Document style transfer with Generative Adversarial Networks

Benjamin Gallusser*
Department of Computer Science
ETH Zürich, Switzerland
gallussb@ethz.ch

Hlynur Jónsson*
Department of Computer Science
ETH Zürich, Switzerland
jonssonh@ethz.ch

Abstract

Transforming the style of images without changing the content has been done successfully with Conditional Generative Adversarial Networks. A common limitation of this approach is that the general structure of an image is not altered. We combine techniques from image style transfer and semantic segmentation to learn spatial translations of content elements from paired image data. With our approach we are able to transform the layout of a short text document while keeping the textual content.

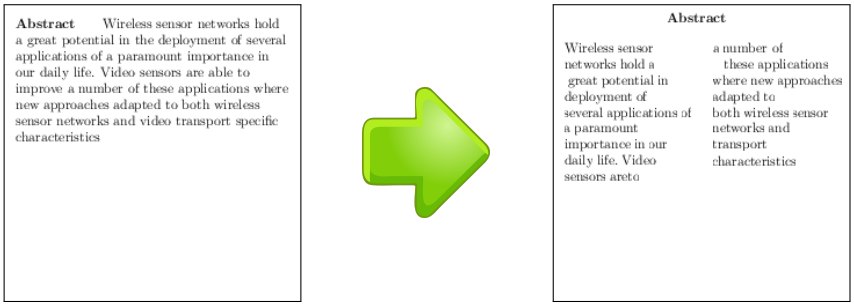


Figure 1: Exemplary style transformation learned by our model

1 Introduction

In recent years, supervised machine learning methods based on deep neural networks have outperformed previous state-of-the-art approaches in many computer vision tasks, such as image classification, object detection and semantic segmentation [1]. One key to success for most of these approaches is the ability to process vast amounts of data to train a statistical model. For common tasks like image classification, curated large datasets are publicly available [2, 3] and facilitate continuous research in these specific domains. However, publicly available data for many other tasks remains sparse. This poses a new challenge for using supervised learning methods, namely data augmentation. Basic techniques such as cropping, rotation, translation, color shifting, etc. lead to a considerable improvement in performance for some computer vision tasks, e.g. image classification [4], but for other tasks more advanced data augmentation methods are explored to get a statistically significant boost in performance. These augmentation methods have to be able to capture the semantics of the input space in order to make substantial yet still content-preserving transformations.

*authors contributed equally

In this project, we perform semantic data augmentation for the task of detecting and annotating the parts of images of text documents. Annotating text documents addresses the challenge of storing information in a universal and, most importantly, searchable data format instead of storing it as continuous and uncategorized text. There are efforts to build a document parser for any textual image [2] [5], such as scientific publications, the front page of a newspaper or an electricity bill. As state-of-the-art approaches for Optical Character Recognition (OCR) are impeccable at extracting text from a given area in an image, the challenge to solve is detecting the different types of elements in a text document, such as headers, paragraphs, tables, etc. Documents with detailed annotations of these type are currently not available in great quantity.

We believe that this dense object detection task [3] can significantly benefit from semantic data augmentation. Objects in text documents are not independent, meaning that the overall structure of the input image contains crucial information about each individual object. Applying classic data augmentation techniques would clearly lead to a loss of structural information. Alternatively, specifying rules for meaningful transformations can be tedious and cannot be easily scaled. Therefore it is desirable to learn such semantic transformations, ideally only using a small annotated dataset.

Semantic data augmentation techniques for producing more text documents for a document parser hence have to transform both the general outlook of a text document as well as the object annotations that correspond to it. We address this challenge by using a conditional Generative Adversarial Network (CGAN) [6] operating on slightly modified images of text documents.

General adversarial networks (GANs) [7] are a generative approach to Machine Learning that has gained a lot of interest over the last few years. The fundamental idea is that two neural networks, called *generator* and *discriminator*, are playing a two-player minimax game during training. The generator tries to learn a mapping from an arbitrary input distribution to some real-world space, e.g. photos. The generated outputs (called *fake*) are fed to the discriminator, alternating with random samples from a ground truth distribution, such as real pictures. The discriminator determines whether its input is real or fake, with the so-called adversarial loss being inferred from it (refer to Section 4.2). The weights of the generator and the discriminator are updated with gradient descent based on backpropagation. In a conditional GAN, the input distribution to the generator is a signal, e.g. from image or text, instead of random noise. The mapping learned by the generator can be used to carry out a style transformation on test data.

In this paper, we will outline some work that our efforts are based on, describe our full pipeline for this data augmentation task, present the performed experiments along with some visual exemplary results and finally point out the challenges and future work we believe to be of interest.

