**BRAIN. Broad Research in Artificial Intelligence and Neuroscience**
ISSN: 2068-0473 | e-ISSN: 2067-3957
Covered in: Web of Science (WOS); PubMed.gov; IndexCopernicus; The Linguist List; Google Academic; Ulrichs; getCITED; Genamics JournalSeek; J-Gate; SHERPA/RoMEO; Dayang Journal System; Public Knowledge Project; BIUM; NewJour; ArticleReach Direct; Link+; CSB; CiteSeerX; Socolar; KVK; WorldCat; CrossRef; Ideas RePeC; Econpapers; Socionet.

2023, Volume 14, Issue 2, pages: 93-114 | https://doi.org/10.18662/brain/14.2/446
Submitted: June 6th, 2023 | *Accepted for publication:* June 28th, 2023

# Document Image Binarization Process

**Marcel PRODAN** [1],
**Costin-Anton BOIANGIU** [2]

[1] PhD student, Eng., University Politehnica of Bucharest, 060042 Bucharest, Romania, marcel.prodan@stud.acs.pub.ro
[2] Professor, PhD Eng., University Politehnica University of Bucharest, 060042 Bucharest, Romania, costin.boiangiu@cs.pub.ro

**Abstract**: *Technology has made significant strides in recent years, which accounts for how pervasive it is in our daily lives. In order to address the fundamental issue with historical document preservation, namely their degeneration, this work suggests using new technology. The method is built on pieces of artificial intelligence that can read the writing from a page and recognize the useful information, converting it into a digital version. Contrary to photographing or scanning, binarizing a document is a considerably more effective method, both in terms of quality—the legibility of the writing—and quantity—the amount of memory needed to retain the resulting image. According to common assessment measures, the suggested fully convolutional network manages to deliver results that are comparable to those of other solutions of a similar nature.*

**Keywords:** *image binarization, artificial intelligence, convolutional neural network, deep neural network, fully convolutional network.*

**How to cite:** Prodan, M., Boiangiu, C.-A. (2023). Document Image Binarization Process. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 14*(2), 93-114. https://doi.org/10.18662/brain/14.2/446

## 1. Introduction

Historical records offer a rich and extensive source of every civilization's culture and tradition. Regardless of the preservation settings, time has left its mark on them through various forms of degradation, which tend to significantly decrease the text's readability. Binarization is thus a useful technique for qualitatively and quantitatively digitizing these documents. Binarization separates the pixels that make up the document's useful information from the pixels that cause its different visual issues.

The challenge this project seeks to address is how to effectively preserve handwritten or printed historical documents in various degrees of degradation by converting them into a digital version that only includes the pertinent portions of the original document.

In order to achieve the most accurate binarization, the current research seeks to describe the many existing binarization approaches as well as to suggest a new architecture inspired by cutting-edge solutions based on fully convolutional networks.

The infrastructures for training, testing, and binarization are all included in the suggested solution, which consists of a fully convolutional network. The DIBCO data sets (n.d.), which are frequently cited in papers proposing different binarization algorithms, were used to train the final model (Patchify, n.d.).

The proposed network is simpler from an architectural standpoint than other comparable methods used in the field of semantic segmentation, yet it nonetheless achieves a respectable result when the trained model's performance is assessed using accepted metrics.

## 2. Related work

### *Classic Thresholding algorithms*

Otsu (1979) is one of the well-known algorithms for locating a global threshold value. Depending on where the pixel intensity value falls concerning the algorithm's threshold intensity value, it seeks to divide the image's pixels into two distinct classes. The threshold value is derived taking into account the maximum variance between the two classes and the minimization of variance within a class. When given an image with sections of uneven lighting, i.e., areas where the intensity of the foreground pixels coincides with the intensity of the background pixels in other areas, a weakness in this algorithm is brought to light. As a result, the algorithm does

not work in these typical circumstances and does not offer a workable answer.

Niblack (1985) is a local and adaptive threshold value detection technique. Given that the method determines a unique threshold value for each pixel in the image depending on the intensities of its nearby pixels, the threshold value fluctuates dynamically over the image surface. The mean and standard deviation of the pixel intensities of the window with the size n×n, with the n parameter imposed, are used to compute the threshold of a given pixel. The window size we select affects the quality of the binarization result because, as n increases, the illumination gradient's influence on the threshold value increases, lowering segmentation accuracy. In addition, the algorithm imposes a parameter k that the user can change to alter the effect that the standard deviation has on the threshold value's ultimate value. k regulates the trade-off between the accuracy and comprehensiveness of foreground pixel recognition. The advantage of this algorithm is that, in contrast to Otsu, it is able to produce a binarization under circumstances where the light gradient is significant. When a window solely displays background pixels, this algorithm's flaw is evident because certain pixels are incorrectly classified.

Another local and adaptive threshold detection technique is Sauvola & Pietikäinen (2000). It suggests a fix for the Niblack algorithm's issue that arises when the window used for statistical calculations to determine the threshold value only contains background pixels. In such cases, the algorithm's objective is to reduce the number of pixels classified as foreground pixels. The algorithm's ability is to calculate the threshold while taking into account the ratio of the local standard deviation's value corresponding to the maximum standard deviation possible. As a result, the detected threshold value will be smaller if the local standard deviation is significantly less than the maximum standard deviation, if the window contains many pixels of the same type, and larger if the local standard deviation approaches the maximum possible value of the standard deviation, if the window contains a noticeable diversity of pixel types.

