Fully Convolutional Neural Networks for Newspaper Article Segmentation

Benjamin Meier
Report for module VT1, Master of Science in Engineering
Zurich University of Applied Sciences (ZHAW)

***Abstract*— Deep learning has triggered substantial improvements in computer vision. Especially convolutional neural networks (CNNs) were found to be very effective for e.g. classification tasks [1] [2] [3] [4] [5]. Fully convolutional networks (FCNs) [6] describe a way to modify a CNN to conduct semantic segmentation. The segmentation of newspaper articles is quite difficult, because the layout can be very complex and it varies for different newspapers. We show that FCNs perform very well for this problem. The proposed FCN-based network is trained end-to-end and compared to an existing solution [7]. It reaches better scores and requires less computation time.**

I. Introduction

Many documents, books and newspapers in large media archives are not yet digitized. If these archives would be available in digital form, they could be used for more detailed data analysis. It would be much easier to find information that is currently only available on printed media. Newspapers especially may contain interesting information; however, it is very difficult to digitize them, because many different page layouts are used. To detect different articles on a single page, it is required not only to do optical character recognition (OCR), but also to detect the article borders. Real-time print media monitoring would also profit from a solution with a high accuracy, because until the quality reaches a very high level, the article segmentation has to be done manually. High quality is required because customers who use the service pay much for highly accurate results.

An already existing solution is proposed by M. Arnold et al. [7]. They combine three different approaches to do the newspaper article segmentation. One is a rule-based method, another a text-based method, and the third method is image-based. The results of these three methods are then combined to a final article segmentation. This work focuses on the image-based approach of [7] and improves it. M. Arnold et al. use a neural network that has a similar architecture as the network proposed by D. C. Cireşan et al. [8]. For every pixel a surrounding window is created, which is used as input for the network. The output of the network is a classification for the pixel in the middle of the window. This classification process is done for each pixel of the input image. We were able to obtain the non-public dataset which is used by [7] for the training. It contains about $5,500$ pages of the largest newspapers from Switzerland. It is used to train our network and to compare it against [7].

Our proposed method does semantic segmentation [6] and uses a background and an article class for the pixel classification. Our network is based on the fully convolutional network (FCN) architecture that is proposed by J. Long et al. [6]. Compared to other approaches, like a patch-based method that classifies each pixel within a window around the pixel [8], the FCN architecture is more efficient in terms of computational time, because it requires only one execution of the network. Notably, it computes low-level filters only once. It is also able to use global information for local classification [6]. In contrast, the patch-based method is not able to use information outside the window for the classification task. A larger window size allows using more information of the context but also increases the computational costs.

A FCN has also the advantage that there are no densly connected layers, which gives the possibility to use input images with different sizes.

Related work will be discussed in Section II. After describing the network model in Section III and the required post-processing in Section IV, the training will be explained in Section V. Finally, Section VI contains some experiments that were done to compare our implementation with [7].

II. Related Work

Because of the large media archives that are not yet digitized, many previous works have focused on similar problems. Not only newspapers, but complete libraries contain not yet digitized material. Compared to printed books, newspaper pages often have a more complex layout, which also may vary for different newspapers and change over time.

T. Palfray et al. [9] focus on the challenge of digitizing old newspapers. Their approach not only does the segmentation but also finds the reading order. The used method works with an accuracy of 85.84% on old newspaper pages and reaches state-of-the art results. A particular conditional random field (CRF) model does the pixel classification. However, it is very specific and focuses only on old newspapers. This paper's solution focuses on new newspaper layouts and may also be used more widely.

B. Gatos et al. [10] propose a method to do article segmentation for newspapers. They first identify lines in the layout and then text and images. In a third step, headings are identified and finally a set of rules is used to recognize the articles. The method reaches a recall of 75.20% and a

precision of 77.15%. However, their rule set is much less dynamic than a neural network. This work's approach also has the advantage that an accurate OCR system is used to preprocess the data. This makes it much easier for the network to learn the general layout of an article.

A general approach is to use semantic segmentation for the newspaper page. Each pixel is classified and then the classification map is analyzed.

M. Arnold et al. [7] use a patch-based approach for the pixel classification. The input images are scaled to 100 pixels in height. The input image for the used neural network is already preprocessed by an OCR tool that labels the images and the texts. For the classification of each pixel, a window of the size 25×25 is used as input for a CNN, which does the classification itself. This CNN is therefore executed for each pixel of the input image. The results are often of good quality. However, compared to our method, it requires much more time for the execution of the neural network and we also reach a higher segmentation quality.

