Instance Segmentation Of Newspaper Elements in JPEG Domain

Aman Raj Patwa, Rahul Kumar Yadav,
Hrutvik Nagrale, Ravi Kumar, Sagar Kumar

May 20, 2021

Abstract

Newspaper archives contain an ocean of information which is being made digitally available. The mass digitization of newspapers, on the other hand, presents its own series of problems. The branch of computer science, Computer Vision is being pushed to its limit to meet such challenges. Almost any information printed on the newspapers can be extracted correctly, To retrieve text from newspapers, technologies such as OCR (optical character recognition) are used.But before we start extracting any information we need to identify which sections of the newspaper are important and which other sections should be ignored. We need to separate articles from advertisements, page headings from main articles, graphs and charts from graphic elements, etc. What makes it more challenging is newspaper layouts, language, text styles, all of these vary from newspaper to newspaper. In this paper we attempt to solve the element wise deconstruction of newspaper, through instance segmentation method. The target categories of newspaper elements will be text articles, heading and advertisements.

Instance Segmentation; JPEG Compression; Fully Convolutional Layers; DCT Coefficients;

1 Introduction

In today's digital era, accessing, preserving and maintaining the non-electric or analog information of newspapers is challenging, so Newspaper publishers

are digitizing their present and previous newspapers. Techniques like Optical Character Recognition (OCR) are being used to extract the text from the newspapers. But the very first step, identifying the useful information to extract is the challenging part. Newspapers have complex layouts, different newspapers have different languages and different text styles which makes tasks more difficult. We have to detect different segments of newspaper like - articles, advertising photos, graphs and headers before starting the extraction of any information.

We'll use Fully Convolutional Networks, or FCNs, which are mostly used for semantic segmentation.The only locally connected layers they use are convolution, pooling, and upsampling. There are less criteria since dense layers aren't included (making the networks faster to train). An FCNs qualified model will fragment and classify the different elements of the newspaper. FCNs identify one or more objects in an image and segment the objects that are present. Earlier works on this were not generalized for different newspapers of different styles and layouts.

2 Literature Survey

Newspapers contain a variety of elements such as articles, advertisements and so on, but the focus of the study was on segmenting articles. Many scholars are attempting to extract articles from newspapers on a particular determined language. There had been very little previous research into detecting an instance segmentation model for newspaper, papers that include work in the JPEG domain, to our knowledge. The initial taken approach was a bottom up approach where the component such as coloumn, titles and paragraphs etc were firstly classified and then every individual components are extracted and finally clustered to form an article. Various methods were used for low level extraction methods such as Run-length smoothing algorithm(RLSA), Conditional random field. Various methods were used for clustering of components such as fixed point models and many paper used rule based heuristic.

In the field of article segmentation, ML methods were also used. There are several approaches taken, one of them is text classification and graphics using an assembly of support vector machine (SVM) but does performs

well in complex layout structure. The logical elements of newspapers are often recognised using a neural network approach. Deep learning techniques were also came into role for segmenting the documents. Recently The mask R-CNN model has been used in reference paper for instance segmentation of newspaper where they identified pages into three elemental parts (articles text, advertisements, and page headers) which involves series of two steps, the first a small Region Proposal Network and the second a feature extraction, such as segmenting non-rectangular shapes and so on.

3 Problem Statement and Objective

Problem Statement: Given a newspaper image, segment all major distinct elements (articles, headings, images and advertisements) in the image.

Objective: Deconstructing the information which is useful and separating out the important part from a less important part in the image is a tedious task. Here, we attempt to segment a newspaper image into major distinct categories - articles, headings, images and advertisements. Since we hope to build a fully convolutional network, this task of segmentation can span different languages and irregular shape, size and orientation, irrespective of the layout of the newspaper.

