text-deep-fake

Supervisor: Renat Khizbullin

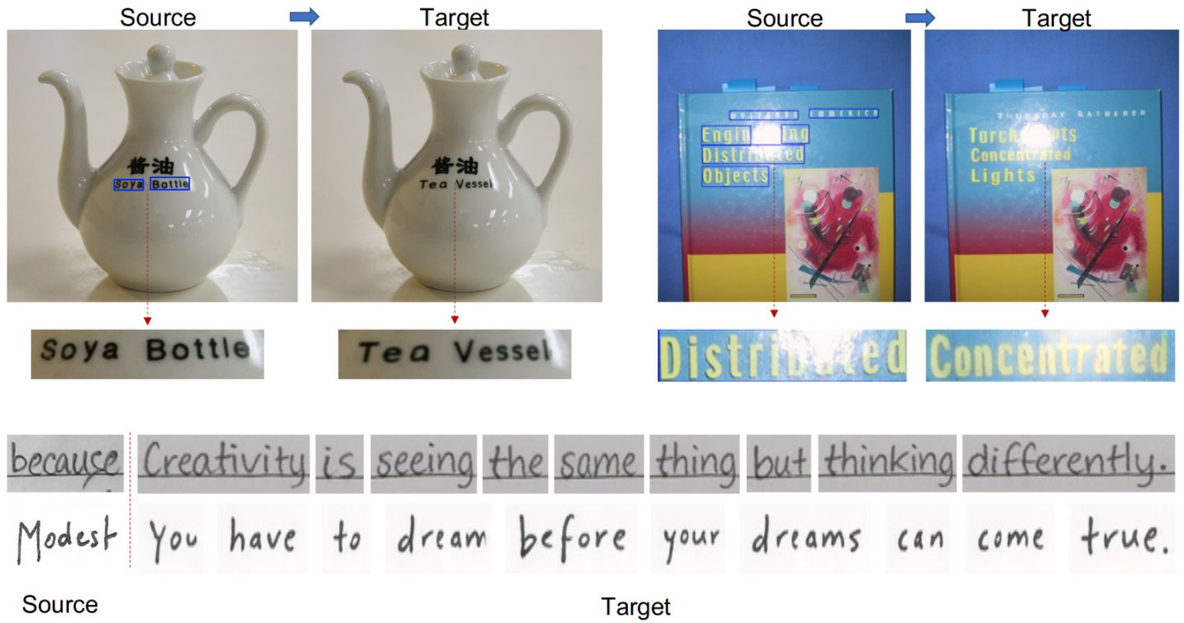
Authors: Alexander Bogdanov

Nikita Koryagin

Olga Kozlova

Problem Statement

- Change text on the picture and transfer original style



Source
Target
Source
Target
Soya Bottle
Tea Vessel
Distributed
Concentrated
because
Creativity is seeing the same thing but thinking differently.
Modest
You have to dream before your dreams can come true.
Source
Target