CNNs are often used for classification tasks, not only for 2D images [2] and 3D images [3], but also for sentences [4] as well as for audio data [5]. Y. LeCun et al. [1] already used CNNs for document recognition in 1998. H. Noh et al. [11] describe a general approach for semantic segmentation, which is based on a CNN with deconvolution [12] and unpooling layers. The advantage is that fine structures can be segmented very detailedly. For 2D image classification tasks, the proposed FCN architecture by J. Long et al. [6] allows extending a CNN architecture for semantic image segmentation instead of classification. Upsampling is done with a transposed convolution, which is also called deconvolution or backward-strided convolution [12].

A. Fakhry et al. [13] use a deconvolutional neural network for biological image segmentation. Their method uses unpooling layers and the network is highly designed to not lose any location information in the convolutional and pooling layers. This allows it to get a very detailed segmentation. Their network reaches state-of-the-art results, however, it is very memory intensive during training. The used Nvidia K80 graphics card with 24 GiB memory was only able to hold a batch size of 15. Our method requires less memory, which makes the training easier.

Another approach is to use Recurrent Neural Networks (RNN) for the semantic segmentation [14] [15]. The advantage is better use of the global information for local classification. However, the training is computationally expensive compared to FCNs.

In this paper, the used approach to address this problem is based on the network proposed by J . Long et al. [6]. The network is modified to reach highly accurate results for this use case. Compared to [13] it requires fewer resources for the training. As can be seen in the benchmark in Section VI-C, our method reaches a higher accuracy than [7] and the network is much faster in execution.

TABLE I: Network layers. `add`N layers are an element-wise addition to the given layer. The parameter p of the `dropout`N layers describes the probability of setting a value to 0.

Name	Kernel size	Stride	Pad	Output size
`input`	-	-	-	$256 \times 256 \times 2$
`dropout1` $(p = 0.3)$	-	-	-	$256 \times 256 \times 2$
`conv1-1`	5×5	1	2	$256 \times 256 \times 32$
`conv1-2`	3×3	1	1	$256 \times 256 \times 16$
`pool-1`	2×2	2	0	$128 \times 128 \times 16$
`dropout2` $(p = 0.3)$	-	-	-	$128 \times 128 \times 16$
`conv2-1`	5×5	1	2	$128 \times 128 \times 16$
`conv2-2`	3×3	1	1	$128 \times 128 \times 16$
`pool2`	2×2	2	0	$64 \times 64 \times 16$
`dropout3` $(p = 0.5)$	-	-	-	$64 \times 64 \times 16$
`conv3-1`	3×3	1	1	$64 \times 64 \times 16$
`conv3-2`	3×3	1	1	$64 \times 64 \times 16$
`pool3`	2×2	2	0	$32 \times 32 \times 16$
`dropout4` $(p = 0.5)$	-	-	-	$32 \times 32 \times 16$
`conv4-1`	3×3	1	1	$32 \times 32 \times 64$
`conv4-2`	3×3	1	1	$32 \times 32 \times 64$
`pool4`	2×2	2	0	$16 \times 16 \times 64$
`dropout5` $(p = 0.5)$	-	-	-	$16 \times 16 \times 64$
`conv5-1`	3×3	1	1	$16 \times 16 \times 64$
`conv5-2`	3×3	1	1	$16 \times 16 \times 128$
`pool-5`	2×2	2	0	$8 \times 8 \times 128$
`dropout6` $(p = 0.3)$	-	-	-	$8 \times 8 \times 128$
`conv6-1`	5×5	1	1	$8 \times 8 \times 128$
`conv6-2`	3×3	1	1	$8 \times 8 \times 256$
`pool-6`	2×2	2	0	$4 \times 4 \times 256$
`dropout7` $(p = 0.3)$	-	-	-	$4 \times 4 \times 256$
`conv7-1`	5×5	1	1	$4 \times 4 \times 256$
`transposed_conv8-1`	2×2	2	0	$8 \times 8 \times 128$
`add8` (layer = pool5)	-	-	-	$8 \times 8 \times 128$
`transposed_conv9-1`	2×2	2	0	$16 \times 16 \times 64$
`add9` (layer = pool4)	-	-	-	$16 \times 16 \times 64$
`transposed_conv10-1`	2×2	2	0	$32 \times 32 \times 16$
`add10` (layer = pool3)	-	-	-	$32 \times 32 \times 16$
`transposed_conv11-1`	4×4	4	0	$128 \times 128 \times 16$
`conv12-1`	5×5	1	2	$128 \times 128 \times 32$
`conv12-2`	5×5	1	2	$128 \times 128 \times 32$
`conv12-3-sigmoid`	1×1	1	0	$128 \times 128 \times 8$
`dropout12` $(p = 0.3)$	-	-	-	$128 \times 128 \times 8$
`conv12-4`	5×5	1	2	$128 \times 128 \times 32$
`conv12-5`	3×3	1	1	$128 \times 128 \times 16$
`conv13-1-sigmoid`	1×1	1	1	$128 \times 128 \times 1$
`upscale13` (factor = 2)	-	-	-	$256 \times 256 \times 1$
`output`	-	-	-	$256 \times 256 \times 1$