### *Solutions based on Shallow Models*

The remedy suggested by Hamza et al. (2005) is made up of an architecture whose individual components are based on the ideas of self-organizing maps (SOM), k-means, and multilayer perceptrons (MLP). Binarization of photos in color format is the intended outcome. The SOM, which gets a three-dimensional vector representing each of the three color channels—red, green, and blue—as input, kicks off the process. The SOM is

trained on sections of the image rather than the entire image. After training the SOM, the k-means algorithm further processes the weights connected to its neurons in order to classify the data.

The labeled weights are utilized to train the MLP after categorization. The decision to use an MLP is supported by the possibility that some color tones may overlap, making them impossible to linearly separate. The MLP then receives the complete image as input and classifies each pixel based on the prior training. In the end, two classes of pixels will be combined into two images, one of which will be the opposite of the other. Knowing the type of image provided as input allows one to select the appropriate one and determine which of the two groups of pixels represents the foreground. Documents will typically have more background pixels than foreground pixels, so the image with the highest proportion of white pixels will be taken into consideration. It has been demonstrated that this technique outperforms traditional algorithms like Otsu and Sauvola in maintaining the final image-specific fine details from the original image, such as varied ornamental components or variations in font thickness. This technique, however, is not without flaws, one of which is the slow processing of large photos. The approach delivers a result for small photos in roughly the same amount of time as traditional techniques.

The approach presented in the work by Kasmin et al. (2017) is based on a series of 8 Support Vector Machines (SVMs), each of which corresponds to a Steerable Local Neighborhood (SLN) type filter and is rotated at a certain angle among the 8 possible orientations: 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. A 3×3 pixel window that is centered on a T pixel, also known as a hotspot pixel, is represented by each SLN. Each of these SLNs contains the neighborhood's grayscale intensities and seeks to create a feature set for the hotspot pixel. Such sets were created based on 5 training photos for each SLN orientation. The sets were then used as input for SVMs to build a model for every SLN orientation. Each of the 8 SLNs was given a weight based on accuracy using the Ground Truth photos. An SVM will produce two probabilities of belonging to the two classes, foreground and background, for each pixel of an input image. In this manner, two sets of probabilities—one set belonging to the foreground and one set belonging to the background—will be constructed for a given pixel, each set including an estimated probability for each direction of the SLN. The authors will create two binarizations based on the weighted product rule and the weighted addition rule using the two sets of probabilities and the weights assigned to each SLN. The two rules' function is to assign a score to every pixel in order to categorize it. Based on a comparison with the Ground

Truth, the best image will be picked. The accuracy of the two categorization rules varied based on the type of image content, it was discovered. For instance, the weighted addition rule offered better document image binarization accuracy (0.9705 ±0.0171), whereas the weighted product rule offered better retina image binarization accuracy (0.9328 ±0.0121). The proposed approach to document binarization outperforms both the strategy based on a single SLN (whose accuracy was 0.9614 ±0.0220) and those utilizing classical algorithms in terms of accuracy.

***Solutions based on Deep Neural Networks***

From neural networks that represented the state-of-the-art in terms of image classification to networks of completely convolutional semantic segmentation, Long et al.'s work (2015) presents a solution based on Fully Convolutional Networks (FCNs). The last classification layer was removed, and each fully connected layer was converted into a convolutional layer in the authors' initial modifications to the designs of AlexNet, the VGG 16-layer net, and GoogleLeNet classifiers. Each network had a convolutional layer made up of 21 1×1 filters attached to the end of it, followed by a deconvolution layer made up of filters that at first perform bilinear interpolation but later learn to scale non-linearly in order to increase resolution by 32 times the output of the convolutional layer. To create a heatmap image accurate to the input image in terms of its spatial dimension, this rescaling seeks to convert the coarse information into pixel-dense information. While maintaining the spatial dimension of the preceding layer, the convolutional layer with 1×1 filters aims to predict belonging to one of the 21 classes provided in the PASCAL dataset (n.d.). The architecture based on the VGG16 classifier (FCN-32s) demonstrated the best performance after evaluating each network on the PASCAL VOC 2011 validation dataset (n.d.), providing a mean IoU (mean intersection over union) of 59.4. Despite the new network's high scores, the output's lack of detail rendered it unsatisfactory. The 32-pixel stride of the final layer caused an abrupt scaling that reduced segmentation quality by illegibly separating pixels belonging to various classes. The authors suggest the addition of "skips" that have the job of transmitting the information generated by specific convolutional layers, which preserve the fine details related to the localization of certain features, to the deeper layers, which hold information with a degree of high semantic but not aspect related details. This will help to refine the level of detail of the segmentation. By halving the stride of the last deconvolution layer from 32 to 16 pixels, the FCN-16s architecture was created. A new convolutional layer with 21 1×1 filters receives the output data from the fourth pooling

layer (pool4) as input to produce predictions. The predictions from convolution layer 7 (conv7), which were first put via an upsampling layer where their spatial dimensionality was doubled to match the predictions from pool4, are then added to these predictions. The final layer, a deconvolution layer with a stride of 16, will use the summation's output as its input. The average IU score for FCN-16s was 62.4, or nearly a 3.0 improvement in performance. From the standpoint of the obviously improved quality of the produced heatmap, this could also be seen visually. The FCN-8s are the product of one last attempt to enhance the architecture. The performance was not much improved when the stride was decreased from 16 to 8 and information was sampled beginning with layer 3 of the pool (pool3), resulting in a mean UI score of 62.7, which was just 0.3 points higher than that of FCN-16s. The article shows that fully convolutional neural networks can operate at the cutting edge of semantic segmentation.

