CNN Report

by Bhargav Bhatkalkar

Submission date: 09-Feb-2019 06:37PM (UTC+0800)
Submission ID: 1075415314
File name: Aditya_Report.pdf (1.4M)
Word count: 3693
Character count: 18731

Report On Machine Learning And Deep Learning Principles

Adithya Narayan

February 9, 2019

1 Neural Networks

1.1 Introduction

Inspired by the functioning of a biological nervous system [1], a neural network is essentially a information processing paradigm. A neural network, while not an algorithm, is a framework which allows for different machine learning algorithms to work in tandem to predict the nature of the output based of a complex set of input data points. More informally put, it is possible to arfue that a neural network is simply an incredibly powerful system that can be used to fit functions through a given set of data points.

1.2 Structure Of a Neural Network

Neural networks are built up from fundamental units known as neurons [2]. As can be seen in Figure 1, these neurons are stacked together with weighted connections between the neurons in successive layers. The first layer, or the input layer, consists of an arbitrarily long feature vector which takes in the input data. This is followed by the hidden layers which take inputs from the previous layer, be it an input layer or just another hidden layer, based off of the weights that are associated with the neurons which are connected to it. Finally, the output layer ouput layer produces the ouput of the network as a arbitrarily long vector.

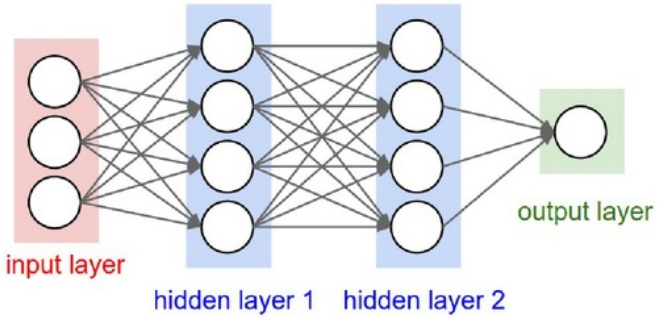

Figure 1: A Neural Network with two hidden layers.

1.3 The Cost Function

A cost function is essentially an equation which is used to determine the error of the ouput of the neural network. It uses a particular function to compare the true output from the dataset aganist

the output produced by the neural network for a given set of weights. Primarily, the goal of any neural network is to minimize the output value of the cost function.

Based off of the cost function used in logistic regression, the regularized cost function for a neural network is given by,

$$J(\Theta) = -\frac{1}{m}\Big[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} log(h_\theta(x^{(i)}))_k + (1-y_k^{(i)}) log(h_\theta(x^{(i)}))_k\Big] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

1.4 Backpropagation

Backpropagation is an algorithm that is used in a neural network to determine a gradient which is then used to determine the weights that need to be used in the network [3]. More simply put, the goal of this algorithm is to minimise the cost function $J(\Theta)$. For this, we require the partial derivative of the cost function $J(\Theta)$ [4]. The procedure for this is outlined below.

We begin by taking a training dataset $(x^1, y^1), (x^1, y^1), ..., (x^m, y^m)$ which contains m training examples. Additionally, consider a L layered neural network. We define δ_j^l as the error in the ouput of node of layer l.

To perform backpropagation, we start off by setting $\Delta_{ij}^{(l)} = 0$. Then the input dataset is iterated over as follows: For $i = 1 : m$,

1. Set $a^{(1)} = x^{(i)}$
2. Perform forward propagation to compute $a^{(l)}$ in the deeper layers.
3. Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
4. Compute $\delta^{(L-1)}, \delta^{(L-2)}, \delta^{(L-3)}, ...$
5. Then compute $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)}\delta_i^{(l+1)}$

After this outside the for loop, $D_{ij}^{(l)}$ is computed as follows,

$$D_{ij}^{(l)} := \frac{1}{m}D_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}, j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m}D_{ij}^{(l)}, j = 0$$

2 Convolutional Neural Networks

2.1 Introduction

For a typical neural network, the input neurons form a vector. Generally, this type of architecture works well enough when the input data is small and is reasonable when expressed as an $N \times 1$ matrix. However, when it comes data such as the ones involving images, this input vector becomes incredibly large. For instance, if you were to consider an image of dimensions $200 \times 200 \times 3$, the neural network would require 12000 neurons with an equal number of weights. Additionally, since the hidden layers too require a reasonable number of neurons, the overall size of the neural network becomes staggeringly large fairly quickly. Hence, an architecture is required to be able to process inputs of higher dimensions more efficiently. For these kinds of input data, Convolutional Neural Networks can be used.