III. Model

The network model is based on the fully convolutional network approach that is proposed by J. Long et al. [6]. The complete network is shown in Figure (1) and each layer is described in Table I. In the beginning the network contains several convolutional and max-pooling layers. There are no densely connected layers in the complete model. The network is built with three logical parts. Initially, there is a feature extraction part that does the convolutions and the max-pooling. This part is a standard CNN. The second part of the network is used for a trained upscaling and the segmentation. Finally, a very small refinement network gives the chance to correct some artifacts. This is mainly useful because the expected output usually only contains rectangular objects. Sometimes it is required to refine the segmentation output to fulfill this requirement. This makes it much easier to post-process the output. The training time decreases much with the refinement layers. At the end, a sigmoid layer is used to do a binary classification.

According to Table I, the feature extraction part contains the layers from `input` up to `conf7-1`. The upscaling network then contains the layers from

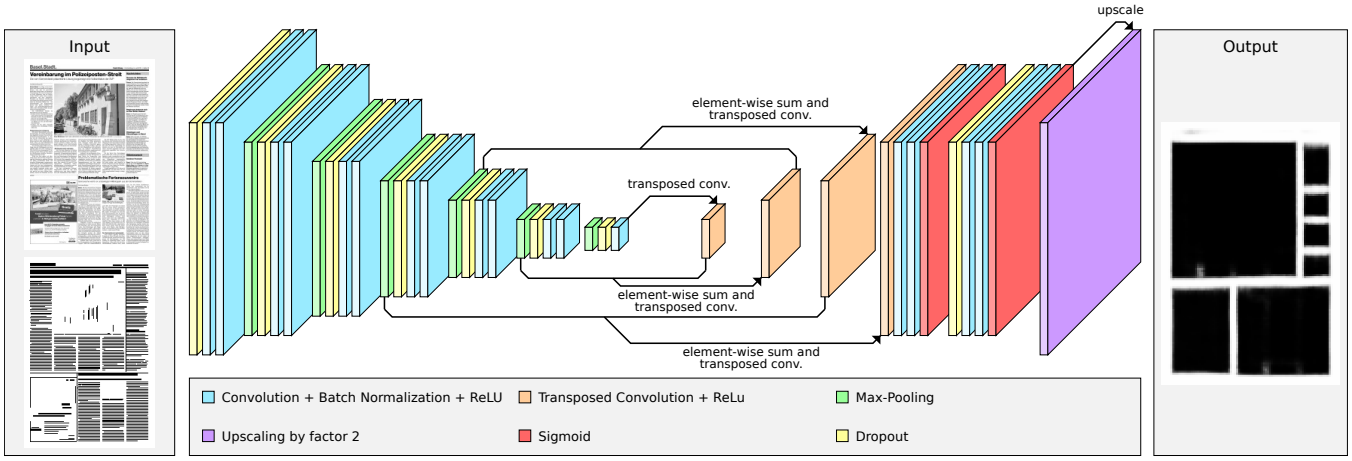
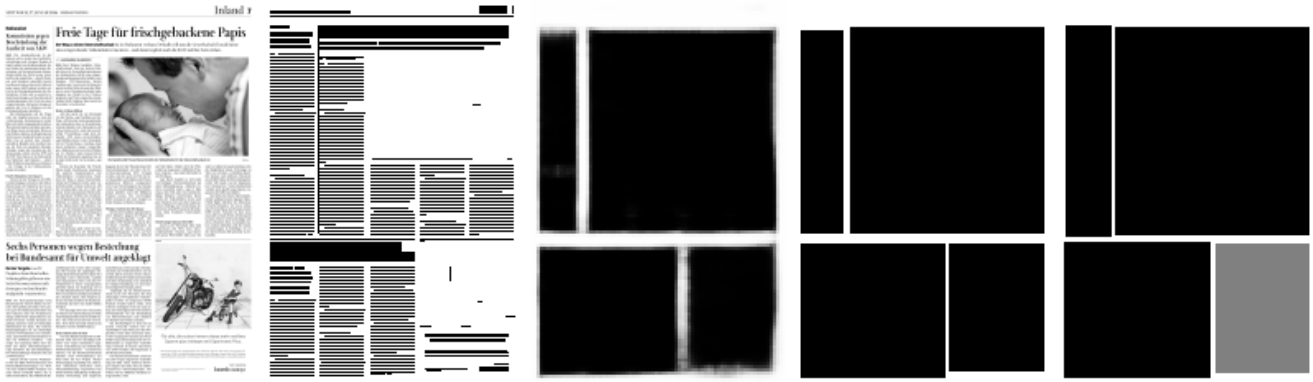


Fig. 1: The network architecture.