2 Related work

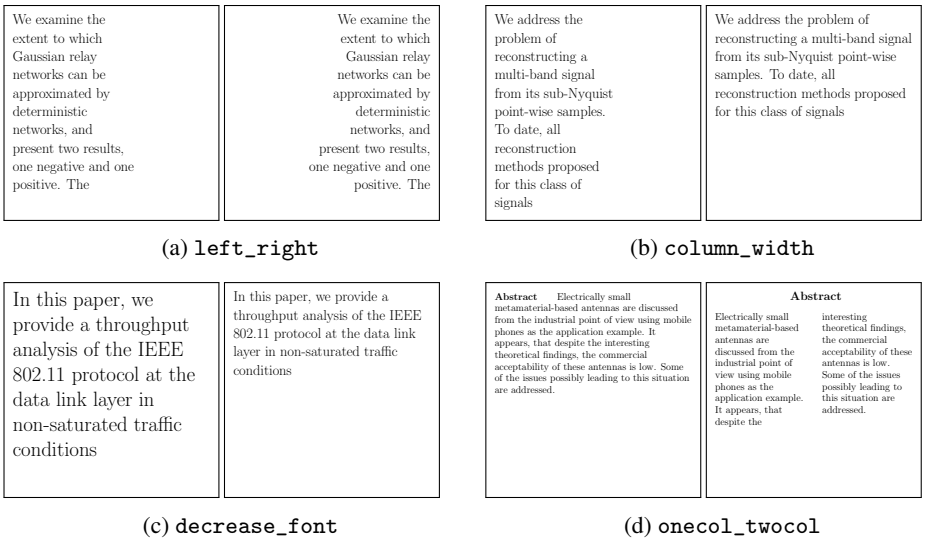
Style transfer Conditional Generative Adversarial Networks (CGAN) [6] have been successful in performing style transfer on natural images [8, 9]. While many approaches need paired training data, meaning a pair of images with identical content and two different styles, newer approaches have also achieved comparable performance for training on unpaired images [10]. A property of many style transfer applications is that the placement of objects is not changed. However, there are some applications of Conditional Adversarial Networks that don't keep the locality in image space, such as face rotation [11].

Semantic segmentation Deep learning approaches have started to dominate the field of semantic segmentation[4] and are state of the art for semantic segmentation [12]. A significant part of the approaches is based on Fully Convolutional Neural Networks (FCN), which have been shown to be robust for semantic segmentation by [13].

[2] e.g. in PDF format

[3] Dense object detection refers to considering all possible locations and possibly detecting a multitude of objects in an image

[4] also known as object detection, object segmentation, scene labeling



(a) `left_right` (b) `column_width`

(c) `decrease_font` (d) `onecol_twocol`

Figure 2: Example image pairs from the used datasets

3 Data

We have created several different datasets, each of them modeling a different transformation that is likely to occur when transforming a full-sized document, summarized in Table 1.

Table 1: Datasets that model exemplary transformations in document style transfer

name	description	image pairs (train)
`left_right`	Move text horizontally from left to right and change the alignment from left to right	1457
`column_width`	Increase the column width of a paragraph	1457
`decrease_font`	Decrease the font size in a paragraph and move the words accordingly	1457
`onecol_twocol`	Transform a one-column paragraph into two columns of text	6369

Visual examples are provided in figure 2. All of the datasets consist of paired images, meaning that there exist two corresponding images with different styles for each text snippet. The content of all documents was collected from papers available on `arXiv`.[5]. All text documents are of size 256x256 pixels and were generated using LaTeX.

4 Methodology

4.1 Preprocessing

As our goal is data augmentation for a supervised learning task, we want to transform both the actual content of the image as well as the semantic annotations of objects contained in the image. Instead of directly feeding images of text documents together with annotations in numerical format[6] to our model, we create a new image that encodes both properties. We use `pdfminer` [7] to extract a bounding box for each word in the text document and replace it with a rectangular bounding box in a unique color from a predefined set. An example of this can be seen in Figure 3. Furthermore, we assign one

[5] `arxiv.org`

[6] e.g. four integers to encode a bounding rectangle for an object: x- and y-coordinate of left upper corner, width, height

[7] https://github.com/pdfminer/pdfminer.six

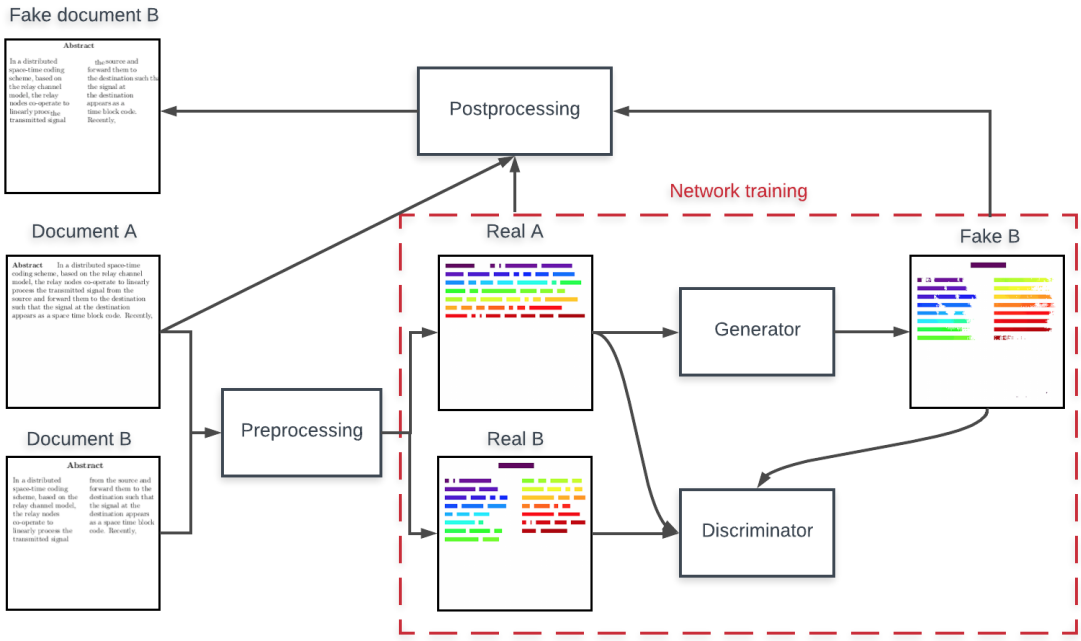


Figure 3: End-to-end pipeline for our approach to document style transfer

channel in the input layer of the neural network to each color, which allows us to operate in a discrete space instead of the continuous RGB color space.