Fully Convolutional Networks also employ the approach described in Tensmeyer & Martinez's work (2017) as a solution. The architecture suggested here, in contrast to the one developed by Long et al. (2015), is based entirely on a convolutional neural network that was created from the start rather than being a translation of an existing classification neural network. The network has four branches as a result of average pooling over layers using a 2×2 size filter. Each branch includes the ability to separately extract features from the input at resolutions of $\frac{1}{1}, \frac{1}{2}, \frac{1}{4}, and\, \frac{1}{8}$. Similar to the "skip" concept described in Long et al (2015), this extraction of details from various spatial dimension lessness and subsequent merging to return them to the input resolution after undergoing some upsampling operations (bilinear interpolation) serve to preserve the location data of the pixels for improved positioning of them in the final heatmap. With regard to neural network parameters, it has a maximum depth of 9 convolutional layers, with 64 filters of size 9×9 on each layer.

Pseudo F-measure (P-FM), a modified form of F-measure (FM), is the proposed cost function. In contrast to FM, P-FM employs various weights that are connected to the pixels in the GT image and that seek to penalize more the misprediction of pixels that can influence the legibility of specific characters in the final product. For instance, the estimated error will be more affected by nearby background pixels that have been mistakenly identified as foreground (False Positives) than by pixels farther away. The authors created this cost function's partial derivative in order to optimize it using stochastic gradient descent (SGD). Palm Leaf Manuscripts datasets and DIBCO 2009-2016 (n.d.) datasets were utilized for training and

validation, respectively. Relative Darkness features (which are determined by counting the pixels that are lighter or darker than a given pixel centered in the chosen size window) and pieces of 256×256 resolution images recovered from the aforementioned datasets were utilized as input, to give the model more details so that it can distinguish between background and foreground pixels more effectively. Although FCNs are capable of learning some non-linear ways of differentiating pixels using only their raw value, there are still some ways of differentiation that are difficult to infer. This is the inspiration behind the usage of extra features. easily comparable to the Sauvola & Pietikäinen algorithm's standard deviation (2000). To determine the best cost function, the authors ran a few trials. FCNs were trained to maximize the following cost functions in this regard: P-FM, FM, Cross-Entropy, and P-FM + FM (the addition of these functions helps to penalize False Negatives on the edge of foreground pixels, for which there are no associated weights in terms of P-FM). The top score utilizing the P-FM metric (94.09) was provided by the FCN trained to optimize the P-FM function, and the best metric scores FM (90.20) and DRD (3.62) were also obtained in tests using the HDIBCO 2016 dataset (n.d.). The Cross-Entropy cost function was optimized to produce Distance Reciprocal Distortion. Peak Signal to Noise Ratio, or PSNR, was measured by the FCN trained with the FM loss function, which scored the highest (18.73). The FCN trained with the P-FM cost function produced a P-FM score of 97.15 while also incorporating the additional Relative Darkness parameters. This article explains how adding more features can significantly enhance binarization performance and how training on segments of photos rather than entire images can reduce the danger of overfitting on the training set, leading to a higher testing score.

## 3. The proposed solution

The suggested approach is based on a Fully Convolutional Network architecture whose component elements were chosen in relation to the architectures provided by Long et al. (2015) and Tensmeyer & Martinez (2017) in order to acquire the most accurate binarization of a document, whether it is handwritten or typed (see Figure 1).

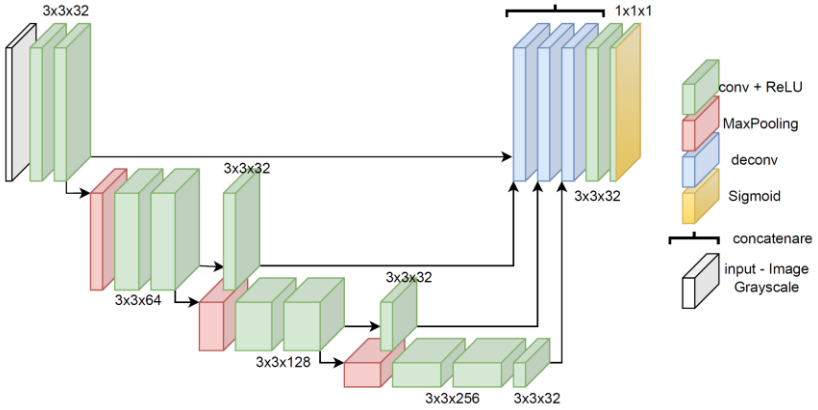



Fig. 1 Description of the proposed architecture
Source: Authors' own conception

The network divides into 4 branches that differ from one another in terms of the depth of the channel, or the number of filters used in a convolution operation, as well as their spatial immensity, or height and width, respectively. In a similar manner to Tensmeyer & Martinez (2017), features will be able to be extracted from input with various resolutions, such as $\frac{1}{1}, \frac{1}{2}, \frac{1}{4}, and \frac{1}{8}$.

Each branch begins with two convolutional layers with the goal of detecting both local features, which preserve how the information is located in the image from a spatial point of view (primary features such as different basic shapes, lines, or curves can enter here, which can further constitute letters, various elongations of handwritten words, or decorative elements), as well as global, semantically rich ones that capture high-level features (for example, elongated letters, elongated words, or decorative elements).