(a) Input: Plain page (b) Input: OCR page (c) Network: Output (d) Output: Polygons (e) Labels

Fig. 2: (a) and (b) show the input for the used network. Image (c) contains the plain output of the network and (d) the detected polygons. (e) shows where the labels actually are positioned.

`transposed_conv8-1` up to `transposed_conv11-1`. Finally, the refinement part includes all layers from `conv12-1` up to `conv12-5`.

The network input has a shape of $256 \times 256 \times 2$ pixels. Newspaper pages are resized with respect to the aspect ratio to fit the 256×256 pixel format. The default background is white. The input size is chosen to maximize the model's quality. It is found that larger inputs do no longer increase the segmentation quality. The model uses two grayscale input images, which are scaled from $[0, 255]$ to the range $[0, 1]$. One input image is the plain newspaper page and the other a preprocessed and OCR scanned version. The network inputs are shown in Figures (2a) and (2b). The OCR input describes where text was found, where images were detected, and it also contains horizontal and vertical lines that are in the plain newspaper page. During testing it could be seen that the best combination is to use both inputs. The reason is because the OCR system does not always work well, but it already gives the network great hints where something interesting could be. Most times the OCR system produces a reasonable output.

Except for two sigmoid layers, the network uses only rectified linear units (ReLU) [16] for the nonlinearities. As can be seen in Table I, the first initial convolution and the convolution after the first max-pooling layer use a kernel size of 5×5, and all other convolutions in the feature extraction layers use a kernel size of 3×3. Each max-pooling layer uses a pool-size of 2×2. The kth pooling layer is called `pool`k, and such max-pooling layers exist from `pool1` up to `pool5`. The feature count of the convolutions is increased during the feature extraction network. The first convolutional layer uses a feature count of 32 and the last one a count of 256. All parameters for all layers are shown in Table I.

As can be seen in Table I, the layer shape before the feature extraction network is $\mathbf{256 \times 256} \times 2$ and thereafter it is $\mathbf{4 \times 4} \times 256$. The upscaling network uses transposed convolutions [12] to train how the different layers should be upscaled. The initial input is the output of the feature extraction network. It is upscaled from $\mathbf{4 \times 4} \times 256$ to $\mathbf{8 \times 8} \times 128$. The feature count is equal to the feature count of `pool5`. This new output and the output of `pool5` are added together element-wise. This summed-up output is then upscaled to $\mathbf{16 \times 16} \times 64$ with a transposed convolution and

Algorithm 1: An algorithm to extract rectangular polygons from a binary map.

function ExtractPolygons (binaryMap);
Input : A binary map binaryMap
Output: A set of rectangular polygons
$L \leftarrow$HoughTransform(binaryMap)
$L \leftarrow$filter(L,only allow lines which go completely trough the image and have a degree of 0 or $\frac{\pi}{2}$)
for line $\in L$ **do**
  Draw line to binaryMap with a white color
end
$P \leftarrow$Detect black objects in binaryMap and get all their rectangular bounding boxes as a polygon
if $|P| = 1$ **then**
  $\text{out}_{\text{polygons}} \leftarrow B$
else
  $\text{out}_{\text{polygons}} \leftarrow \emptyset$
  for $p \in P$ **do**
    $\text{out}_{\text{polygons}} \leftarrow \text{out}_{\text{polygons}} \cup$ ExtractPolygons(get pixels of P in binaryMap as binary map)
  end
end
return $\text{out}_{\text{polygons}}$

Algorithm 2: An algorithm to refine polygons in a way that they match better to some OCR elements.

function RefinePolygons (polygons, ocr);
Input : A set of polygons polygons and a set of OCR elements / polygons ocr
Output: A set of refined polygons
tmp $\leftarrow$ polygons
for $i \in \{0, 1\}$ **do**
  $\text{increased}_{\text{polygons}} \leftarrow \emptyset$
  for $p \in$ polygons **do**
    if p has a rectangular shape **then**
      Increase p on every side by 1 pixel unless it touches another $q \in$ polygons
    end
    $\text{increased}_{\text{polygons}} \leftarrow \text{increased}_{\text{polygons}} \cup \{p\}$
  end
  tmp $\leftarrow \text{increased}_{\text{polygons}}$
end
$\text{out}_{\text{polygons}} \leftarrow \emptyset$
for $p \in \text{increased}_{\text{polygons}}$ **do**
  if there is at least 1 OCR element of ocr inside p **then**
    Decrease p on all 4 sides until a OCR element pixel would leave p. Stop then.
  end
  $\text{out}_{\text{polygons}} \leftarrow \text{out}_{\text{polygons}} \cup \{p\}$
end
return $\text{out}_{\text{polygons}}$