2.2 Architecture

Convolutional Neural Networks arrange their neurons in a more sensible different way. The neurons of a convolutional neural network are arranged in such a fashion such that they have volume. Each acivation volume of a convolutional neural network posseses width, depth and height. Additionally, for the sake of efficiency, not all the neurons between two activation volumes are connected together.

2.3 Layers Of a CNN

Given that there are a number of proposed architectures for CNN's [6], the layers that are in a particular CNN depends on the selected architecture. However, there are layers that are generally seen in most CNN's. These are briefly gone over in this section.

2.3.1 Convolutional Layer

This layer computes the ouput of the neurons that are connected to a small region in the input layer. Here, a dot product is performed between the neuron's weights and the region they are connected to in the input layer [5]. Simply put, this layer consists of a set of earnable filters. While each of these filters only cover a small region in the input layer, they extend through the full depth of the input layer. During the process of training, these filters are convolved across the height and width of the input layer. This generates a 2-D activation map that ouputs the response of that filter at every spatial position. This is repeated for every filter that exists in this layer. For instance, if we have 12 filters in the convolution layer, 12 each will produce a 2-D ouput matrix. These ouput maps are then stacked to generate the depth of the ouput. A diagrammatic representation of this can be seen in Figure 2.

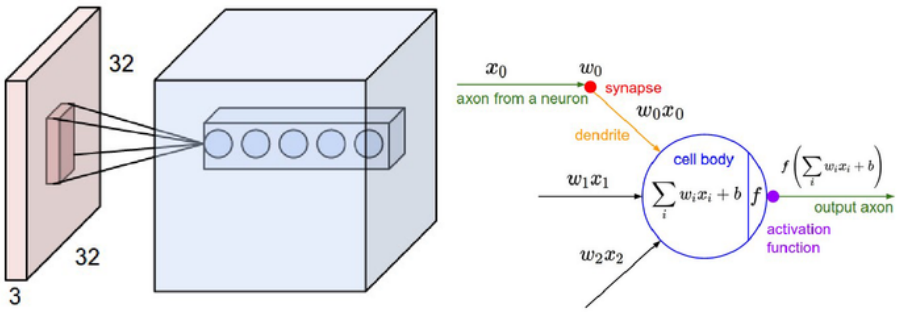

Figure 2: A convolutional layer.

2.3.2 Pooling Layer

Quite often, a pooling layer is added between convolution layers in a convotional neural network. It progressively reduces the spatial size of representations to reduce the number of computations required by the network. Essentially, it serves as a method to control overfitting of the data. As can be seen in Figure 3, this layer acts independently on every slice of the input; resizing it spatially using the MAX operation. Besides max pooling, which is described above, there are a number of different pooling methods such as l_2-norm pooling and average pooling. However, most of these have fallen out of favour compared to max pooling due to max pooling's improved performance [7].

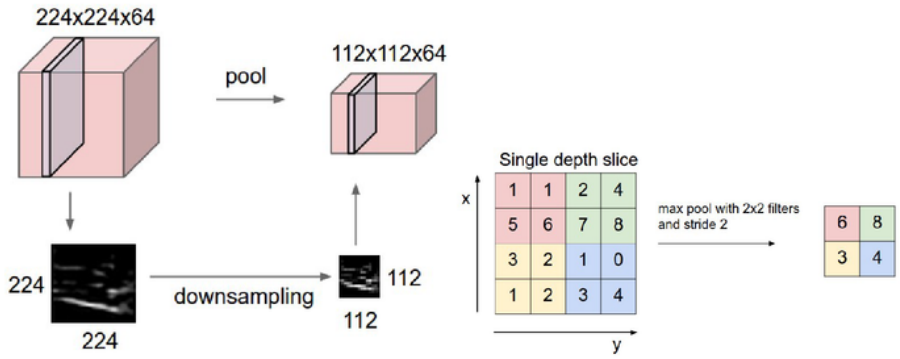

Figure 3: MAX Pooling operation.