Through the use of a Max Pooling layer, it is possible to move from one branch to another. This operation selects and forwards only those areas where a particular characteristic is present in the most obvious way (convolution works by translating areas with high feature map values into those where specific features have been picked up by the filters). In order to do this, the largest value within a window of a specific size is selected (2×2 in the current work). The window is then moved along the feature maps produced by the preceding convolution. The network is kept invariant to the numerous places (translations, rotations, and various scalings thereof) where a particular feature can be located thanks to the Max Pooling procedure.

Filters (kernels) are used to perform each convolution operation (apart from the final one, prediction), which looks for features in a window

with a fixed size of 3x3 pixels. As we travel from the upper to the lower branch, the number of filters doubles and the resolution at which features are sampled is cut in half:

- Branch 1: has 32 filters with a resolution of 128×128.
- Branch 2: a 64×64 resolution and several 64 filters.
- Branch 3: has a 32×32 resolution and a total of 128 filters.
- Branch 4: a 256-filter system with a 16x16 resolution.

In addition to branch 1, the three lower branches, 2, 3, and 4, will each include a convolution layer that seeks to equal branch 1's number of feature maps.

In order to match the resolution of branch 1, the feature maps produced by the final convolution layer on branches 2, 3, and 4 will then be rescaled. A deconvolution layer that is attached to the end of each branch is used to scale the image. Each deconvolutional layer will employ 3x3 kernels, which can learn over time to perform non-linear resizing that can enhance the quality of upscaling, as opposed to layers that implicitly perform bilinear interpolation. This method was applied in the research that Long et al. (2015) proposed. The following method is used to resize the outputs of the lower branches:

Branch 2 will double the output resolution (deconvolution with stride = 2).

Branch 3 will result in a four-fold increase in output resolution (deconvolution with stride = 4).

Branch 4 will result in an 8-times increase in output resolution (deconvolution with stride = 8).

Similar to Tensmeyer & Martinez (2017), we performed an operation of concatenating them, followed by a new convolution layer to reduce the number of feature maps to the initial characteristic size of branch 1, or 32, after ensuring that the outputs of all branches have the same dimensions in terms of both resolution and number of feature maps. Convolution is used to preserve spatial position information and concatenation is used to combine coarse information with a strong semantic character from the lower branches with fine information to increase the amount of detail in the final prediction.

After a final convolution layer, which this time uses kernels of size 1×1 since its sole function is to bring the channel depth down to 1, the prediction is made.

Since the final prediction will only take into account the background and foreground membership classes, respectively, they can be labeled using a single pixel with a value of 1 for the background and 0 for the foreground.

Because the numbers it returns are in the range [0,1], which is perfect for making predictions, we used the sigmoid (see Figure 2) activation function for the prediction layer. In addition, the sigmoid function is advised for binary predictions (Tensmeyer & Martinez, 2017) as opposed to softmax, which is advised for multi-class classifications (Long et al., 2015).

$$\text{Sigmoid Function: } S(x) = \frac{1}{1 + e^{-x}} \quad \#(1)$$

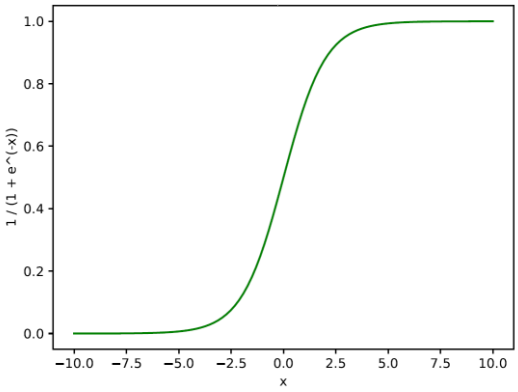



Fig. 2 Sigmoid function
Source: *Bishop, C. M., & Nasrabadi, N. M. (2006)*

In the backpropagation step, Binary Cross-Entropy is employed as the cost function to update the network weights (see Figures 3 and 4). As the error (the difference between observation and prediction) increases, this function offers an increasingly bigger gradient, making it suited for binary classifications, where the output produced by the network is in the range [0,1]. A higher value of the gradient will result in a larger step toward network optimization, or finding the weight values for which the function has a value that is as close to the minimum as possible.

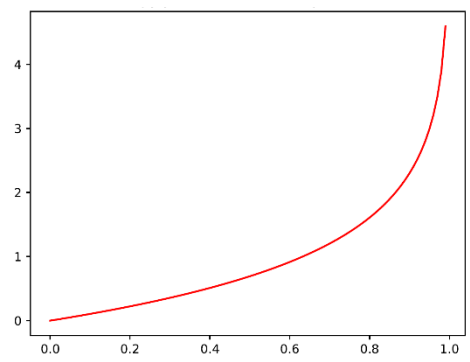



Fig. 3 Cross-Entropy correct prediction is 0

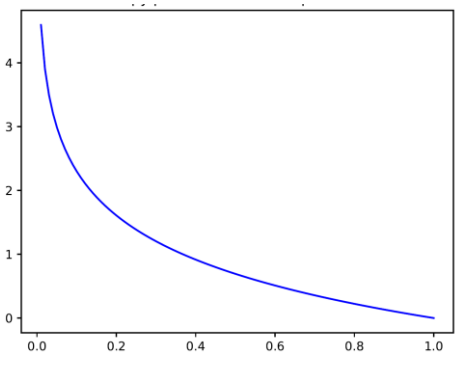



Fig. 4 Cross-Entropy correct prediction is 1

Source: Authors' own conception

### *The Dataset*

The model was trained, validated, and tested using 128×128-pixel images that were cut out of documents and provided as ground truth by the DIBCO data sets from 2009 to 2019 (n.d.). These data sets include photographs of both handwritten and machine-printed papers, as well as parchments with varied degrees of degradation.