uses the same feature count as `pool4`. Again the output is added element-wise to the output of `pool4` to create a new representation of the data. This new layer is then again upscaled to $\mathbf{32 \times 32} \times 16$ with the same feature count as `pool3` and added to the output of `pool3`. The new layer uses a feature count of 16. A last time, the new layer is upscaled by a factor of 4, which creates an output with the shape $\mathbf{128 \times 128} \times 16$. As can be seen, the output shape of the upscaling network is not equal to the input shape of the initial network that would be required for a semantic segmentation. The reason for this is that a higher resolution is not required for the segmentation, because the newspaper articles are usually very coarse and often rectangular. These facts allow it to work internally with a lower resolution and still get high accurate results.

The refinement network improves the output of the upscaling network. Without the refinement network, the fine details in the output do not always look good. It can happen that borders are not straight or that there are some holes inside the found articles. This small additional logical network improves the output and decreases the required training time. It contains two convolutions with a filter size of 5×5 and 32 features, then a convolution with the size 1×1, which uses a sigmoid nonlinearity with 8 features. This sigmoid layer is used as a bottleneck. It forces the network to compress the information and to pre-classify it. Then there is another convolution with a filter size of 5×5 and 32 features. The last layer of the refinement network is a convolution with a filter size of 3×3 and 16 features. The input and the output of the refinement network have the shape $128 \times 128 \times 16$.

Finally, there is a layer that does the classification. It is a sigmoid convolution with the filter size 1×1 and 1 feature. Sigmoid is used instead of a softmax layer because there are only two classes. After this layer, an upscaling layer scales the x and y axis by a factor of 2. This is done to create a network output with the same width and height as the network input. Next, the output classifies each pixel of the input and has therefore a shape of $256 \times 256 \times 1$. The network produces values near to 1 if a pixel is classified as background and values near to 0 if the pixel is classified as foreground. It can return any value in $(0, 1)$, which is the reason why an additional function has to be used to binarize the output. Different threshold values were tested, and it was determined that it works the best if all values ≥ 0.35 are rounded up to 1 and all values ≤ 0.35 are rounded down to 0.

In general, the model may be used for input images with the size $64n \times 64m$ with $n, m \in \mathbb{N}_{>0}$, because the complete network only contains max-pooling and convolutional layers.

IV. Post-Processing

The output data of the model is a classification map that classifies each pixel of the input image. This map must be post-processed to generate a polygon for each article. It is known that articles generally have a rectangular shape. The

binarized network output can be post-processed to reach a higher accuracy. For this reason, the rectangular objects are searched with the help of the Hough-transform. The algorithm is described in Algorithm 1. Its output is a set of polygons with a rectangular shape.

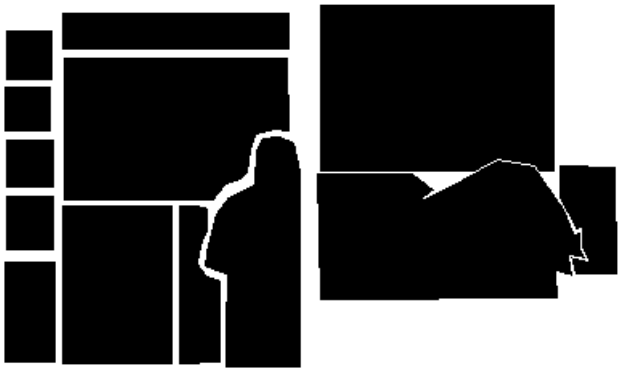
As described in Section V, the network is trained to find the article labels that are decreased by 2 pixels. Because of this, the found article polygons must be increased. Sometimes just increasing the labels in all directions does not result in a very good solution, as it is likely that the article borders match well with OCR blocks. Because of this, there is a simple refinement algorithm that tries to improve the detected articles polygons locally. It is described in Algorithm 2. Its input is a set of article polygons and OCR element polygons and the output is a set of refined article polygons.

V. Training

The used dataset from the company ARGUS der Presse AG contains about $5,500$ newspaper pages. All pages are from the largest Swiss newspapers. They all contain some labels, but unfortunately only a very few pages are fully labeled. There are two types of labels. The black label type is used for articles and the gray label type for advertisements. The two different labels can be seen in Figure (2). Almost all labels have a rectangular shape, but some have much more difficult shapes, as may be seen in Figure (3). For this work, all advertisement labels are converted to article labels, because it is not the main objective of this work to differentiate between real articles and advertisements.