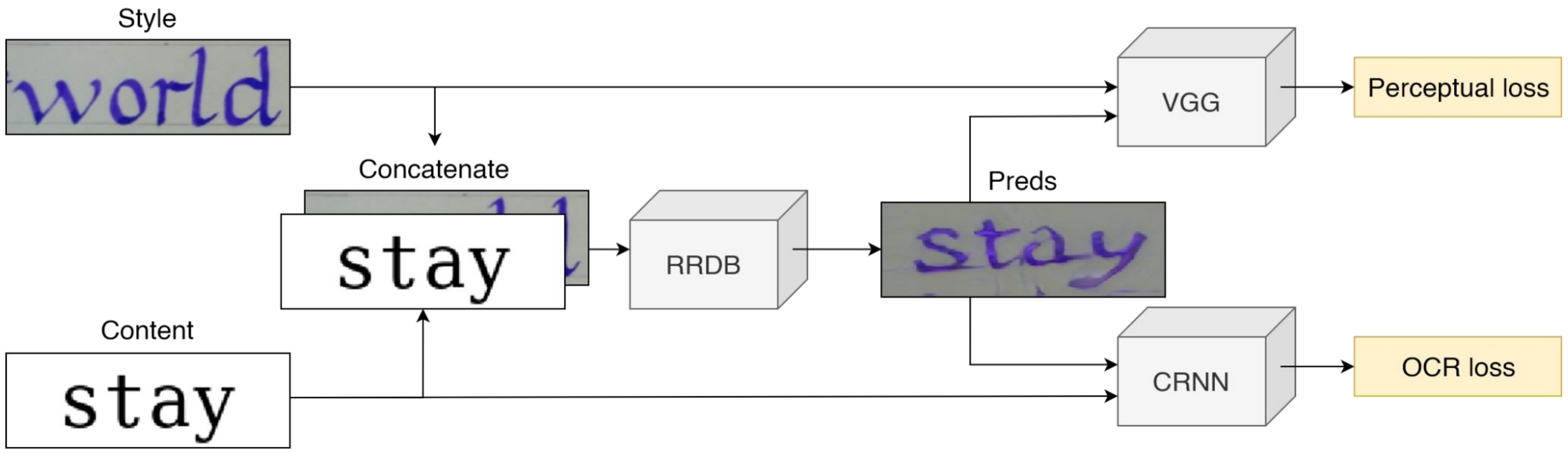
Image from TextStyleBrush paper

Goals

- Repeat the development path of the TextStyleBrush
- Reproduce results
- Publish open-source code

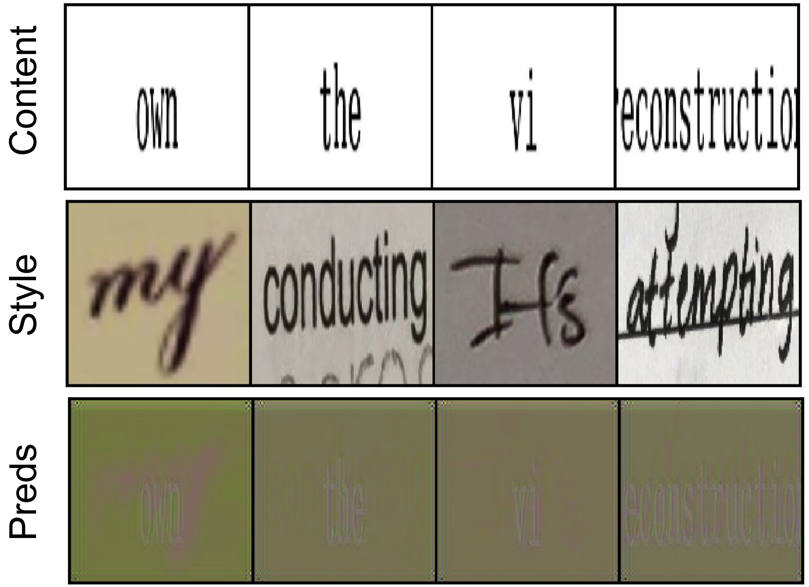
Baseline

Architecture



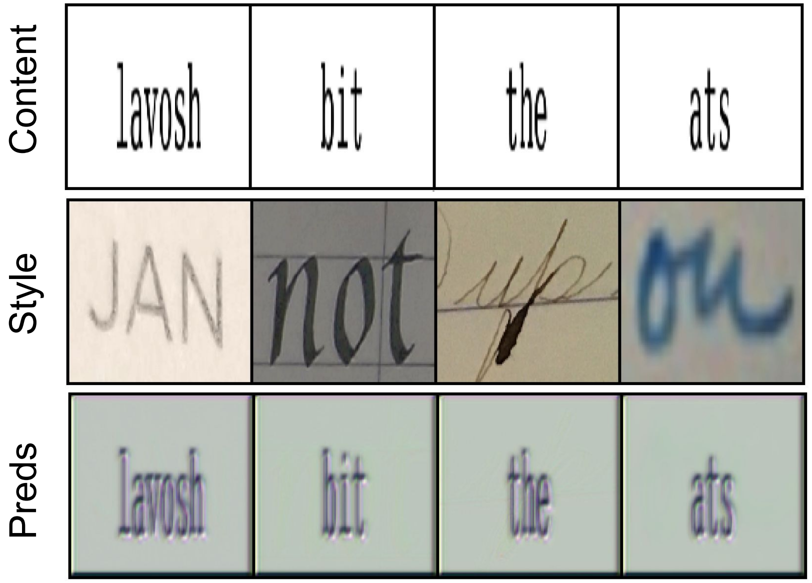
Baseline results

- Trained model on IMGUR5K dataset
- Output of the model is an overlay of the input pics
- We attempted to use different hyperparameters – no success