The image fragments were converted to black and white, and after that, their pixel values were normalized to fall between [0, 1].

The training process took place across 300 epochs, and we utilized the mean intersection over union (mIoU), accuracy, and the value of the cost function (loss) as metrics to assess how well the convolutional network performed during training.

We chose mean intersection over union (see the equation below) as the primary metric for tracking the model's convergence because it calculates the degree of overlap of the prediction on the surface of the ground truth and provides a clearer picture of the model's progress. This is because the accuracy and value of the cost function cannot adequately reflect the performance of the model if there is an imbalance in the distribution of pixels from different classes (class imbalance).

$$mIoU = \frac{TP}{TP + FP + FN} \quad \#(2)$$

TP denotes the number of accurate predictions for a class-0 pixel. FP is a measure of how often predictions incorrectly classified a pixel as class 0 when it actually was class 1. The number FN indicates how many forecasts assigned a pixel to class 1, which is class 0. Within a binary classification process, the values 0 and 1 represent classes.

### *Document binarization*

The first step was the preparation operations on the document before performing a binarization on the whole thing.

Given that the network's learned weights perform best on images with a resolution of 128×128, which is the resolution used to train the model, the choice was to resize the document so that it may be divided evenly into 128×128 parts. This was accomplished by using pixels that had been taken from the image to extend its width and length by the necessary number of lines and columns.

The generated image is then divided into 128×128 parts, each of which is utilized to generate a forecast. The predictions will assign the value 1 (foreground) to pixels with values above 0.5 and the value 0 (background) to pixels with values below 0.5. After that, the pixels will be resized to have values between 0 and 255, or black and white.

After the prediction has been processed, effective binarization is obtained by removing from the prediction a certain number of lines and columns that match the dimensions of the original image.

## 4. Methodology: The Fully Convolutional Network implementation

### *The Model*

We began the model's implementation by creating the file model.py, with the function my_model(input shape), which takes as a parameter a tuple of the type (a, b, c), where c characterizes the input's spatial dimension and a, b, and c are the input image's height, width, and channel size, respectively.

### *Convolution layers*

Utilization of the Conv2D, BatchNormalization, and Activation classes provided by the layers module of the Keras library (n.d.) was employed to construct the convolution layers. These classes assisted the Conv2DWithBatchnorm function, which returns a convolutional layer with a size determined by the kernel size argument and a number of filters determined by the filters parameter.

Convolution operation output is subjected to batch normalization, which involves rescaling the numbers to have an arithmetic mean of 0 and a standard deviation of 1. By doing this, it will be made sure that the information moves along the network with a fixed distribution of the values. Since it permits the use of greater values for the learning step (learning rate), as demonstrated by Ioffe and Szegedy (2015), this has been shown to

promote quicker model training. The switching from one branch to another is accomplished through a layer of the Max Pooling class.

***Layers of deconvolution***

The action of resizing the output produced by branches 2, 3, and 4 occurs after each final convolution layer in a way that coincides with the resolution of branch 1. The Conv2DTranspose class, which is also part of the layers module of the Keras library, was used to implement this process of employing deconvolutional layers to increase resolution.

***The concatenation operation***

Concatenating all the feature maps on the four branches yields a total of 128 feature maps, which were reduced to the original 32.

***The layer of prediction***

The operation of concatenating these features was made through Concatenate class, also present in the framework keras.layers, after making sure that all branches produce a comparable output in terms of spatial dimensions (32 feature maps apiece, each at a resolution of 128×128). The final convolution operation, which aims to make the ultimate prediction, is performed at the conclusion.

***Compiling the model***

The cost function of the optimization method and the performance evaluation criteria for the built-in neural network must be chosen when creating the model.

The model that was produced is known as branching completely convnet 2. Everything has been put together to use the stochastic gradient descent (SGD) algorithm with a learning rate of 0.1 to optimize the Binary Cross Entropy function.

The Adam class in the keras.optimizers module implements the stochastic gradient descent algorithm. Using the input and output tensors, the Model class aids in the construction of a neural model. Were defined the accuracy and mean intersection over union metrics when declaring the metrics for the compile method's model performance evaluation.

***The Model implementation (pseudocode)***

The data processing sequence is briefly presented below.

**Algorithm 1:** Experimental model - proposed solution

**Data:** DIBCO dataset
**Result:** Images binarization using FCN — Fully Convolutional Network

$testImage \leftarrow [128 \times 128]$;

$groundTruthIimage \leftarrow [128 \times 128]$;

```
// Getting the data and processing it
```

**foreach** *testImage, groundTruthIimage* **do**
    Process DIBCO images to create inputDataset;
    Normalization;
**end**

```
// Building the Model
```

Building the Model;
Building the convolution layers;
Concatenating the feature map;
Prediction layer;
Compiling the Model;
Training the model;
Evaluation;

Source: Authors' own conception

## 5. Results and Evaluation

### *Training model performance*

With a learning rate of 0.1 over 300 epochs, the model was trained in roughly 5 hours. Accuracy, loss (cost function value), and mIoU were the metrics we used to track the model's development throughout training.