Newspaper pages that are not fully labeled must be preprocessed. The non-labeled content would confuse the network, which is the reason why it should be removed. The network would not be able to train what an article is and what is background unless all articles are labeled. The page content that is more than 3 pixels away from every article label is changed to white. This is done for the pixels in the plain input as well as in the OCR input. The 3 pixel border is very important, because the neural network must learn how the borders of the articles look like. Often there are some straight lines between the articles, which are very helpful to detect the border between articles. On the other hand, these 3 pixel borders add some noise, because they often contain some parts of other articles or images. The neural network learns to ignore these additional parts, which is not always correct. Different approaches were tested, and shrinking the labels works much better than the other ways do. Other tested methods are to just use the fully labeled pages for the training or also to not shrink the labels but remove everything outside of them.

As described, some labels do not have a rectangular shape. Unfortunately, sometimes for very similar input pages the labels are rectangular and sometimes not. Some other pages are even wrongly labeled. Because of this, the dataset is filtered. All images that are wrong labeled or that have highly non-rectangular shapes, like the images that are shown in Figure (3), were searched. The found image records are removed from the training set. The resulting new dataset improves the quality of the neural network. It contains about $4,135$ pages.



(a) Non-rectangular labels. (b) Non-rectangular labels.

Fig. 3: (a) and (b) show some labels that do not have a rectangular shape.

The labels used for the articles often have borders that have a width of just 1 pixel, which is not optimal for the training. Because of this, all labels are decreased by 2 pixels, which makes the minimum border size equal to 5 pixels. This makes the training and especially the post-processing much easier.

The network uses an input shape of $256 \times 256 \times 2$ as described in Section III. For the training, the downscaled input is placed at a random position. This is done to increase the input variety. The newspaper pages are also randomly mirrored in the y axis with a probability of $p = 0.5$. These two methods allow it to train the network better with the limited available data.

The used binary cross-entropy loss function uses weights for wrong classifications. The borders between the articles are the most important part, but often they are small. This makes it extremely important to classify them correctly. Nevertheless, it is not a big problem if there are some wrongly classified pixels inside an article. Pixels that are classified as article instead of as background get an error factor of 1.8 and all other classification errors get an error factor of 1.0.

To make the training more efficient and to avoid overfitting, the network uses much dropout [17]. This can be seen in Figure (1) and in Table I. Dropout is used directly after the input to add some noise. The network also uses L_2 regularization with a weight decay of 0.0001.

Due to the internal covariate shift problem, batch normalization is used [18]. The batch normalization is performed for every convolutional layer, which allows the network to converge to a better optimum.

For the optimization, the Nesterov momentum is used with a learning rate of 0.01. The batch size is 16.

P. Luc et al. [19] proposed a network for semantic seg-

(a) Benchmark input: Image 1 (b) Benchmark input: Image 2 (c) Benchmark input: Image 3 (d) Benchmark input: Image 4

(e) Benchmark input: Image 5 (f) Benchmark input: Image 6 (g) Benchmark input: Image 7 (h) Benchmark input: Image 8

Fig. 4: Different input images that are used for the benchmark. The green area shows the correctly classified article pixels, the blue area the non-detected article pixels and the red area the pixels that are classified as article but are background.

mentation that uses an adversarial network for regularization. This regularization approach was also evaluated to see if the quality of the network improves. The training time increased because also the discriminator network has to be trained, but on the other hand there was no improvement in quality. For this reason the proposed regularization by P. Luc et al. [19] is not used in the final implementation.

VI. Experiments

The experimental setup uses a Geforce GTX 780 to train the network. The used computer has 128 GiB of RAM and an Intel XEON E5-2620 CPU, but the training as well as the classification processes do not require much memory. The filtered dataset from ARGUS der Presse AG contains $4,135$ labeled pages, but only 406 pages are fully labeled. The training is done in two stages. First, the training uses the full training set and all input images are preprocessed as described in Section V. This is done for 210 epochs. After these epochs, the dataset is limited to only the fully labeled pages. There are only 406 fully labeled pages, which makes this training step harder. Unlabeled content is no longer removed as described in Section V. The neural network now has the possibility to learn that some newspapers use their name as a large title on the top of the page. This title is not part of any article and should be ignored for the segmentation process. The training on the fully labeled data is done for 150 epochs. After these two stages, the training for the given dataset is finished.

During the training, the F_1 score is calculated for all pixels. This score already gives a hint if the training works well or not. Because it is pixel-based it is good locally, but for the global view it might not be optimal. For this reason two other scores are defined that work on the output of the post-processed data.