4.2 General adversarial networks

The generator G in GANs learns an estimate $p_{\hat{y}}(\hat{y})$ of a probability distribution $p_y(y)$:

$$G : z \mapsto \hat{y} \quad .$$

Usually, z is noise sampled from a distribution $p_z(z)$. However, in a conditional GAN setting, we add a sample x from an input distribution $p_x(x)$ (called *style A*) as input, so the generator is

$$G : (x, z) \mapsto \hat{y} \quad .$$

We refer to $p_y(y)$ as *style B* (see Figure 3). The discriminator D either takes a pair (x, y) or a pair $(x, \hat{y})$ and outputs a scalar:

$$D : (x, \tilde{y}) \mapsto [0, 1] \quad ,$$

which is the probability that $\tilde{y}$ is a real sample from $p_y(y)$ and not a sample from $p_{\hat{y}}(\hat{y})$ generated by G. The discriminator is trained on maximizing the probability of assigning the correct label when given x and y. The generator in turn is trained to minimize the probability of the generated samples $\hat{y}$ being classified as fakes by D, i.e. minimizing $log(1 - D(x, G(x, z))$. The objective of the cGAN, the so-called adversarial loss, is then

$$\min_G \max_D \mathcal{L}_{cGAN} = \min_G \max_D \mathbb{E}_{x,y}[\log D(x, y) + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))] \tag{1}$$

In addition to the adversarial loss, we also impose an *image loss* between $\hat{y}$ and y to make the generator better. In image space, $L1$ loss is typically used instead of $L2$ loss because it has shown to create better and sharper images. We define this loss as

$$\mathcal{L}_{img}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \tag{2}$$

Therefore, our final objective is

$$\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{img}(G) \tag{3}$$

where λ is a weighting parameter.

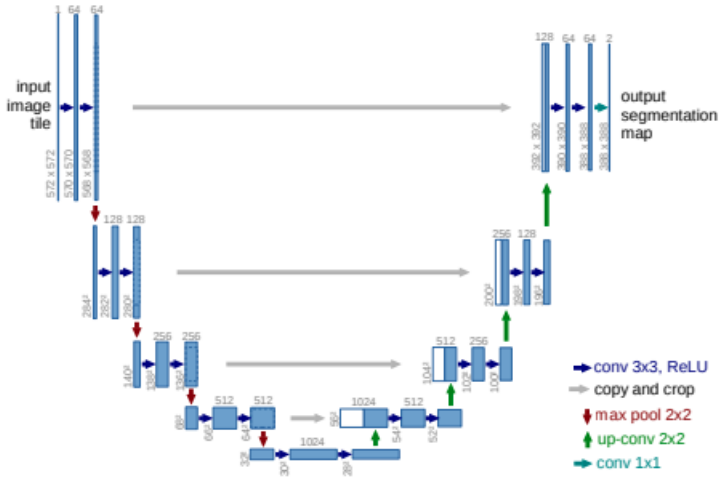


Figure 4: U-Net structure. Image from [14]

4.3 Network architecture

Our model is based on the `pix2pix` architecture [8], which has shown to be useful for image-to-image style transformation. Both the generator and the discriminator are based on convolutions and deconvolutions. Each layer of the generator and the discriminator consists of the same basic steps: (De)Convolution-BatchNorm-ReLU.

4.3.1 Generator

The architecture of the generator is a U-Net, introduced in [14]. The network has a "U"-like structure (see Figure 4) consisting of two parts, an encoder and a decoder. The encoder uses convolutions to map the original image to a compressed latent space. From this information bottleneck, the decoder uses deconvolutional layers to transform the latent representation back to the original shape of the input. Layers from the encoder and the decoder are connected in pairs with so-called skip connections to give the decoder the possibility to use features from all layers in the encoder, not only from the bottleneck layer.

4.3.2 Discriminator

The discriminator should be able to distinguish high-level structures that determine whether an image is real or fake. We use a patch-wise FCN [13], which has been shown to be a powerful discriminator in GANs and has fewer parameters compared to regular CNNs due to not having fully connected layers at all. We first concatenate a pair of images from style A and style B, where B is either real or fake. Then we pass it to the FCN. The discriminator's goal is to classify a single `NxN` patch either real or fake given the input image. We employ the generator on all patches of the input, using stride 2. The mean of all the outputs will be the final output of the discriminator.