The training dataset, as we said in the previous sections, was made up of patches with a size of 128x128 that were taken from the official DIBCO datasets (n.d.) from the years 2009-2016 and 2018-2019. There were 7339 of these slices in the dataset, adding up to 120242176 pixels.

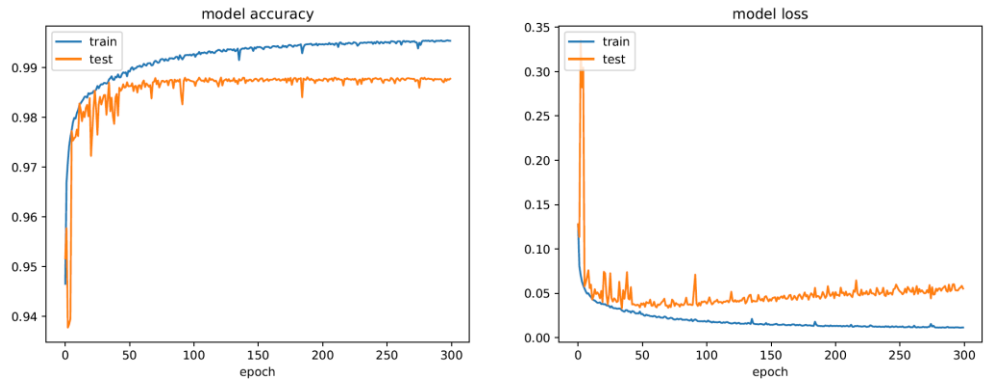



Fig. 5 Accuracy (left) and loss (right) evolution - shows how accuracy and loss measures changed as training progressed
Source: Author's own conception

Both accuracy and loss have made significant progress since the beginning of time. This shows that the model consistently makes numerous accurate predictions for the dataset's dominating class, the background pixel class.

The incorrect forecasts of foreground pixels have a negligible weight in comparison to the accurate predictions of background pixels because there are many more background pixels. Thus, in order to gain a better understanding of the model, it is required to make use of various measures that assess performance both from the standpoint of accurate predictions at the data set level and the level of the class.

mIoU has a substantially slower growth rate than the accuracy and loss metrics, which is a result of the foreground pixels' inaccurate predictions. The mean intersection over union metric on both the training and validation sets started to converge around the 0.6 threshold after a little more than 250 epochs.

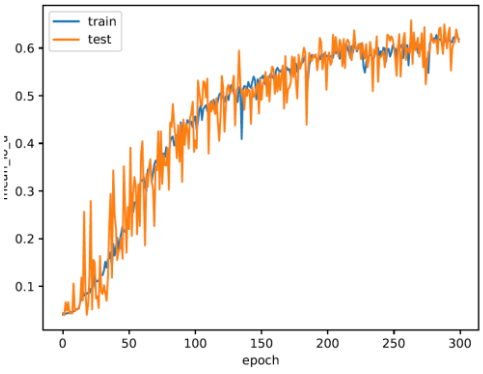



Fig. 6 mIoU values during training

Source: Authors' own conception

As observed in Figure 6, the mIoU on the validation set increases directly proportional to the one calculated on the training set, indicating that the model does not exhibit overfitting, despite occasional volatility brought on by the rapid learning rate.

### *Model effectiveness on the test set of data*

1812 patches of size 128x128 that were taken from the DIBCO 2017 dataset made up the test dataset. Batches of size 32 were fed to the model.

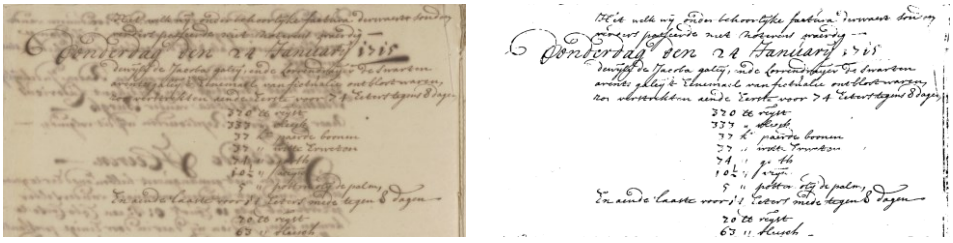
Using the test data set, the model was able to achieve the following results: mIoU score of 0.589 (58.9%), accuracy of 0.9823, loss of 0.0931, precision of 0.916, recall of 0.883, and F-measure of 0.899 (89.9%). It took 10 seconds to deduce the 1812 photos.

### *Binarization tests on many sorts of documents*

This section examines the model's performance against a variety of flaws that can be present in a document-type image, such as the bleed-through effect and lighting regions of excess or deficiency. Any model attempting to acquire the most accurate binarization has difficulties of this kind, so we believed that the model's performance could be evaluated by putting it to such a test. The DIBCO 2017 test dataset was used to create the images.

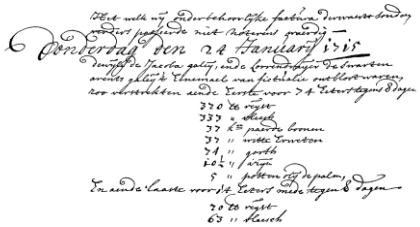
The time required to binarize a whole document varies depending on its size. The amount of patches that can be recovered from the image has a direct correlation with the inference time. The model delivers a prediction in roughly 30ms when using the GPU.