A. Diarization Error Rate Score

The diarization error rate (DER) score is known from speaker diarization and was adopted to the newspaper article segmentation problem by M. Arnold et al. [7]. It uses three error types. The miss error e_{miss} counts the area of the articles that were not found by the system, $e_{\text{falsePositive}}$ counts the area that is classified as an article but which should be classified as background and, finally, $e_{\text{confusion}}$ counts the area of wrongly associated articles. Each detected article is

TABLE II: Detailed benchmark results for 8 images. The best results are in **bold**.

Implementation	Input image	DER score	Completeness score	Classification time $[s]$
New approach	Benchmark image 1	**0.1888**	**0.3000**	**0.0550**
PANOPTES	Benchmark image 1	0.7244	0.0000	1.8310
New approach	Benchmark image 2	**0.0183**	**1.0000**	**0.0460**
PANOPTES	Benchmark image 2	0.0986	**1.0000**	1.8730
New approach	Benchmark image 3	**0.0236**	**0.7500**	**0.0440**
PANOPTES	Benchmark image 3	0.3729	0.5000	1.5130
New approach	Benchmark image 4	**0.0744**	**0.8750**	**0.0470**
PANOPTES	Benchmark image 4	0.5573	0.1250	1.6150
New approach	Benchmark image 5	**0.2249**	**0.5000**	**0.0530**
PANOPTES	Benchmark image 5	0.5328	0.0000	1.6070
New approach	Benchmark image 6	0.2388	**0.4000**	**0.0440**
PANOPTES	Benchmark image 6	**0.2203**	**0.4000**	1.5260
New approach	Benchmark image 7	**0.0338**	**1.0000**	**0.0470**
PANOPTES	Benchmark image 7	0.4158	0.0000	1.6700
New approach	Benchmark image 8	0.2550	0.4000	**0.0440**
PANOPTES	Benchmark image 8	**0.1150**	**0.6000**	1.5140

associated with the most overlapping label if there is at least one overlapping label. All other detected articles that also overlap with this label are counted for the overlapping area into $e_{\text{confusion}}$.

The DER score sums up all these errors and is normalized by λ, which is equal to the total area of all reference labels on the page. The formula for the DER score is shown in Equation (1).

$$S_{\text{DER}} = \frac{e_{\text{confusion}} + e_{\text{miss}} + e_{\text{falsePositive}}}{\lambda} \tag{1}$$

The lower the score, the better is the result. The score is always in the interval $[0, \infty)$, which means for 0 the result is perfect.

B. Completeness Score

An intuitive explanation for the completeness score is that is defined as the articles that were found correct divided by the total number of articles. This means, the score value is always a rational number in the interval $[0, 1]$, where 0 is the worst and 1 is the best value. This score has a very low tolerance on matching found articles with labels and therefore requires a high segmentation quality.

The OCR input shown in Figure (2) contains several small non-rectangular blocks b_k. These blocks represent text or an image. The set B contains all blocks. Every labeled reference page P contains some labels l_i. At least one label must be in P. If a page does not contain any blocks or labels, the completeness score is always 0.

$$B = \{b_0, b_1, ...\}, |B| > 0 \tag{2}$$

$$P = \{l_0, l_1, ...\}, |P| > 0 \tag{3}$$

For each reference label l_i, there exist some OCR blocks b_k that are completely inside the label l_i. This set is called $\mathfrak{B}[l_i]$. Labels must not overlap, which means they never share the same blocks.

$$\mathfrak{B}[l_i] \subseteq B \tag{4}$$

$$\mathfrak{B}[l_i] = \{b_k \in B | b_k \text{ is inside } l_i\} \tag{5}$$

$$\forall l_i, l_j : i \neq j \Rightarrow \mathfrak{B}[l_i] \cap \mathfrak{B}[l_j] = \emptyset \tag{6}$$

TABLE III: Benchmark results summary for over 81 images. σ describes the standard deviation. The best results are in **bold**.

	Our approach	PANOPTES
Avg. DER score	**0.1378**	0.2976
Avg. Completeness score	**0.5444**	0.2079
Min. DER score	0.0051	**0.0000**
Max. DER score	**0.5920**	1.0028
Min. Completeness score	**0.0000**	**0.0000**
Max. Completeness score	**1.0000**	**1.0000**
DER score σ	0.1343	0.2265
Completeness score σ	0.3464	0.3011
Avg. classification time [s]	**0.0476**	1.6553

The segmentation S of the used system contains some article polygons a_i. $\mathfrak{B}[a_i]$ contains all OCR blocks that are completely inside the article polygon a_i. Article polygons never overlap.