2.3.3 ReLU Layer

A shorthand for 'rectified linear unit', this layer applies an activation function [8] of the form,

$$f(x) = max(0, x)$$

The purpose of this layer is to increase the non-linear properties of the descision function and that of the overall network without affecting the receptive field hyperparameter of the convolutional layers. Besides ReLU, other functions,such as $\sigma(x) = (1 + e^{-x})^{-1}$ or $f(x) = tanh(x)$, can be used for improving the non-inear properies of the system. However, ReLU is often prefered due to its training speeds and because it doesn't significantly reduce the general training accuracy [8].

2.3.4 Fully Connected Layer

A fully connected layer is essentially a layer of neurons which have connections to all activation in the preceeding layer. This layer is often used for high level reasoning in a CNN. Tpically, activations in this layer are computed by matrix multiplicaition followed by addition to a bias matrix(a bias offset).

3 Semantic Segmentation

3.1 Introduction

Semantic segmentation falls in the field of image segmentation. Often, when processing an image, it is necessary to segment the image into different parts so as to be able to distinguish different features of the image. For instance, suppose you had an image containing a person cycling on a road. And suppose you needed to rain a network so as to be able to tell what type of bikes are most likey to be used on a specific type of road. To be able to effective automate this process, it would be uselful if the computer could distinguish the bike from the road; hecnce the necessity of image segmentation.

Semantic segmentation holds on to the same end obective. The goal of this algorithm is to label each pixel in an image with a corresponding class of what is being represented. An import distinction to make however, is that by semantic segmentation, you are not distinguishing different

insatnces of the same class. For instance, if you were to have two obects of the same class, semantic segmentation does not help In these cases, instance segmentation is preffered.

Simply put, every pixel is in the image is associated with a particular output class and a resultant mask is generated. This allows us to process defferent objects in the image by operating exclusively on the masks generated through semantic segmentation.

3.2 Architecture

Multiple approaches are possible for designing the architecture for semantic segmentation. Here some of the basic architectures used for semantic segmentation will be described.

3.2.1 Naive Approach

A fairly straightforward and simple method of doing semantic segmentation is by stacking convolutional with the ouput as a segmentational map. In this method, the CNN directly maps the input image to its correspoinding segmentation through succesive feature mapping across the layers.

Typically, for deep convoutional networks, earlier layers tend

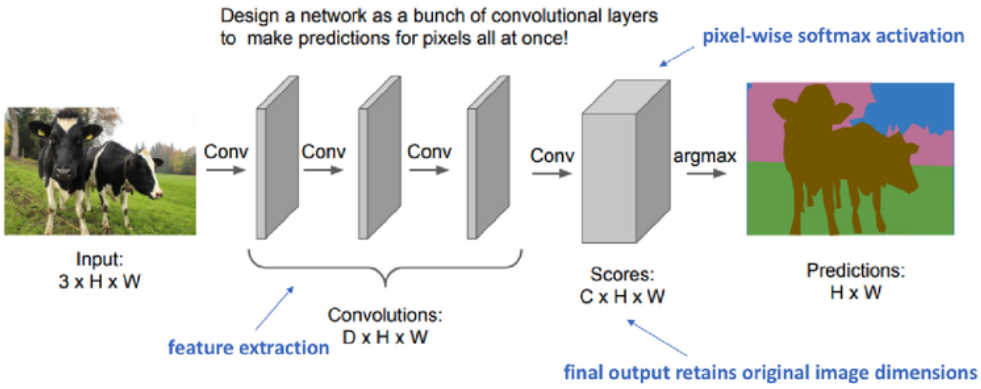

Figure 4: A Naive Model For Semantic Segmentation.

In order to maintain expressiveness, we typically need to increase the number of feature maps (channels) as we get deeper in the network.