4 Proposed Methodology

We will use a unique approach to go about the problem. There will be major two steps:-

1) Dataset preparation : In this step, the given dataset of newspaper image will go through a series of conversions, first the RGB colour channels of the input image will be converted to YCbCr colorspace. Then after it will be downsampled . After this, we will use Discrete Cosine Transformation (DCT) on each of the channels. The output of the DCT transformations will be recombined to form DCT colorspace image. In the next step, we'll use this image as the input to our model.

2) Element-wise Segmentation on the input images: This step will segment each distinct element in the newspaper images that we will receive from the first step . A FCN model is being implemented here for the instance segmentation. The model will produce different masks for each distinct element (articles, page headings and advertisements) in the newspaper image. This FCNs will be trained using Russian newspaper dataset available publically and a Manually Masked dataset prepared by us.

4.1 DCT Transformation

An image represented in the discrete cosine transform (DCT) form are the sum of sinusoids of varying magnitude and frequency. The discrete cosine transform of a two-dimensional image is calculated using the below dct mathematical equation. For a regular image, as per the DCT property all of the visually relevant data are to found in just a few DCT coefficients.

The two-dimensional DCT of an M-by-N matrix A is defined as follows.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

Figure 1: DCT mathematical equation.

Digital Y'CbCr (8 bits per sample) is derived from analog R'G'B' as follows:

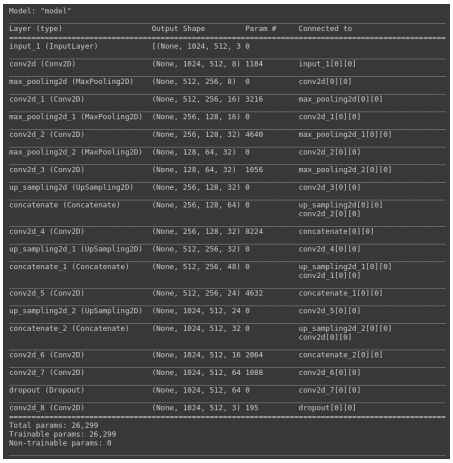
$$\begin{aligned} Y' &= 16 + (65.481 \cdot R' + 128.553 \cdot G' + 24.966 \cdot B') \\ C_B &= 128 + (-37.797 \cdot R' - 74.203 \cdot G' + 112.0 \cdot B') \\ C_R &= 128 + (112.0 \cdot R' - 93.786 \cdot G' - 18.214 \cdot B') \end{aligned}$$

Figure 2: RGB to YCbCr image Conversion.

4.2 Network Architecture

The following is the model summary for the FCN.