$$S = \{a_0, a_1, ...\} \tag{7}$$

$$\mathfrak{B}[a_i] \subseteq B \tag{8}$$

$$\mathfrak{B}[a_i] = \{b_k \in B | b_k \text{ is inside } a_i\} \tag{9}$$

$$\forall a_i, a_j : i \neq j \Rightarrow \mathfrak{B}[a_i] \cap \mathfrak{B}[a_j] = \emptyset \tag{10}$$

The completeness score counts the labels and the article polygons that contain exactly the same blocks. This number is divided by the total count of labels. The formula for calculating the completeness score is shown in Equation (11).

$$S_{\text{COMPLETENESS}} = \frac{|\{(a, l) \in S \times P | \mathfrak{B}[a] = \mathfrak{B}[l]\}|}{|P|} \tag{11}$$

C. Benchmark

The new proposed method is compared with the previous implementation of PANOPTES [7]. The required classification time of the network as well as the quality of the segmentation are compared. As described in Section I, there are three approaches in PANOPTES [7], which are combined together to create a final segmentation. The new method only

replaces the visual approach, therefore the tests focus on this approach. This system as well as PANOPTES are just a small part of a larger software system. For this reason, only the classification time of the neural network is measured.

The required training time our system is around 5 h, whereas PANOPTES [7] only requires 1 h 15 min of training. As can be seen, [7] requires much less training time. Their used model has $252,706$ parameters, whereas our model has $1,435,065$ parameters, which might be the reason for the longer training time. However, our model computes the complete classification of an image in an average of about 47.6 milliseconds and [7], having to execute the network for each pixel, therefore requires about $1,655$ milliseconds in average for the total classification. This is shown in Table III. As a conclusion, our model requires more training time, but then has a much lower runtime. Both implementations use Algorithm 1 to compute the polygons. This algorithm is relatively slow and decreases the relative runtime difference between the two models.

Both implementations resize the input image before the neural network processes it. This makes the effective runtime almost constant. It would be possible for both implementations to process larger input images, but this does not improve the quality.

To compare the results of [7] and the new implementation, the scores that are defined in Sections VI-A and VI-B are used. Figure (4) shows some images that are used as input for the benchmark. Green colored areas show which pixels are classified correctly. The blue pixels are false negative, which means they are classified as background, but they are articles. Finally, the red pixels are classified as article pixels but really are background pixels. The scores for these images are listed in Table II. A larger benchmark was done for over 81 fully-labeled images for which a summary of all results is shown in Table III. As can be seen, the new solution often outperforms [7].

VII. Discussion

This paper presented an FCN-based approach to learn newspaper article segmentation. To demonstrate the improvements with this architecture, it was compared with PANOPTES [7], which is based on the approach proposed by A. Fakhry et al. [13]. Future projects might differ between the found article types. E.g., the network could use 3 output classes, which differentiate between articles, advertisements and background. Another method to reach this objective could be to use a second neural network, which is only responsible for classifying a single article as advertisement or as a real article.

J. A. Montoya-Zegarra et al. [20] proposed a method for multi-class semantic segmentation of urban areas. They reach a high accuracy. The used method could be modified and used for newspaper article segmentation.

VIII. Conclusions

FCNs are very efficient for semantic segmentation tasks. This paper shows that the segmentation of newspaper articles works very well and also very efficiently with FCNs.

FCNs are not only faster in execution than patch-based approaches, but also reach the same or higher segmentation quality. FCNs therefore can use larger input images and analyze the input in more detail with the same computational time, which can also increase the quality of the output.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," *arXiv preprint arXiv:1604.03265*, 2016.

[4] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[5] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[7] M. Arnold, M. Cieliebak, T. Stadelmann, J. Stampfli, and F. Uzdilli, "Panoptes–automated article segmentation of newspaper pages for."

[8] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[9] T. Palfray, D. Hebert, S. Nicolas, P. Tranouez, and T. Paquet, "Logical segmentation for article extraction in digitized old newspapers," in *Proceedings of the 2012 ACM symposium on Document engineering*. ACM, 2012, pp. 129–132.

[10] B. Gatos, S. Mantzaris, K. Chandrinos, A. Tsigris, and S. J. Perantonis, "Integrated algorithms for newspaper page decomposition and article tracking," in *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*. IEEE, 1999, pp. 559–562.

[11] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[12] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

[13] A. Fakhry, T. Zeng, and S. Ji, "Residual deconvolutional networks for brain electron microscopy image segmentation," *IEEE Transactions on Medical Imaging*.

[14] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, "Scene labeling with lstm recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3547–3555.

[15] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling." in *ICML*, 2014, pp. 82–90.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[17] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[19] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *arXiv preprint arXiv:1611.08408*, 2016.

[20] J. Montoya-Zegarra, J. Wegner, L. Ladickỳ, and K. Schindler, "Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, p. 127, 2015.