This didn't necessarily pose a problem for the task of image classification, because for that task we only care about what the image contains (and not where it is located). Thus, we could alleviate computational burden by periodically downsampling our feature maps through pooling or strided convolutions (ie. compressing the spatial resolution) without concern. However, for image segmentation, we would like our model to produce a full-resolution semantic prediction.

3.2.2 Encoder/Decoder Architecture for CNN

One popular approach for image segmentation models is to follow an encoder/decoder structure where we downsample the spatial resolution of the input, developing lower-resolution feature map-

pings which are learned to be highly efficient at discriminating between classes, and the upsample the feature representations into a full-resolution segmentation map.

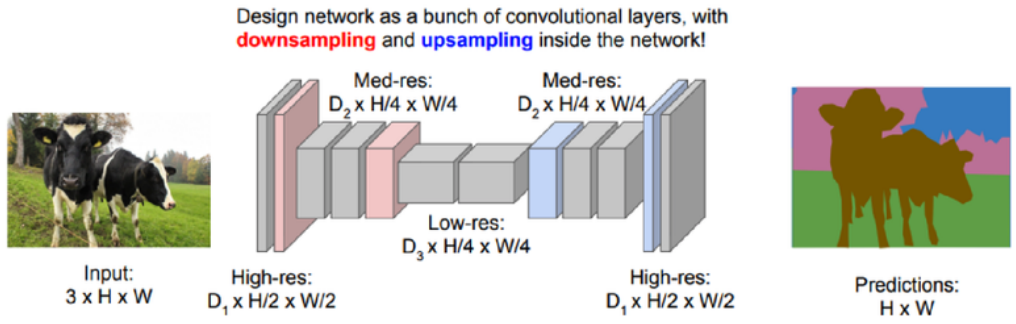

Figure 5: The Encoder/Decoder Archiecture of A CNN.

Typically, there are a few different approaches that we can use to upsample the resolution of a feature map. Whereas pooling operations downsample the resolution by summarizing a local area with a single value (ie. average or max pooling), "unpooling" operations upsample the resolution by distributing a single value into a higher resolution. Whereas a typical convolution operation will take the dot product of the values currently in the filter's view and produce a single value for the corresponding output position, a transpose convolution essentially does the opposite. For a transpose convolution, we take a single value from the low-resolution feature map and multiply all of the weights in our filter by this value, projecting those weighted values into the output feature map.

4 Soft Attention In Neural Networks

4.1 Introduction

For a neural network, attention gives the neural network the ability to focus on a specific subset of its inputs.

4.1.1 Soft Attention

Attention mechanisms compute a mask which is used to multiply features. This seemingly innocent extension has profound implications: suddenly, the space of functions that can be well approximated by a neural net is vastly expanded, making entirely new use-cases possible. Why? While I have no proof, the intuition is following: the theory says that neural networks are universal function approximators and can approximate an arbitrary function to arbitrary precision, but only in the limit of an infinite number of hidden units. In any practical setting, that is not the case: we are limited by the number of hidden units we can use. Consider the following example: we would like

to approximate the product of $N >> 0$ inputs. A feed-forward neural network can do it only by simulating multiplications with (many) additions (plus non-linearities), and thus it requires a lot of neural-network real estate. If we introduce multiplicative interactions, it becomes simple and compact.

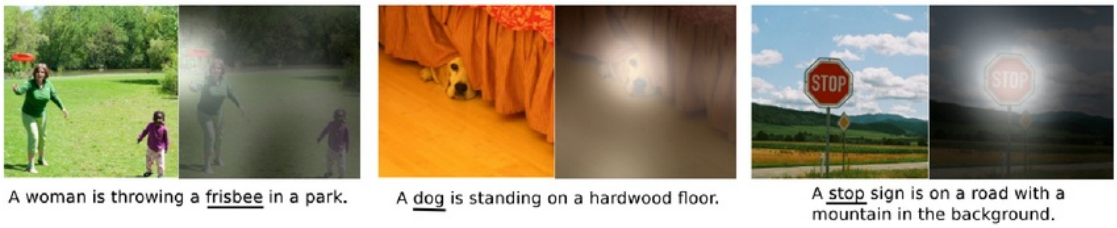

Figure 6: Examples of soft attention applied to different images.