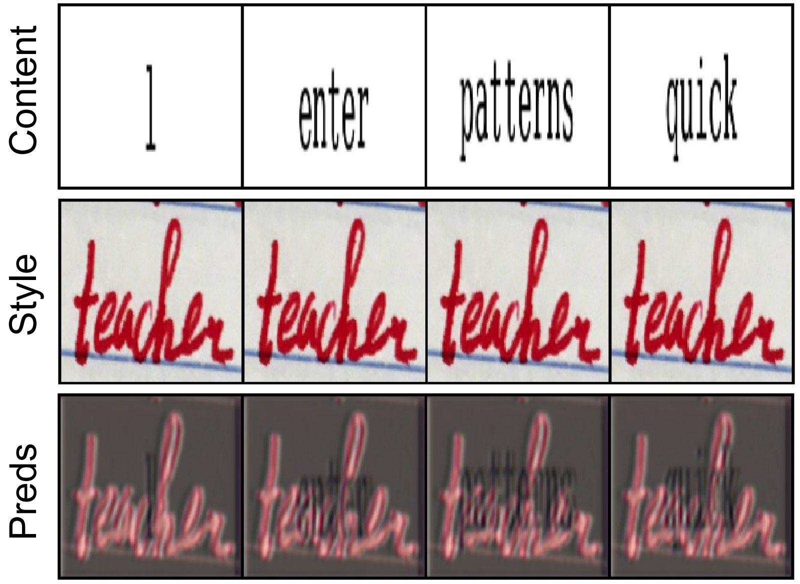
Use L1 loss instead of OCR loss

- We assume that the cause of bad results was a bug in OCR loss implementation – that's why we changed it to L1 loss
- Outcome is almost equal to content image except for the background color



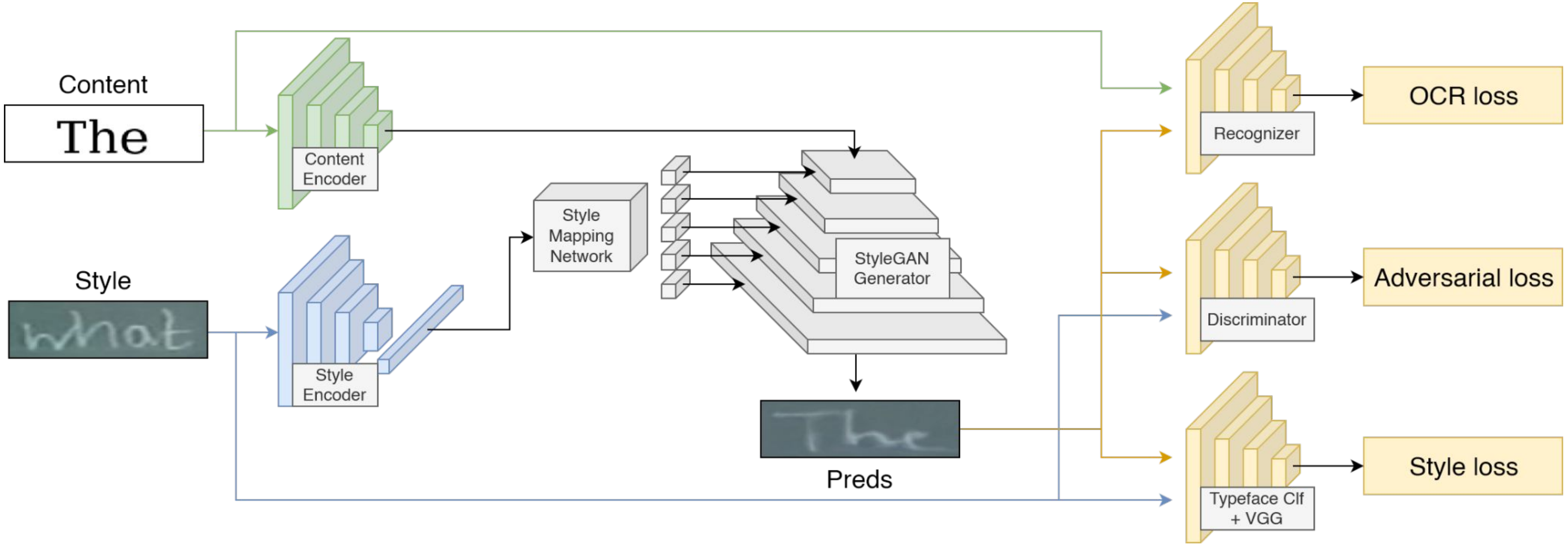
Only one style transfer

- Simplify the task to transfer only one style instead of all of them
- Result is still just an overlay of the input pics