Figure 3: RGB image and Corresponding DCT image.



```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 1024, 512, 3 0
_________________________________________________________________________________________________
conv2d (Conv2D)              (None, 1024, 512, 8) 1184      input_1[0][0]
_________________________________________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 512, 256, 8)  0         conv2d[0][0]
_________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, 512, 256, 16) 3216      max_pooling2d[0][0]
_________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D) (None, 256, 128, 16) 0      conv2d_1[0][0]
_________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, 256, 128, 32) 4640      max_pooling2d_1[0][0]
_________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D) (None, 128, 64, 32)  0      conv2d_2[0][0]
_________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, 128, 64, 32)  1056      max_pooling2d_2[0][0]
_________________________________________________________________________________________________
up_sampling2d (UpSampling2D) (None, 256, 128, 32) 0        conv2d_3[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)    (None, 256, 128, 64) 0        up_sampling2d[0][0]
                                                           conv2d_2[0][0]
_________________________________________________________________________________________________
conv2d_4 (Conv2D)            (None, 256, 128, 32) 8224      concatenate[0][0]
_________________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D) (None, 512, 256, 32) 0      conv2d_4[0][0]
_________________________________________________________________________________________________
concatenate_1 (Concatenate)  (None, 512, 256, 48) 0        up_sampling2d_1[0][0]
                                                           conv2d_1[0][0]
_________________________________________________________________________________________________
conv2d_5 (Conv2D)            (None, 512, 256, 24) 4632      concatenate_1[0][0]
_________________________________________________________________________________________________
up_sampling2d_2 (UpSampling2D) (None, 1024, 512, 24) 0     conv2d_5[0][0]
_________________________________________________________________________________________________
concatenate_2 (Concatenate)  (None, 1024, 512, 32) 0       up_sampling2d_2[0][0]
                                                           conv2d[0][0]
_________________________________________________________________________________________________
conv2d_6 (Conv2D)            (None, 1024, 512, 16) 2064     concatenate_2[0][0]
_________________________________________________________________________________________________
conv2d_7 (Conv2D)            (None, 1024, 512, 64 1088      conv2d_6[0][0]
_________________________________________________________________________________________________
dropout (Dropout)            (None, 1024, 512, 64 0         conv2d_7[0][0]
_________________________________________________________________________________________________
conv2d_8 (Conv2D)            (None, 1024, 512, 3) 195       dropout[0][0]
=================================================================================================
Total params: 26,299
Trainable params: 26,299
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figure 4: Model summary

The proposed model is made up of various convolution layers. We have followed an encoder-decoder type model in which first the image is downsampled, which is used to learn lower resolution feature mapping, after which upsampling is followed to distinguish between classes and generate the mask for each of the classes in full spatial resolution.

Downsampling: The input to the model is 1024 * 512 * 3 dimensions image consisting of three RGB channels. The first layer consists of 8 filters of size (7 * 7), which means the output from this layer after the convolution will be of size (1024 * 512 * 8). further, this output is routed into a Max Pooling layer. This max pooling layer applies a stride of (2 * 2). It means two of the dimensions will be halved. Hence the output from this layer will