4.4 Semantic Segmentation loss

Most style transfer approaches focus on learning a mapping from a typical RGB image of style A to another RGB image of style B. Most importantly, the location of objects in the image is not fundamentally changed. In that setting using an L1 loss as described in Equation 2 supports e.g. the desired shift in colors or change of texture quite well, as has been shown in [8]. Even though we use colors to enumerate all the boxes in an image, the continuity property of RGB space is not appropriate for this discrete coding scheme. We therefore change the image loss from equation 2 to the cross-entropy loss:

$$\mathcal{L}_{img}(G) = \mathcal{L}_{cross-entropy}(y, \hat{y}) = -\sum_{c} y_c \log(\hat{y}_c) \quad ,$$

where y is the ground truth and $\hat{y}$ is the estimate by the model and c is the number of output classes. This approach is inspired by semantic segmentation problems [12], where for each pixel the output is a probability distribution over all classes. Each channel represents either background or a specific word box.

4.4.1 Focal loss

Focal loss [15] was designed to tackle the problem of class imbalance between foreground and background and also amplifying the gradients caused by outputs of low confidence. Binary Focal loss extends the binary cross-entropy loss by adding a modulating factor $(1-p)^\gamma$ if $y = 1$ and p^γ otherwise. The binary focal loss is therefore defined as:

$$\mathcal{L}_{focal}(y, \hat{y}) = -(1-\hat{y})^\gamma y \log \hat{y} - \hat{y}^\gamma (1-y) \log(1-\hat{y}) \quad .$$

Focal loss can be easily extended to a multi-class setting and can then be used to replace the image loss from Equation 2:

$$\mathcal{L}_{img}(G) = \mathcal{L}_{focal_c}(G)$$

4.5 Postprocessing

The final task of our document style transfer pipeline is reinserting the text from the original image according to the generated color boxes. We use `opencv` [16] on the preprocessed colored images to get boxes containing one word each from the original text images. These words are then used to create a new image based on the output of the generator. We employ two different methods as explained below. For both of them, the first step is to create a binary mask[8] per output channel. This can be done once for each channel in the softmax outputs of the generator, which allows us to use a confidence threshold $\theta \in [0, 1]$ for creating the binary mask. Alternatively, we can use the actual output image and create a binary mask for each color that occurs in the image.[9]

Channel-wise median The median of the coordinates for all pixels that were marked in the indicator mask determines the center of a new word box. The median is statistically robust and enables us to deal with noisy outputs. This method is not able to adapt the size of a word.

Blob detection

1. **Channel-wise contour detection** We use `opencv` to detect the contours of each object (so-called blobs) in the indicator mask and embed it in a rectangular bounding box. We disregard bounding boxes with very few pixels and try to find one large or multiple big boxes that are located within a small area. These measures tolerate both single outliers as well as a slivered box. We use the left upper corner of the determined insertion rectangle as the reference point for word insertion.
2. **Zooming** We compare the height of the determined insertion rectangle to the height of the word to be inserted and scale the word accordingly.
3. **Horizontal word alignment** We condition the position of each word on words previously inserted on the same line to ensure a natural and regular horizontal spacing.

5 Experiments and results

To evaluate the generality and performance of our methods we test them on different datasets as summarized in Table 1. All experiments are performed on a single Nvidia TitanX GPU with 12GB of memory.

5.1 Evaluation metrics

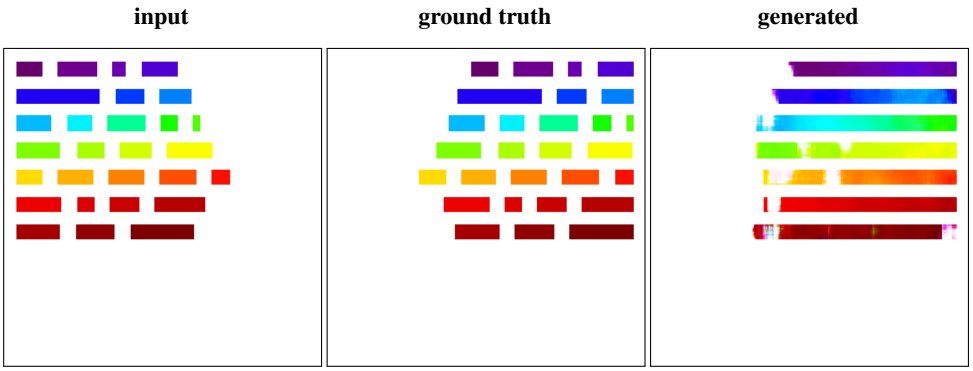
Current evaluation for synthesized images is either directly performed by humans[10] or with metrics that try to mimic human judgement (as summarized in [8]). The latter methods use standard pretrained