StyleGAN generator

Architecture

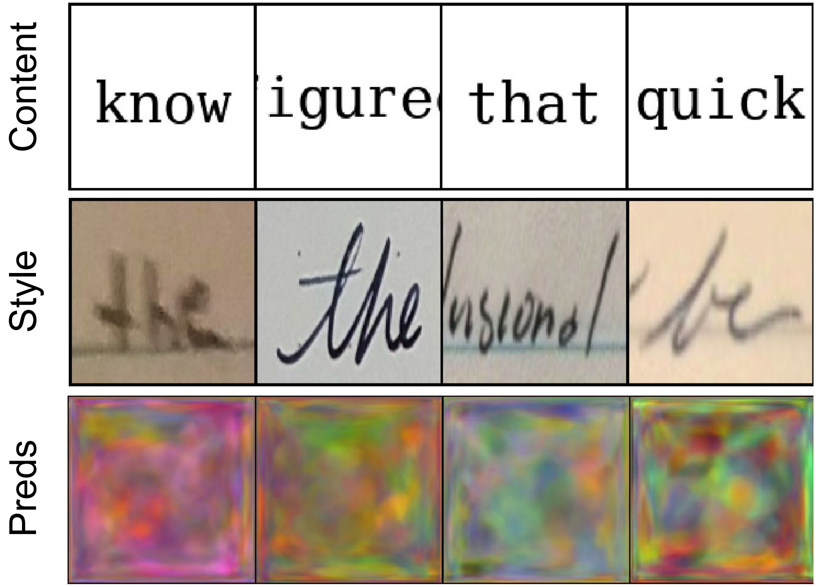


Implementation and first runs

- Optimize OCR and style loss at the same time
- As a variation of style loss we used gram loss:

$$L_{gram}(y, y') = ||G_j(y) - G_j(y')||$$

- Result picture is messy: neither content nor style are transfered



Typeface classifier

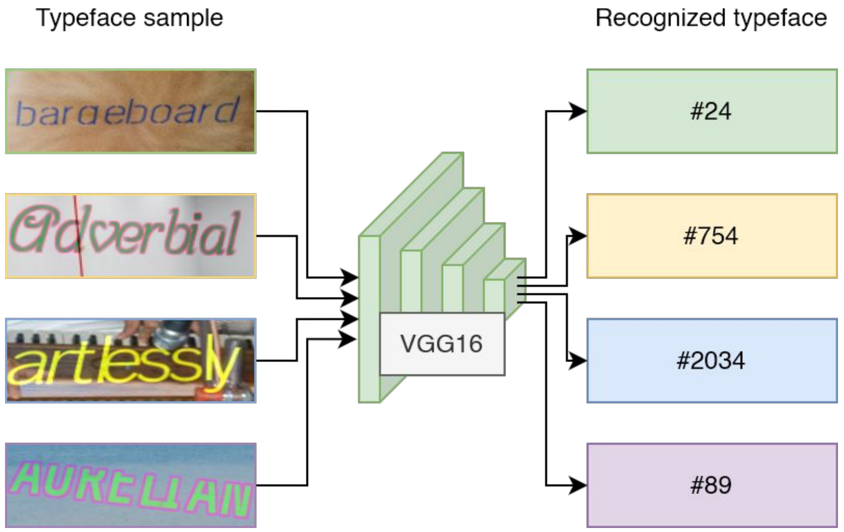
In the original paper style loss consisted of gram loss, perceptual loss and typeface loss. Some info about typeface loss:

