHH15] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *CoRR*, vol. abs/1505.04366, 2015. [Online]. Available: http://arxiv.org/abs/1505.04366

[PCMY15] G. Papandreou, L. Chen, K. Murphy, and A. L. Yuille, "Weakly- and semi-supervised learning of a DCNN for semantic image segmentation," *CoRR*, vol. abs/1502.02734, 2015. [Online]. Available: http://arxiv.org/abs/1502.02734

[SEZ$^+$13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229

[SLJ$^+$14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[SZ14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[TB16] M. Thoma and S. Bodenstedt, "Semantic segmentation: An overview," 2016.

[ZJR$^+$15] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," *CoRR*, vol. abs/1502.03240, 2015. [Online]. Available: http://arxiv.org/abs/1502.03240