5 Batch Normalization

5.1 Intoduction

Batch normalization is one of the methods used to speed up the rate at which a Deep Neural Network can be trained [9]. Before getting into the details of exactly how this process is done, a little background is required to understand why exactly traning DNN's used to be slow and how batch normalization hels in speeding up the process. Particularly, the problem of 'Internal Covariate Shift' has to be understood.

5.2 Internal Covariate Shift

Suppose we were to take an arbitrary d-layered neural network. To find the weights of the neurons in each layer of this network, we define an arbitray loss function L. As is evident from the section on neural networks, the gradient of the weights of a specific layer of neurons depends on it's input(which for any of the hidden layers is the output of the preceeding layer). Once this is done, the gradients are backpropagated and the weights are updated.

Now, suppose in this d-layered network, we were to select the i^{th} layer. This layer behaves no different from any other layer in the network in that it takes in some input and maps it to some output. However, it is here that the neural network runs into a problem. Unlike the first layer, for which the input data remains more or less constant, when the weights of the $i - 1^{th}$ layer changes, the input that the i^{th} layer receives from the previous layer changes. In effect, the distribution of the data recevied from the previous layer keeps on changing. This is known as inernal covariate shift and it slows down the training of a neural network.

5.3 Batch Normalization

To help with the problem of internal covariate shift, the process of batch normalization turns out to be useful. In essence, the batch normalization layer normalizes the output of each layer in the network along with a couple of exra operations.

To be more specific, it starts out by computing the mean and the variance of the input received by a particular layer by the following equations

$$\mu_B = \frac{1}{m}\sum_{1}^{m} x_i$$

$$\sigma^2 = \frac{1}{m}\sum_{i=}^{m}(x_i - \mu_B)^2$$

Then, the layer inputs are normalized using the values obtained from the above equations as shown. Here, x_i is subtracted by the mean and divided by the standard deviation to zero center the activations.

$$\bar{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Following this, the input that is to be received by a layer is scaled and shifted.

$$y_i = \gamma\bar{x_i} + \beta$$

It is perhaps this final step that is the most interesting. γ and β are known as the hyperparameters of batch normalization layer, where γ acts as the scaling factor while β acts as the shifting factor. When you ook closely, you notice that normalized input to the next layer has a mean of β and a standard deviation of γ.

While this may appear interesting, a good question to raise is, why does this actually work? As mentioned before, the primary issue with deep neural networs is that when the parameters of earlier parts of the network changes, the statistical distribution of the latter part of the network keep on shifting(internal covariate shift). This variation of distibution is primarily determined by the mean and the standard deviation of the data that the layer receives. Hence, by adding a batch normalization layer whose output statistics can be controlled by the hyperparameters β and γ, we can, to some extent, adjust the statistical distribution to minimize the effects of covariate shift.

6 Conditional Random Fields

6.1 Introduction

A variant of Markov models, Conditional Random Fields(CRFs) are a probabilistic framework fro labeling and segmenting structured data [10]. Unlike normal neural networks which take in a set of input and classify each based off of some model, CRFs take in a sequence of inputs and generate an output based off of the distribution of the inputs. More formally put, a conditional random field simply generates a probability distribution $p(y|x)$ with an associated graphical structure [11].

The mathematical background for Conditional Random Fields has quite a number of pre-requisites. Most of these are well beyond the scope of this report. However, a general understanding of these prerequisites [11] are recommended.

7 U-Net

7.1 Introduction

An improvement on the 'fully convolutional' architecture mentioned before, the U-net architecture consists of a contracting path to capture the context and a symmetric expanding path that enables

precise localization. It also offers a significant improvement in performance with $512 * 512$ pixel images taking less than a second to segment on a fairly recent GPUs.