### *Bleed-through on documents with handwriting*



(a) original image;
Source of the figure: DIBCO

(b) proposed binarization;
Source: Authors' own conception

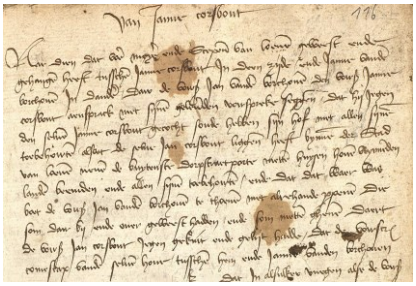


(c) ground truth; Source of the figure: DIBCO

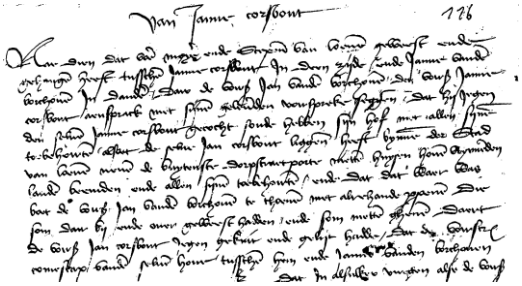
Fig. 7 Shows a qualitative depiction of the author's own solution in relation to a handwritten manuscript with bleed-through

Figure 7 illustrates how the model may identify the writing produced by the bleed-through effect as useless information and classify it as background. Given that the hues of the two categories are similar, a problem occurs when the letters created by the bleed-through effect and the letters expressing the meaningful information overlap.

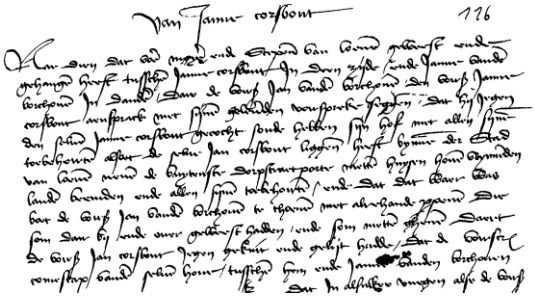
### *Written materials that have stains*



(a) original image;
Source of the figure: DIBCO



(b) proposed binarization
Source: Authors' own conception



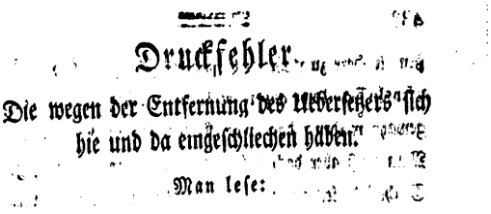
(c) ground truth; Source of the figure: DIBCO

Fig. 8 Shows a qualitative representation of the user's solution in relation to a stained handwritten paper

Figure 8 shows a qualitative representation of the user's solution in relation to a stained handwritten paper.

### *Bleed-through on printed materials*


(a) original image;
Source of the figure: DIBCO



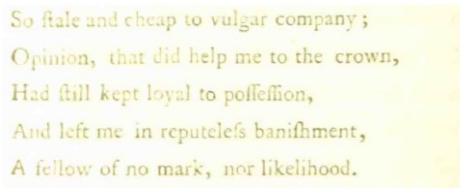

(b) proposed binarization
Source: Authors' own conception


(c) ground truth; *Source of the figure: DIBCO*

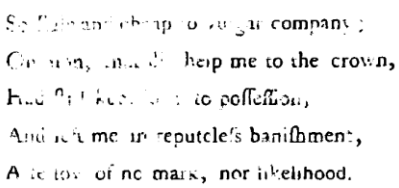
Fig. 9 Shows a qualitative portrayal of the author's solution within the context of a bleed-through-affected printed paper

Figure 9 shows a qualitative portrayal of the author's solution within the context of a bleed-through-affected printed paper. It also illustrates that the model in this instance can identify the letters in the foreground. The issue arises, though, when other elements have the same color as the foreground, including decorative accents or bleed-through letters that are the same color and shape as the foreground.

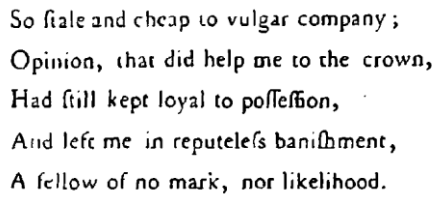
### *The impact of harsh illumination on printed materials*




(a) original image
Source of the figure: DIBCO




(b) proposed binarization
Source: Authors' own conception




(c) ground truth
Source of the figure: DIBCO

Fig. 10 Shows a qualitative portrayal of one's own solution within the context of a printed page

Figure 10 shows a qualitative portrayal of one's own solution within the context of a printed page that has been negatively impacted by excessive illumination.

The letters' strength is significantly diminished by the light, which in certain spots results in an unacceptably close resemblance between the foreground and background shades. As seen in Figure 10, this causes problems for the model and leads to inaccurate predictions for foreground pixels influenced by intense light.