- We generated synthetic images with various fonts

- 2500 random fonts

- We trained VGG16 to classify fonts

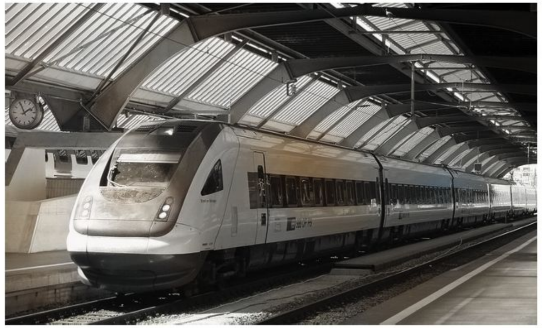
- Calculate L1 loss between embeddings of last layers of the trained model



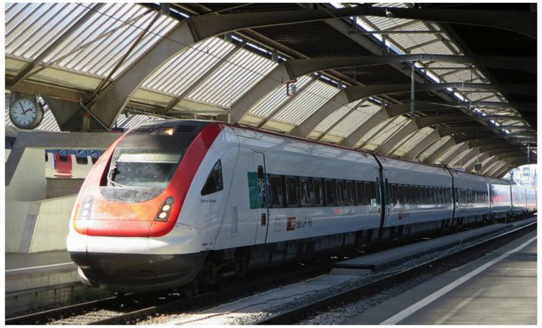
Adversarial loss

- Important part of the method we research is adversarial loss function
- To test the correctness of the implementation we added it to some simple and clear task – colorization

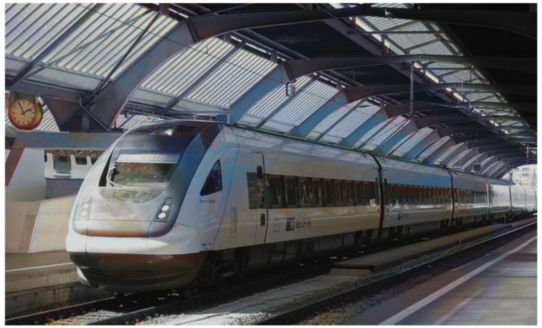
Result without adversarial loss



Original colored picture



Result with adversarial loss



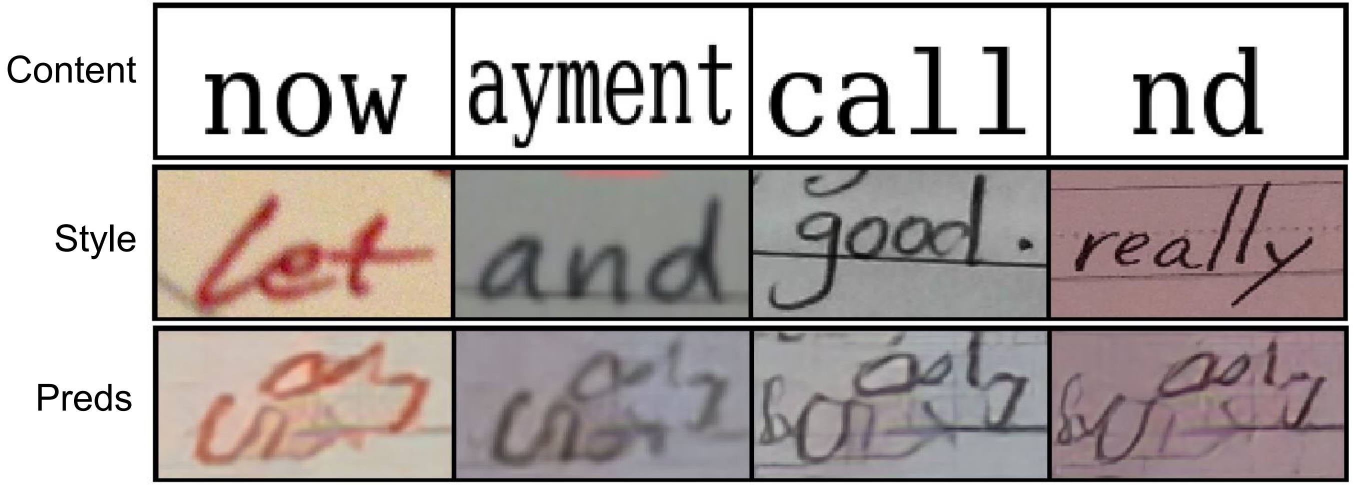
Single loss

Concerns

- Previous results were awful, so we decided to investigate loss functions and model’s architecture
- We started from loss functions: in next experiments we optimized only one loss function and observed results