be (512 * 256 * 8). This completes a single layer of the convolutional layers. The following layer is also a convolutional layer with 16 filters of magnitude (5 * 5). The output from this layer is (512 * 256 * 16). Again, the output is routed to a Max Pooling layer, strides (2 * 2). The dimensions are again halved along the height and width. The output from this layer is (256 * 128 * 16). further a convolutional layer with 32 filters of size (3 * 3) which results in of size (256 * 128 * 32) is routed to a Max Pooling layer whose strides are (2 * 2). We get (128 * 64 * 32) as output from this layer. Coming to the last layer, convolutional layer uses 32 filters of size (1 * 1) in downsampling step. The output results in size (128 * 64 * 32).

Upsampling: Next is the upsampling process in the model. Upsampling is performed to extend the mask that has been produced, after downsampling in lower resolution, to a higher spatial resolution. The output from the last layer is upsampled using a stride of size (2 * 2) which doubles the resolution in height and width. The output from the upsampled layer is (256 * 128 * 32). This is then combined with the second last layer from the downsampling. This process is called concatenation. The result after concatenation is (256 * 128 * 64). Next is the convolutional layer which performs convolution over this output with 32 filters of kernel size (2 * 2). The output from this layer is (256 * 128 * 32). This layer is again fed into upsampling layer and concatenate the output from the corresponding layer in downsampling giving us a result of (512 * 256 * 48) which passes through a convolutional layer consisting of 24 filters of size (2 * 2). This outputs a spatial dimension of size (512 * 256 * 24), which in turn is upsampled with a stride of (2 * 2) and concatenated with its corresponding layer in the downsampling step. The output dimension is of size (1024 * 512 * 32). After that, output passes through two convolutional layers consisting of 16 and 64 kernels, followed by a dropout regularization step. In the final step, we use three kernels of size (1 * 1) to match the input size. The final output spatial dimension is (1024 * 512 * 3).

5 Newspaper images segmentation Data Sets

5.0.1 Russian Newspaper dataset

This data was collected in order to train and test a machine learning algorithm for classifying text, images, and document context. There are 101 scanned images of Russian newspapers and magazines in the collection. The bulk of the images have a resolution of 2400x3500 pixels and are 300 dpi and A4 in size. For each image, pixel-based ground reality masks were manually created.Ground reality pixel-based masks were manually produced for all images.The ground real masks are named after the initial images using the postfix -m. The three classes are the text area, picture area, and background. The graphic field is represented by pixels with the colour 255, 0, 0 (rgb, red colour), the text region is represented by pixels with the colour 0, 0, 255 (rgb, blue colour), and the background is represented by all other pixels on the screen. Images with various coloured backgrounds are used in the dataset.

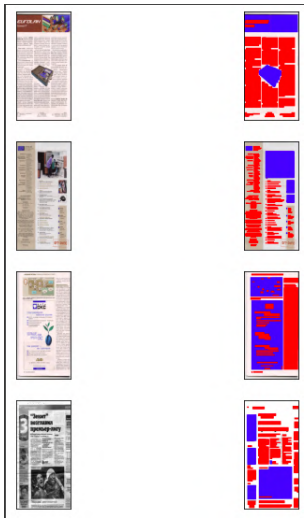


Figure 5: Example: Russian Newspaper Dataset

5.0.2 Manual Dataset Preparation

This dataset contains 150 newspaper images in English, each with an initial image and a masked image. Any of the images are approximately 1024x512 pixels. Ground reality pixel-based masks were manually produced for all images. The ground real masks are named after the initial images using the postfix -m. Article, image, background, heading, and other valuable contents are the five classes.Images with various coloured backgrounds are used in the dataset.