### *Evaluation of comparable systems*

Tensmeyer & Martinez (2017) and Long et al. (2015) are the systems we have previously mentioned that presented an approach comparable to the one we employed and they served as inspiration and source of ideas.

| Model | Number of convolutions | Upsampling | Downscaling |
|---|---|---|---|
| Tensmeyer | 24 | Bilinear interpolation | Average Pooling |
| Long | 18 | Deconvolution | Max Pooling |
| Proposed solution | 13 | Deconvolution | Max Pooling |

| Model | Features combination | Input resolution | Extra characteristics |
|---|---|---|---|
| Tensmeyer | Concatenate | 256×256 | Relative Darkness |
| Long | Sum | Variable | - |
| Proposed solution | Concatenate | 128×128 | - |

| Model | Maximum number of filters | Maximum number of filters | Prediction function |
|---|---|---|---|
| Tensmeyer | 64 | 9 | Sigmoid |

| | | | |
|---|---|---|---|
| Long | 4096 | 7 | Softmax |
| Proposed solution | 256 | 3 | Sigmoid |

Table 1: Comparison of architectures
Source: Authors' own conception

Table 1 shows that the suggested solution combines architectural components that were influenced by the two comparable options. The NVIDIA Tesla K40c hardware resources we employed, which have a lower processing performance than that shown in the papers by Tensmeyer & Martinez (2017) and Long et al. (2015), were taken into consideration when designing this architecture. The number of filters, the size of a filter, the number of convolutional networks, as well as the size of the input, were some of the factors we attempted to lower in this regard. All of these convolutional model properties have a direct impact on the model's trainable parameter count as well as the training duration. However, this parameter decrease also clearly had an impact on the accuracy of the forecast. As can be seen in Tensmeyer's & Martinez's solution (2017), we think adding additional convolutional layers would enhance binarization more than raising other parameters. Long et al. (2015) handles a multi-class segmentation problem where a larger number of filters are needed (4096), in order to be able to identify as many features as feasible that can distinguish the 21 classes in the PASCAL data set.

In Table 2, we emphasized the individual scores for each architecture. In terms of the metrics employed in the two works, FM and mIoU, we were able to compare the suggested answer to the alternative solutions. As can be seen, the scores from the model's own analysis are comparable to those from Long et al. (2015) and Tensmeyer & Martinez (2017).

| Model | Dataset | FM | mIoU | Loss function |
|---|---|---|---|---|
| Tensmeyer | HDIBCO 2016 | 90.2 | - | Binary Cross-Entropy |
| Long | VOC2012 | - | 62.2 | Cross-Entropy |
| Proposed solution | DIBCO | 89.9 | 58.9 | Binary Cross-Entropy |

Table 2: Compared performances between architectures
Source: Authors' own conception

Given that there are variations in the test data sets and the method of segmentation used, the proposed solution's placement with respect to the

two cannot be perfect. In order to rate the three models fairly, a proper comparison would make use of standard test data sets and metrics.

## 6. Conclusions

Considering the state-of-art solutions in the field of semantic segmentation, both multi-class, such as Long et al (2015), and binary, like the one proposed by Tensmeyer & Martinez (2017), the solution proposed in this paper has proven a good potential for obtaining fast results and with reduced processing resources. Unlike other approaches that used the Caffe framework (BAIR, n.d.) for architecture development, choosing to implement the entire model as well as the training, testing, and binarization infrastructure using the Python language (Kasmin et al., 2017) and available libraries, was the best option for the existing resources. However, the simplicity of the architecture came with a compromise in terms of the visual quality of the binarization. As shown in the results section, certain cases presented difficulties in correctly categorizing pixels due to highlighted defects within the documents. Also, rising the number of convolutional layers to introduce new levels of abstraction in the feature hierarchy, allowing the model to learn new differentiating factors between useful writing, decorative elements, or the bleed-through effect, could lead to improved results. A larger dataset containing documents with more diverse and intense levels of degradation could enable the model to better handle such situations.

## Acknowledgment

The results presented in this article have been funded by the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735.

## References

BAIR. (n.d.). *Deep learning framework*. Caffe. Retrieved June 2, 2023, from https://caffe.berkeleyvision.org/

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

DIBCO. (n.d.). *DIBCO dataset*. DIBCO. Retrieved June 2, 2023, from *https://dib.cin.ufpe.br/#!/resources/dibco*

Hamza, H. & Smigiel, E., & Belaid, E. (2005). Neural based binarization techniques. In *Eighth International Conference on Document Analysis and*

*Recognition (ICDAR'05), Seoul, Korea (South),* Vol. 1 (pp. 317- 321). IEEE. https://doi.org/10.1109/ICDAR.2005.168

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*. https://doi.org/10.48550/arXiv.1502.03167

Kasmin, F., Abdullah, A., & Prabuwono, A. S. (2017). Ensemble of Steerable Local Neighbourhood Grey-level Information for Binarization. *Pattern Recognition Letters, 98*, 8-15. https://doi.org/10.1016/j.patrec.2017.07.014

Keras. (n.d.). Keras Python library. Keras. Retrieved June 3, 2023, from *https://keras.io/api/*

Long, J., & Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA* (pp. 3431-3440). https://doi.org/10.1109/CVPR.2015.7298965

Niblack, W. (1985). *An introduction to digital image processing*. Strandberg Publishing Company.

Otsu. N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076

Patchify. (n.d.). *Project description*. Patchify Python Library. Retrieved June 1, 2023, from https://pypi.org/project/patchify/

Pattern Analysis, Statistical Modelling and Computational Learning. (n.d.). *The PASCAL Visual Object Classes Homepage. Pascal datasets*. PASCAL Retrieved June 3, 2023, from *http://host.robots.ox.ac.uk/pascal/VOC/index.html*

Sauvola, J., & Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, *33*(2), 225–236. https://doi.org/10.1016/S0031-3203(99)00055-2

Tensmeyer, C., & Martinez, T. (2017). Document Image Binarization with Fully Convolutional Neural Networks. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017* (pp. 99-104). https://doi.org/10.1109/ICDAR.2017.25