[8] size of the image, 256x256

[9] This corresponds to a pixel-wise argmax operation on the softmax outputs

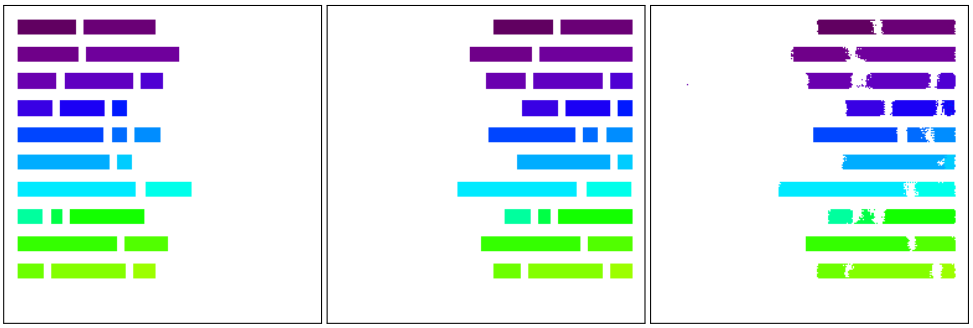
[10] on crowdsourcing platforms such as *Amazon Mechanical Turk*

neural networks for object recognition and score how well the objects in the synthesized images can be detected. In contrast to that, our experiments are not conducted on natural images. Therefore, using an evaluation metric based on out-of-the-box object recognition is not feasible. However, in contrast to natural images we are able to monitor the network's semantic segmentation loss (refer to Section 4.4) on the validation set (see Figure 10). As this is a discrete loss, we find that it can quantify the notion of structural correctness to a certain degree. On top of this indicator, we mostly have to rely on the human judgement of our collaborators.



(a) using L1 loss

(b) using Cross-entropy



(c) using Focal loss

Figure 5: Results of different loss functions on training data, with early stopping. The images for 5b and 5c were visualized by calculating the argmax of the softmax output for each pixel.

5.2 Optimization and regularization setup

For training the model, we follow the guidelines from [7]. We alternate between training G and D using minibatch stochastic gradient descent and apply the Adam optimizer [17] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Our initial learning rate is set at 0.0002 and starts to linearly decline at epoch 100 until it reaches 0.

For regularization, we apply dropout with a rate of 50%. Furthermore, we add a small random uniform noise to both x and y each time they are used in the training process. On top of the regularization effect, this also avoids only having discrete values in the dataset. Inputting ground truth images with discrete values to the discriminator is undesirable as fakes generally won't reach having discrete (one-hot encoded) outputs, which constitutes a straightforward possible decision criterion for the discriminator.

5.3 Comparing loss functions

Evaluations on the results when using the L1 loss on RGB images show line-to-line correspondence with ground truth, but the generated images converge into a continuous color spectrum (see 5a). Therefore, we don't have spaces between words and we lose a lot of desired structure in the output. Furthermore, it is not clear how to deal with colors that are not present in the input image, as mentioned in Section 4.4.

By reformulating the problem to a semantic segmentation problem we can notably improve our results, as depicted in Figure 5b. On top of line-to-line correspondence we can observe a clear notion of separated boxes, corresponding accurately to the ground truth. Altogether, this makes re-inserting the words in postprocessing straight-forward.

For Focal loss we cannot observe obvious improvements in Figure 5. However, extensive visual evaluation shows that the spacing between words is generally more accurate on the test data, the validation (semantic segmentation) loss slightly lower and the outputs contain less noise compared to using the Cross-entropy loss.

5.4 Visual results for multiple learned spatial translations

All results presented in Figures 5-9 are cherry-picked.