The Images have been masked using GIMP, an open-sourced cross-platform image editor. Each section of newspaper is masked using one of different like. The colour encodings get changed once it is loaded into the model as BGR. The colour encodings in the original masked dataset are:

1. Article : Red
2. Heading : Black
3. Image : Green
4. other : Pink

6 Experiment

Experiment 1: In experiment 1, we kept the size of the input to be (128 * 128 * 3). This is a compressed image obtained after Min-Max Normalization. As we can see, this is not quite producing the result that we wished to obtain. The kernel size in the downsampling step has been kept constant. The kernel size is of (3 * 3) here. The outputs are shown below:

Observation: In the above results, we can see that the output is only able to classify the article with the blue color. The headings are not that much distinguishable, and same is the case with the headings and other texts.

Training/Validation Loss:
: The reasons for such indistinguishable results are quite obvious. The input size is too crunched up to identify individual feature mapping at lower

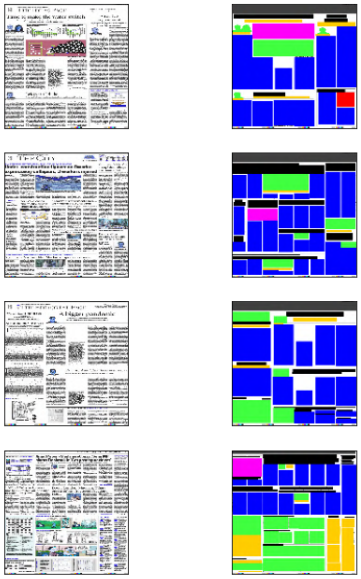


Figure 6: Example: Manual Masked Dataset

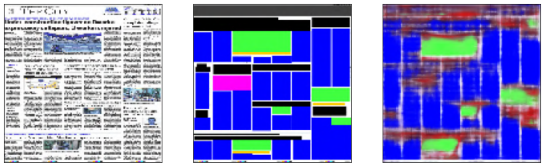


Figure 7: Experiment 1: Output

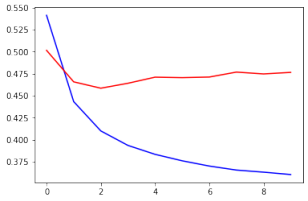


Figure 8: Experiment 1: Loss

resolution. The kernel size might also have affected the result since it is kept constant during the whole downsampling process. As a result, the mask produced during the upsampling step has been spread and are not distinctive of any single feature.

Experiment 2: In experiment 2, to obtain a better result and classify them into different classes with their corresponding masks, we increased the size of the input to (256 * 256 * 3). The image as well as the mask is compressed from its original dimension using min-max normalization. The kernel size is kept constant as in the previous experiment. The outputs are shown below:

Observations: In the above results, we can see that the articles are better

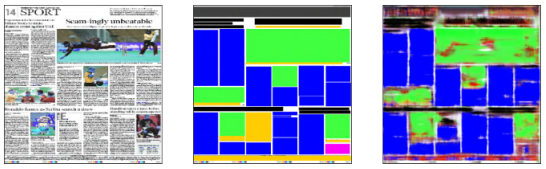


Figure 9: Experiment 2: Output

classified with blue color, with more sharp edges. The images are also better classified with the green masks, however it consists of some tints of different colors. The headings are not so well distinguishable, but it is better classified than the previous experiment in which no heading was classified. The other texts have also been slightly masked with red color, though not accurately.

Training/Validation Loss: Reasons: The reason why this experiment proved better than the previous experiment is that the input image is slightly more distinguishable than the previous ones. During the downsampling process, with the size of the input increased, the low resolution feature mapping is able to learn more distinctive features such as the articles and the images, which have been classified better. Even the headings of the article are classified but it doesn't have much sharp edges.

Experiment 3: In this experiment, we kept the size of the input to be the same as the previous experiment. We changed the kernel size from (3

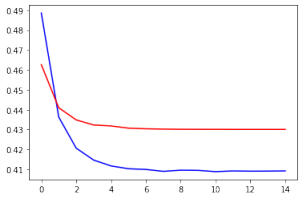


Figure 10: Experiment 2: Loss

* 3) to reflect the change in the spatial dimension as we went from higher dimension to lower ones. The size of the kernel have been changed in the downsampling process, first convolutional layer is of size (7 * 7), then the next layer's kernel size is changed to (5 * 5), then in the next layer, changed kernel size to (3 * 3) and in the last step of the downsampling process, the kernel size is (1 * 1). The outputs are given below:

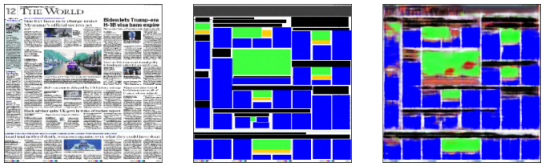


Figure 11: Experiment 3: Output

Observations: The headings are much more precisely classified in their black colors. The articles with the blue color have kept their sharp edges, thus, distinguishing different elements. One more thing to notice is that the texts below the images are classified with the red color. This is a different color because it is neither an article nor a heading. The images have been particularly identified with their green masks. These masks have near-perfect sharp edges.