7.2 Network Architecture

As can be seen in Figure 7, a U-net consists of a contracting path on the left and an expansive side on the right [12]. The contracting path on the left is identcal to that of a typical convolutional network. It consists of repeated 3×3 unpadded convolutions each followed by ReLU. Additionally, this is followed by a 2×2 max-pooling operation with a stride of 2 for downsampling where on each downsampling step, the number of feature channels are doubled [12]. Every repeating element of the expansive side consists of an up-sampling step followed by a 2×2 up-convolution. This also halves the number of feature channels. After this $3\times$ convolutions are performed. Each of these are followed by a ReLU unit. Finally, 1×1 convolution is used to map the component vector to the desired number of output classes.

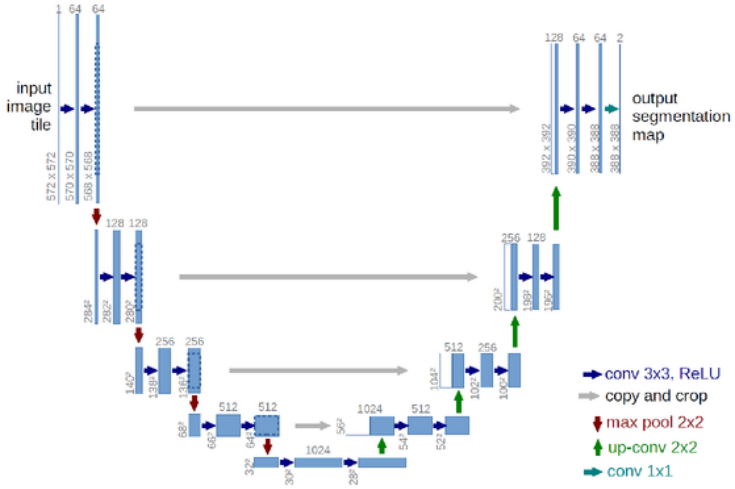

Figure 7: U-Net Architecture. [12]

7.3 Upsampling

In the previous section, the term 'upsampling' was mentioned. Here a basic idea of what this means along with the common approaches to upsampling will be discussed. Upsampling isa essentially a term associated with the interpolation of the pixels for the purpose improving the resolution of your output image. In a U-Net, upsampling is used to improve the resolution of the image across each layer in the expansive side of the network. While there a variety of upsampling methods, only three will be mentioned here:nearest neighbor, bilinear interpolation and transposed convolution.

7.3.1 Nearest Neighbor

Perhaps the simplest method for upsampling is through the use of Nearest Neighbor. Also known as proximal interpolation, in this method, when the image is scaled up, the empty sapces left behind are simply filled by the nearest pixel in the output image. While this method generates

sharp image, it also results in jagged edges in the images. A visualization of the same can be seen in Figure 8.

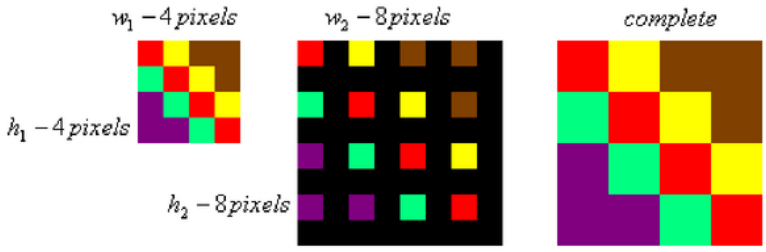

Figure 8: The functioning of Nearest Neighbor

7.4 Training

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe. Due to the unpadded convolutions, the output image is smaller than the inputby a constant border width. To minimize the overhead and make maximum use of the GPU memory, we favor large input tiles over a large batch size and hencereduce the batch to a single image. Accordingly we use a high momentum (0.99) such that a large number of the previously seen training samples determine the update in the current optimization step.

The energy is computer by a pixel wise soft-max over the final feature map combined with the cross entropy loss function. The soft max, defined as[12],

$$p(x) = \frac{exp(a_k(x))}{\sum_{k'=1}^{K} exp(a_{k'}(x))}$$

Here, $a_k(x)$ denotes the activation in the feature channel k at the pixel position $\mathbf{x}$, K denotes the number of classes and $p_k(x)$ is the approximated maximum function. Then, from the above the cross entropy the penalizes each position of the deviation of $p_{l(x)}(x)$ using the equation,

$$E = \sum_{x \in \Omega} w(x) log(p_{l(x)}(x))$$

where $l : \Omega \rightarrow \{1, ..., K\}$. Additionally, We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set, and to force the network to learn the small separation borders that we introduce between touching cells. The separation border is computed using morphological operations. The weight map is then computed as

$$w(x) = w_c(x) + w_0 \dot{e} xp\left(- \frac{((d_1(x) + d_2(x))^2}{2\sigma^2} \right)$$

References

[1] "Artificial Neural Networks as Models of Neural Information Processing — Frontiers Research Topic". Retrieved 2018-02-20.