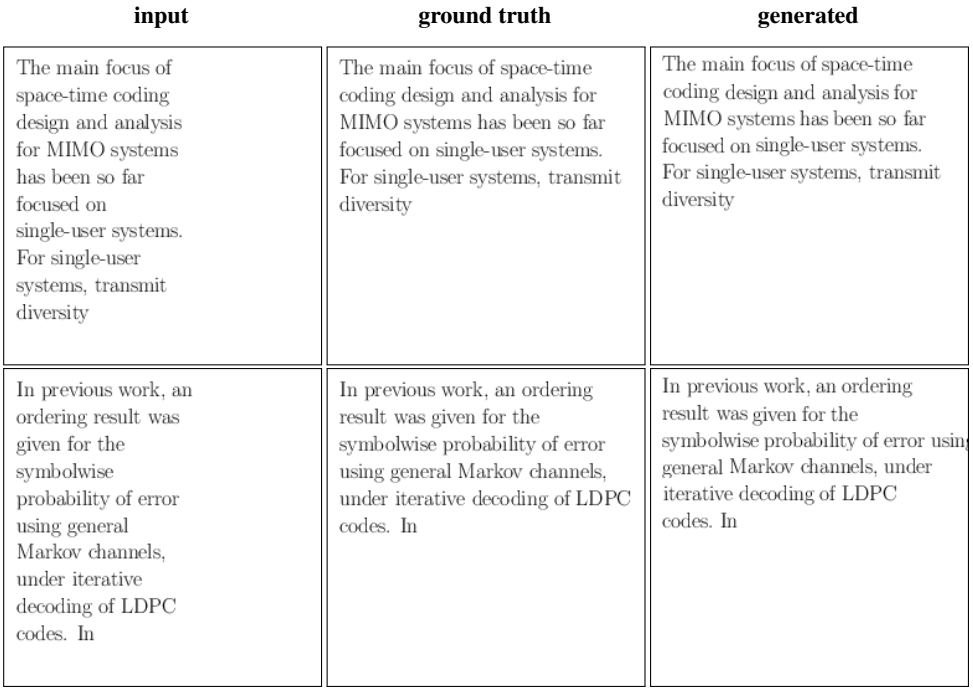


Figure 6: Test set results for the `column_width` experiment

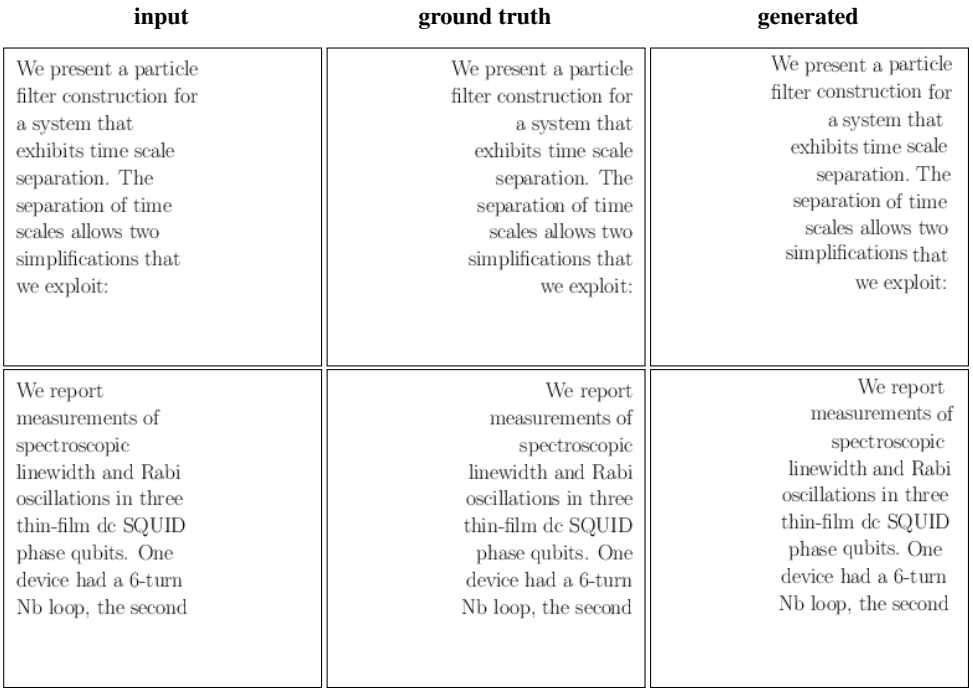


Figure 7: Test set results for the `left_right` experiment

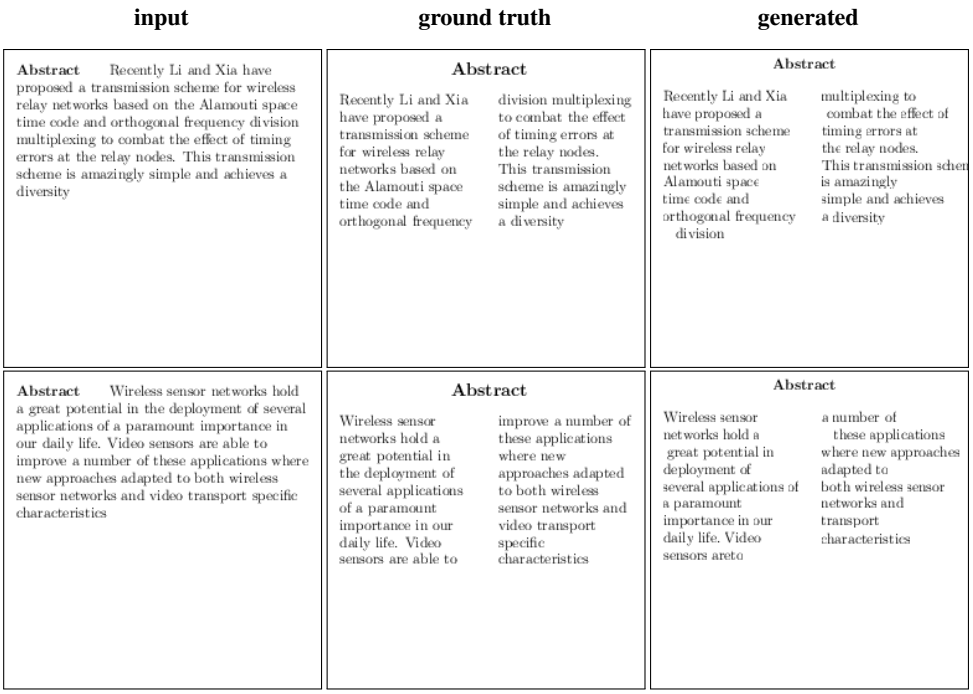


Figure 8: Test set results for the `onecol_twocol` experiment

5.5 Validation

We monitor the semantic segmentation loss $\mathcal{L}_{img}$ during training on a small validation set (see Figure 10). While we can observe a nice convergence for both training and validation semantic segmentation loss for simple transformations like `left_right`, we are prone to overfitting and have to do early stopping between epoch 50 and 100 for experiments on advanced transformations such as