Training/Validation Loss:
Reasons: The reason for better classification in the headings may have been that during the downsampling process, the kernel size also decreased with the decrease in spatial dimension of the input. This may have helped in

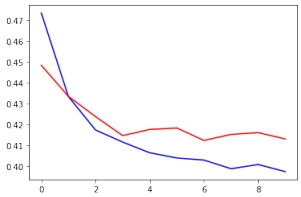


Figure 12: Experiment 3: Loss

classifying more separate elements for the model.

Experiment 4: In this experiment, the input size have been changed to (512 * 512 * 3) dimensions. The kernel size has been kept the same as in experiment 3 to cope with the changes of the spatial dimension in the downsampling process. The results are shown below:

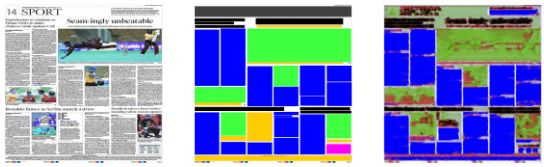


Figure 13: Experiment 4: Output

Observations: This is a better result than the previous one because we are able to clearly identify the edges for each of the distinct elements. The headings have been more properly identified with the black color. We can see improvements from experiment 4 because here the headings are not that much smudged. Moreover, the text below the images have been identified clearly with the red color. The article classification is also fine. The only problem is with the images. They contain more varied colors than green. Their mask is not properly colored as well as it contains tints of red color. This is an improvement over the previous experiments.

Training/Validation Loss:
Reasons: The reasons for this improvement is that the input has been fed

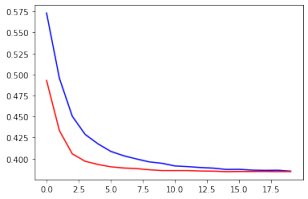


Figure 14: Experiment 4: Loss

with increased dimensions, plus the kernel size is changing as we go deeper into the convolutional layer. This helps in distinguishing the feature map for different elements separately.

Experiment 5: In this experiment, the input for the model has the dimension (1024 * 512 * 3). The kernel size has been kept the same as experiment 4. The outputs are given below:

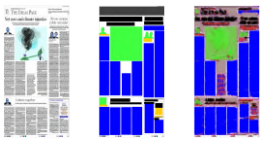


Figure 15: Experiment 5: Output

Observations: Though the input size has been increased, there is hardly any difference from previous experiment 4. The images have sharp edges, the articles are well classified, but the heading is more smudged than before. The right most corner has the headings in the masked input image but the predicted mask has red color, which is wrong, because it implies that this is not a heading. Hence, this is a little poor as compared to experiment 4.

Training/Validation Loss:
Reasons: The poor classification of headings might have been due to the poor train-test split. Other reasons such as increase in the input size might also contribute to the result but it is not obvious.

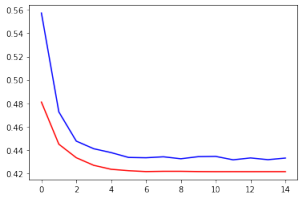


Figure 16: Experiment 5: Loss

6.1 Implementation Detail

Serieswise steps has been taken to implement our model till predicting the output. With the use of google colab (GPU) as backend, every model has been trained. There we have four model with different input format image.Following are the step:
1) Importing library: Following libraries are required.

1. os
2. numpy
3. cv2
4. matplotlib
5. sklearn
6. keras
7. scipy
8. math
9. tensorflow

2) Loading dataset: The original dataset contains the images in the RGB color space which is to be converted to YCBCR color space which is then passed into a DCT transformation function which returns output in form of DCT image. The output images are then used to train and test the model.

3) Splitting dataset: The data set contains an image and the corresponding mask image hence, needs to be segregated. All images and Masked images

are separated by its image title. Further the images is splitted to x-train and x-test and the masked images are splitted to y-train and y-test.