[2] Zell, Andreas (1994). "chapter 5.2". Simulation Neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. ISBN 978-3-89319-554-1.

[3] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaaron (2016) Deep Learning. MIT Press. p. 196. ISBN 9780262035613

[4] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088): 533536. Bibcode:1986Natur.323..533R. doi:10.1038/323533a0.

[5] Krizhevsky, Alex & Sutskever, Ilya & E. Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[6] https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[7] Scherer, Dominik; Mller, Andreas C.; Behnke, Sven (2010). "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition". Artificial Neural Networks (ICANN), 20th International Conference on. Thessaloniki, Greece: Springer. pp. 92101.

[8] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). "ImageNet classification with deep convolutional neural networks" (PDF). Communications of the ACM. 60 (6): 8490. doi:10.1145/3065386. ISSN 0001-0782.

[9] Ioffe, Sergey; Szegedy, Christian. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"

[10] John Lafferty, Andrew McCallum, Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001), 2001.

[11]

[12] Olaf Ronneberger, Philip Fisher, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation.

CNN Report

ORIGINALITY REPORT

30%	26%	15%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.jeremyjordan.me Internet Source	9%
2	akosiorek.github.io Internet Source	5%
3	arxiv.org Internet Source	4%
4	Submitted to Thammasat University Student Paper	2%
5	en.wikipedia.org Internet Source	1%
6	"Medical Image Computing and Computer Assisted Intervention – MICCAI 2018", Springer Nature America, Inc, 2018 Publication	1%
7	Submitted to Xianjiatong-Liverpool University Student Paper	1%
8	Daniele Liciotti, Marina Paolanti, Rocco Pietrini, Emanuele Frontoni, Primo Zingaretti. "Convolutional Networks for Semantic Heads	1%

Segmentation using Top-View Depth Data in Crowded Environment", 2018 24th International Conference on Pattern Recognition (ICPR), 2018
Publication

9	link.springer.com Internet Source	1%
10	D. Roja Ramani, S. Siva Ranjani. "Chapter 9 U-Net Based Segmentation and Multiple Feature Extraction of Dermascopic Images for Efficient Diagnosis of Melanoma", Springer Nature, 2019 Publication	1%
11	Jun-Li Xu, Da-Wen Sun. "Computer Vision Detection of Salmon Muscle Gaping Using Convolutional Neural Network Features", Food Analytical Methods, 2017 Publication	1%
12	Submitted to Hong Kong Baptist University Student Paper	1%
13	Vatsala Singh, Ifeoma Nwogu. "Analyzing Skin Lesions in Dermoscopy Images Using Convolutional Neural Networks", 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018 Publication	1%

14	"Proceedings of 2018 Chinese Intelligent Systems Conference", Springer Nature America, Inc, 2019 Publication	1%
15	Submitted to Visvesvaraya Technological University Student Paper	<1%
16	Submitted to UT, Dallas Student Paper	<1%
17	www.tceic.com Internet Source	<1%
18	Tingyi Yan. "Chinese Semantic Role Labeling Based on Conditional Random Fields", 2009 Third International Symposium on Intelligent Information Technology Application Workshops, 11/2009 Publication	<1%
19	Submitted to Loughborough University Student Paper	<1%
20	"Intelligent Systems and Applications", Springer Nature America, Inc, 2019 Publication	<1%
21	en.m.wikipedia.org Internet Source	<1%
22	"Brainlesion: Glioma, Multiple Sclerosis, Stroke	

and Traumatic Brain Injuries", Springer Nature, 2019
Publication

<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		