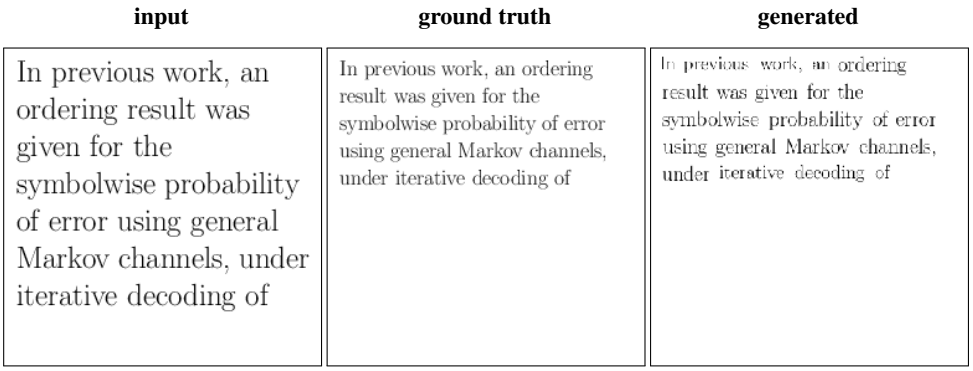
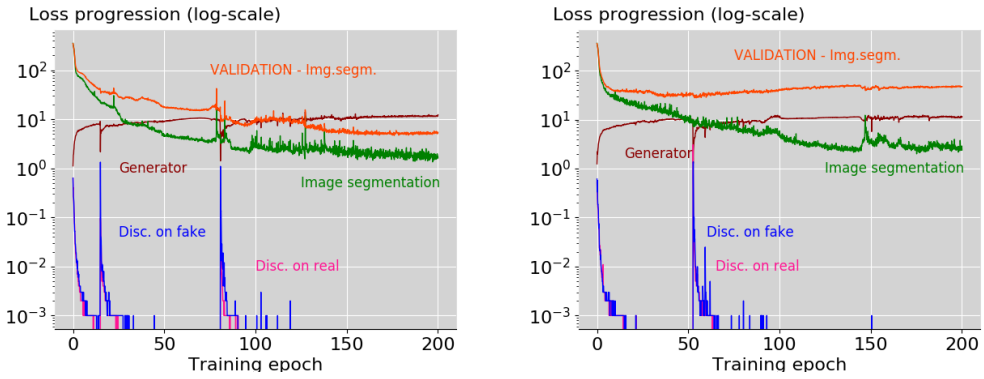
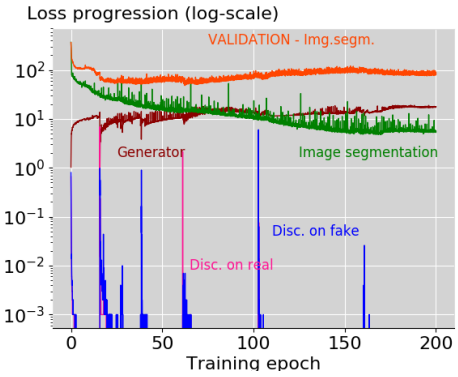


Figure 9: Test set results for the `decrease_font` experiment



(a) `left_right`

(b) `column_width`



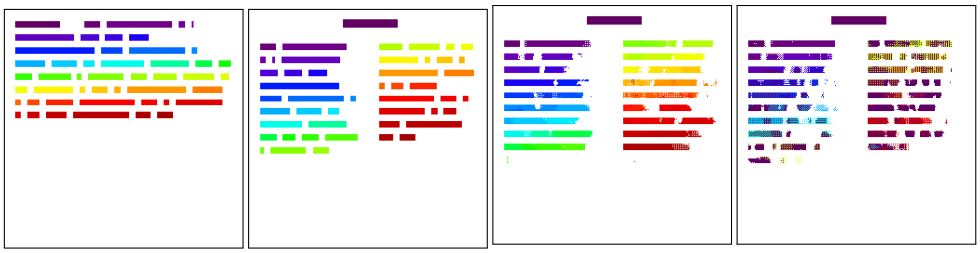
(c) `onecol_twocol`

Figure 10: Loss plots for different experiments

`onecol_twocol`. As shown in Table 1, we increase the size of the training set for `onecol_twocol` by more than a factor of four. This leads to a marginal improvement in validation loss and visual image quality on the test set. Regularization measures as discussed in Section 5.2 also lead to minor improvements.

5.6 Ablation study

Looking at Figure 10, we are suspicious about the impact of the adversarial loss, as it seems like the semantic segmentation loss is causing most of the gradients for the weight updates. As a consequence, we remove the adversarial loss (Equation 1) entirely and only keep the semantic segmentation loss (Equation2) to make sure that the discriminator is actually necessary to get good results.



input | ground truth | generated with full cGAN architecture | generated without adversarial loss

Figure 11: Comparison between results on training data after 100 epochs with and without using the adversarial loss

Figure 11 shows that the generator unable to produce interpretable results without the adversarial loss. The generated images suggest that it is possible to transform the overall structure, but an explicit spatial translation of words can generally not be achieved.

5.7 Postprocessing

Our method `channel-wise median` (see Section 4.5) is able to achieve satisfying results on easy tasks such as the `left_right` experiment (refer to Figure 12a). This basic method does not introduce any additional domain knowledge in postprocessing and underlines the power of our Machine Learning model.

However, we were not able to generate natural-looking text on harder tasks such as `column_width` with the `channel-wise median` method, as shown in Figure 12b. For these tasks, our method `blob detection` is necessary to produce pleasantly aligned text. We are aware that we have introduced additional domain knowledge in this method, but consider it rather minor, as it only adapts the horizontal alignment between words. The harder challenge, namely moving the word to a new approximate location within the document, is done by the cGAN.