4) Model Preparation: The Above Model summary includes convolutional layers and Pooling which extracts outs features from the image.

5) Image Batch and Mask Batch: we used 7 as image batch size.The ImageDataGenerator takes the initial data, transforms it randomly, and then returns the converted data.

6) CallBack functions Setup:

1. Model Checkpoints
2. EarlyStopping

These 2 parameter need to be setup before the model fitting.

7) Model fitting:

1. Steps per epoch = 100
2. epoch = 20
3. verbose = 1

These are the essential parameter to train the model. The model training is done.

8) Predicting Output: The Above trained model is used to predict the output tor the test dataset image.

9) Loss function: The Above model uses ADAM optimiser and loss function is calculated through binary cross entropy.

7 Results

This section consists of different experiments results as output image and the loss and accuracy vs epoch curve.

The curve consist of :
loss : Blue
validationloss : Red
Accuracy : Green
ValidationAccuracy : yellow.

7.1 Result 1: Russian Newspaper Segmentation

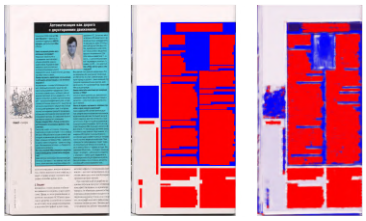


Figure 17: Predicted output of the Russian dataset.

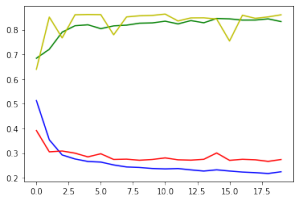


Figure 18: Loss and Accuracy graph of the Russian dataset..

7.2 Result 2: Manually Masked Newspaper Segmentation with RGB input image.

Figure 19: Predicted output with RGB image input.

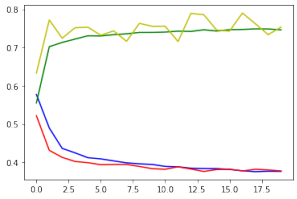


Figure 20: Loss and Accuracy graph of RGB image input.

7.3 Result 3: Manually Masked Newspaper Segmentation with DCT input image.

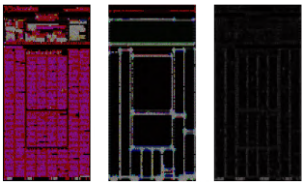


Figure 21: Predicted output with DCT image input.

7.4 Result 4: Manually Masked Newspaper Segmentation with DCT image and RGB masked input image.

Tablewise Summary

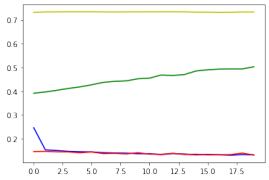


Figure 22: Loss and Accuracy graph of DCT image input.

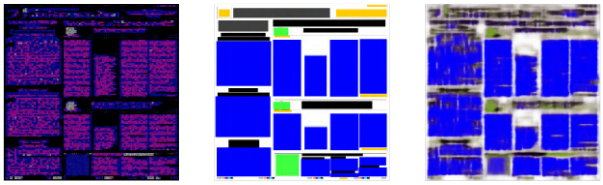


Figure 23: Predicted output with DCT image and RGB masked input.

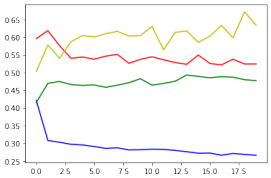


Figure 24: Loss and Accuracy graph of DCT image and RGB masked input image

	RGB	***DCT***	***Mixed***	***Russian dataset***
Train loss	36.28	12.60	47.24	21.79
Train Accuracy	78.25	73.12	66.44	90.59
Test loss	37.74	13.11	52.42	27.41
Test Accuracy	75.32	73.38	63.36	86.01

Figure 25: Loss and Accuracy table.

8 Conclusion

In this paper, A FCN model has been presented for segmenting a newspaper elements (articles, advertisements, background and article headings).we have experimented with the RGB image and DCT transformed image for our datasets.The experimental results of our model shows the accuracy and loss of all 4 types of experiments we performed. Excluding the Russian dataset , for manual masked dataset ,accuracy for RGB and DCT while training is 0.7825 and 0.7312 respectively , While Mixed accuracy will be 0.6644. In the result while experimenting we get highest training accuracy while using Russian dataset which is 0.9059 .Now from the result Test Accuracy is higher for Russian dataset and minimum for mixed. Hence,this concludes all work done.

References

[1] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," 2017 IEEE International Conference on Computer Vision (ICCV), Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.32.

[2] Vladimir Iglovikov,Selim Seferbekov, Alexander Buslaev,Alexey Shvets, : TernausNetV2: Fully convolutional network for Instance Segmentation.

[3] A. M. Vilkin, I. V. Safonov, M. A. Egorova. Algorithm for segmentation of documents based on texture features Pattern Recognition and Image Analysis March 2013, Volume 23, Issue 1, pp 153-159.

[4] A. Bansal, S. Chaudhury, S. D. Roy, and J. Srivastava, "Newspaper article extraction using hierarchical fixed point model," in 2014 11th IAPR International Workshop on Document Analysis Systems. IEEE, 2014, pp. 257–261.

[5] 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. 129–13.

[6] https://archive.ics.uci.edu/ml/datasets/Newspaper+and+magazine+images+segmentation+dataset