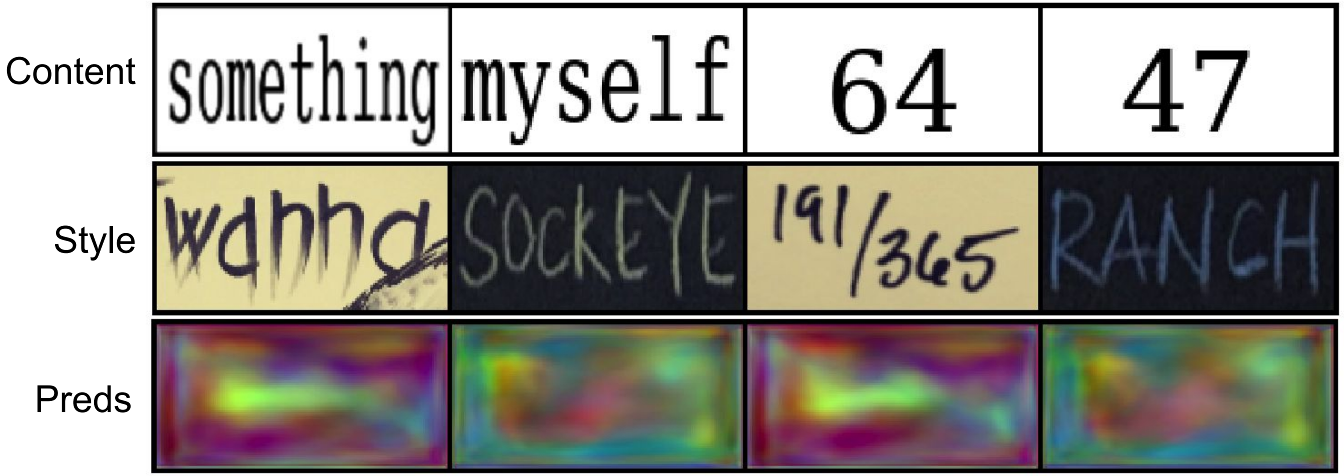
Style loss

- Optimize only style loss
- In results we saw that style was transferred: same color of text and background, same width of lines



OCR loss

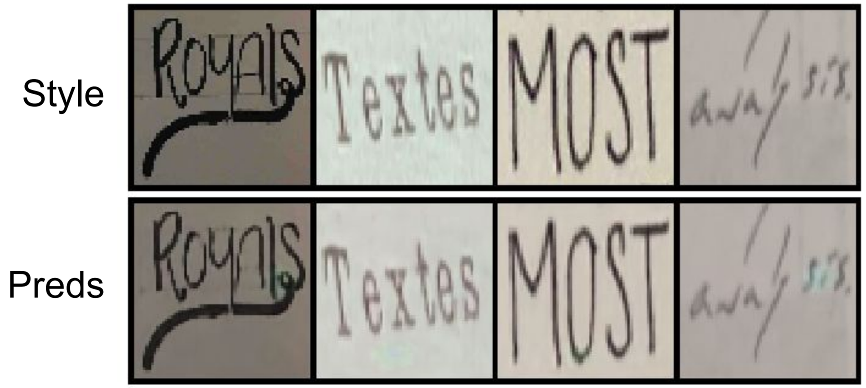
- Optimize only OCR loss
- In results pictures are messed again
- Optimization of OCR loss is useless ⇒ loss implementation or training process contains bugs