In many postprocesses documents a few words from the input do not appear, because the only blob of the respective color is too small or the color is not present at all in the generated image. By operating on full softmax outputs instead of argmax outputs and adjusting the threshold θ for creating the channel-wise binary masks, we are able to make most words appear in postprocessing. This comes with the trade-off of being more sensitive to noise. The empirically determined value of θ differs from experiment to experiment and lies in the range of 0.01 to 0.5.

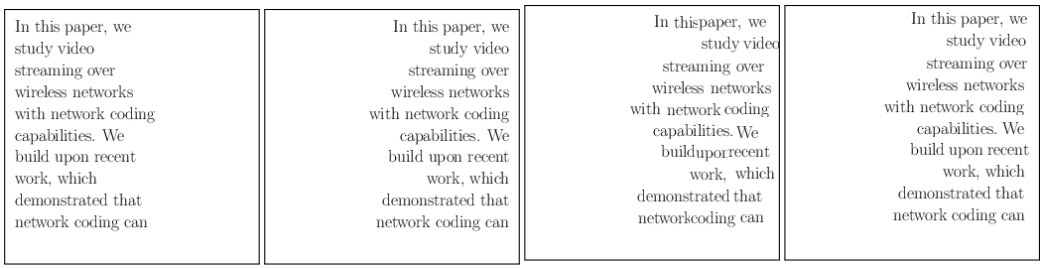
5.8 Unpaired data

We have conducted efforts to port our approach to a cGAN architecture that works on unpaired data as proposed in [10], but could not achieve results comparable to the quality of the paired-data setting. The generator was not able to produce real-looking images without the explicit support of the semantic segmentation loss, which can only be employed if paired data is available. Simply strengthening the generator and weakening the discriminator by adapting their respective depth did not improve our results.

6 Conclusions and future work

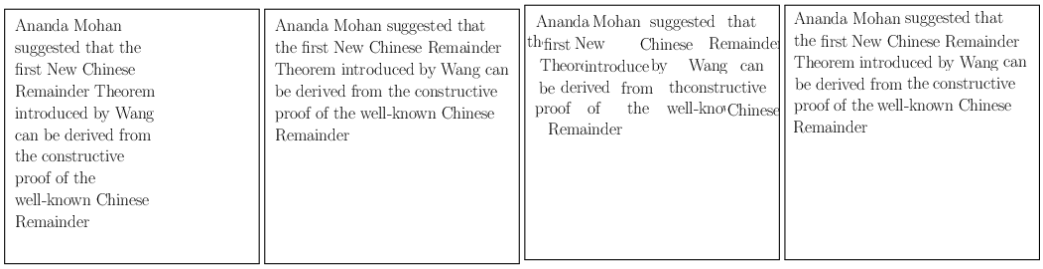
In this project, we have shown that conditional Generative Adversarial Networks are able to perform spatial translations of objects, which to the best of our knowledge has not been shown in the literature up to this point. Using learned spatial translations, we are able to perform a variety of style transfers on small documents with up to 50 words. Our approach currently has two major limitations:

- We are relying on paired training data, which is often unavailable in real scenarios where data augmentation would be beneficial. Making the generator more powerful and the discriminator weaker at the same time seems to be a straight-forward approach that should be investigated further.



input | ground truth | generated PP with medians | generated PP with blob detection

(a) on `left_right` experiment, 150 training epochs



input | ground truth | generated PP with medians | generated PP with blob detection

(b) on `column_width` experiment, 150 training epochs

Figure 12: Comparison of post-processing methods (on test data)

- The results shown in this article have not been scaled up yet to a full-size document, e.g. an A4 page. It is not clear whether the same neural network architecture would be able to handle hundreds of objects /output classes and a much larger resolution for input pictures. With adequate hardware, this could be investigated further.

A great difficulty in this project was to make GAN training more stable, which is still an ongoing research direction. We have achieved an acceptable balance between the magnitude and progression of the different loss functions but see much more potential, especially for unpaired data, if the progression of the adversarial loss could be improved.

While iterating to improve our approach, we noticed the immediate need for a more fine-grain evaluation metric for synthesized images of the transformed structure. We believe that an approach relying on scene recognition could be promising.

Acknowledgments

We would like to express our gratitude to Johannes Rausch and Ce Zhang for their helpful advice and ongoing discussions throughout this project.

References

[1] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[2] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[3] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[4] Luis Perez and Jason Wang. “The effectiveness of data augmentation in image classification using deep learning”. In: *arXiv preprint arXiv:1712.04621* (2017).

[5] Peter WJ Staar et al. “Corpus Conversion Service: A machine learning platform to ingest documents at scale”. In: *arXiv preprint arXiv:1806.02284* (2018).

[6] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[7] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[8] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *arXiv preprint* (2017).

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. “Image style transfer using convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2414–2423.

[10] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *arXiv preprint* (2017).

[11] Rui Huang et al. “Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis”. In: *arXiv preprint arXiv:1704.04086* (2017).

[12] Alberto Garcia-Garcia et al. “A review on deep learning techniques applied to semantic segmentation”. In: *arXiv preprint arXiv:1704.06857* (2017).

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[15] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *arXiv preprint arXiv:1708.02002* (2017).

[16] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[17] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).