Autoencoders

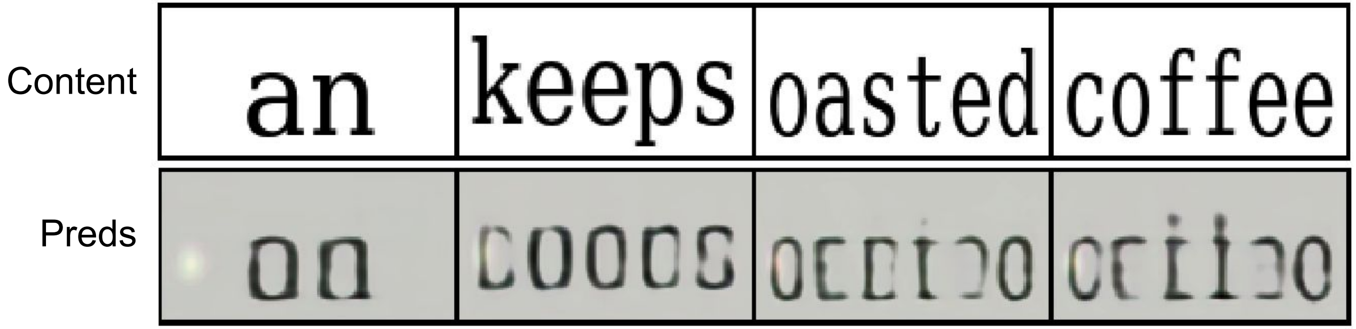
Style autoencoder

- To check our implementation of model with StyleGan architecture, we trained it as an autoencoder. If this architecture is not able to restore original input images, then information does not clearly affects output.
- We trained StyleGan-based model as a style autoencoder, optimizing L1-loss between model’s output and input style image
- Model is fully able to restore input style



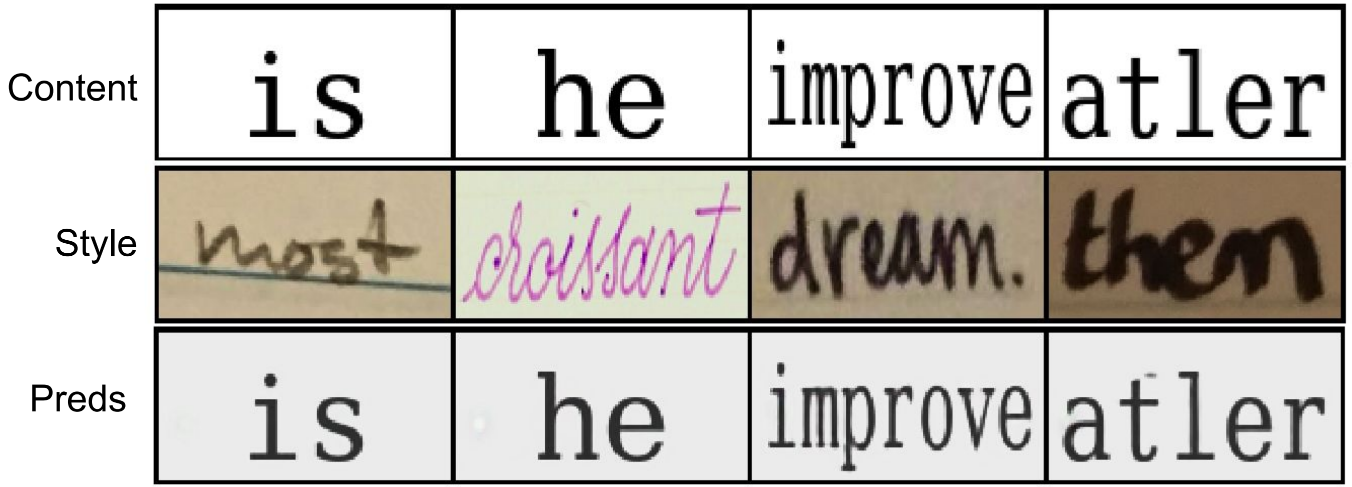
Content autoencoder

- We trained our model with StyleGan as a content autoencoder, optimizing L1 loss between the output and the input content image
- This experiment demonstrates that the model is not able to restore the content image, therefore the information about the content image does not clearly affect the output



Content autoencoder

- To fix this problem we modified StyleGan by removing noise
- As a result, model perfectly restored content



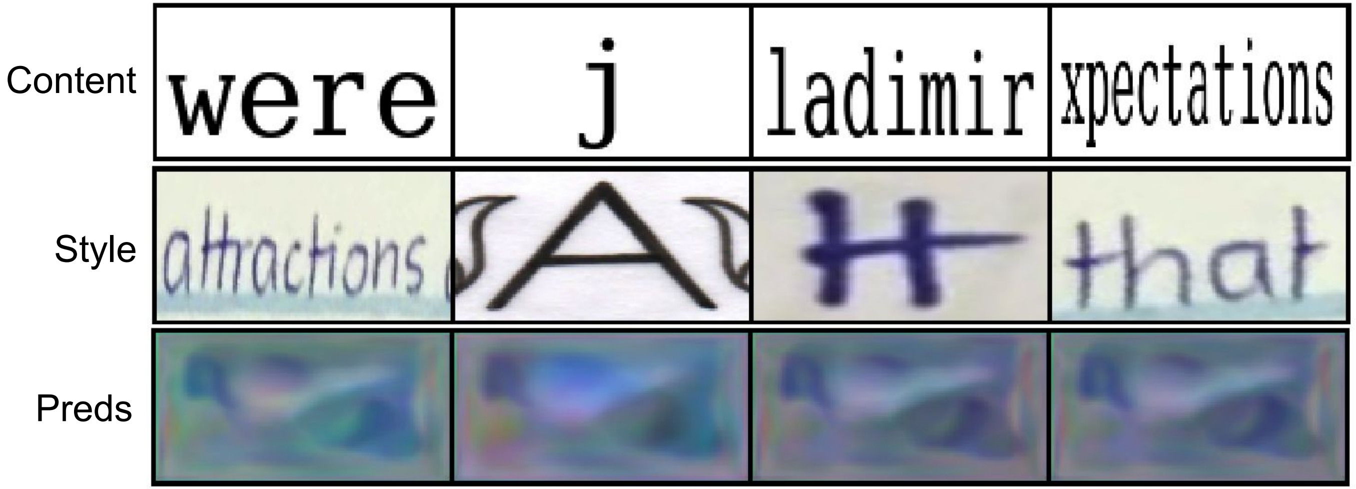
Stabilization of training

Replacement of OCR loss

- To make train process more stable we replaced OCR and CTC loss with pretrained TRBA and Cross Entropy correspondingly
- Expected advantages:
    - Better at dealing with text distortion thanks to dedicated block
    - Train dataset is bigger and more diverse because training was conducted
    - with augmentations and pseudo-patterning

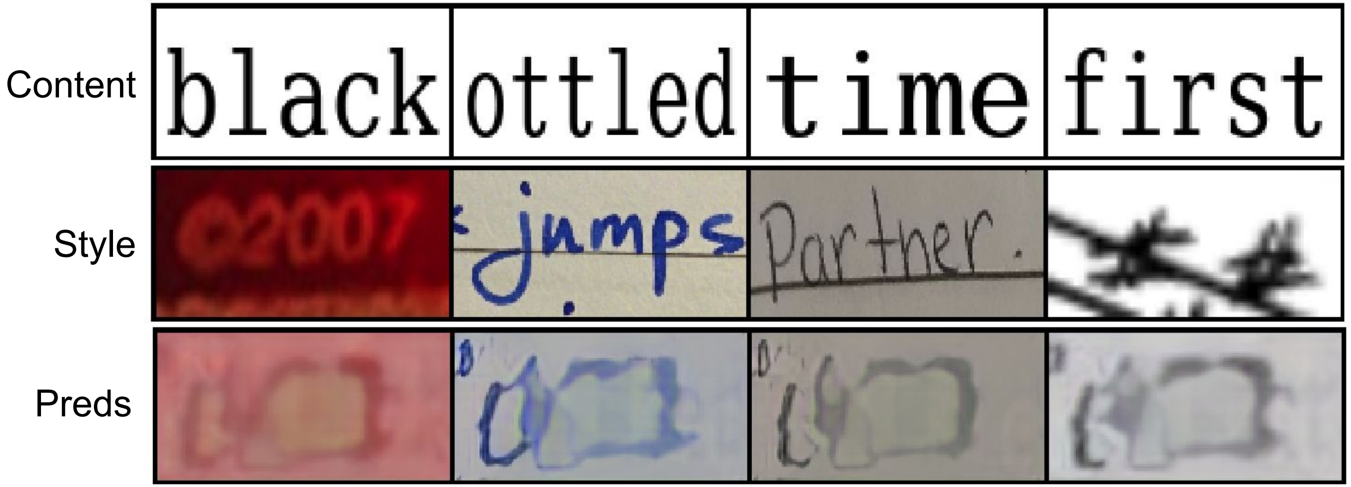
Simultaneous training

- We trained the network to optimize both loss functions (style and OCR) simultaneously
- Results are still not interpretable – neither style, nor content are transferred



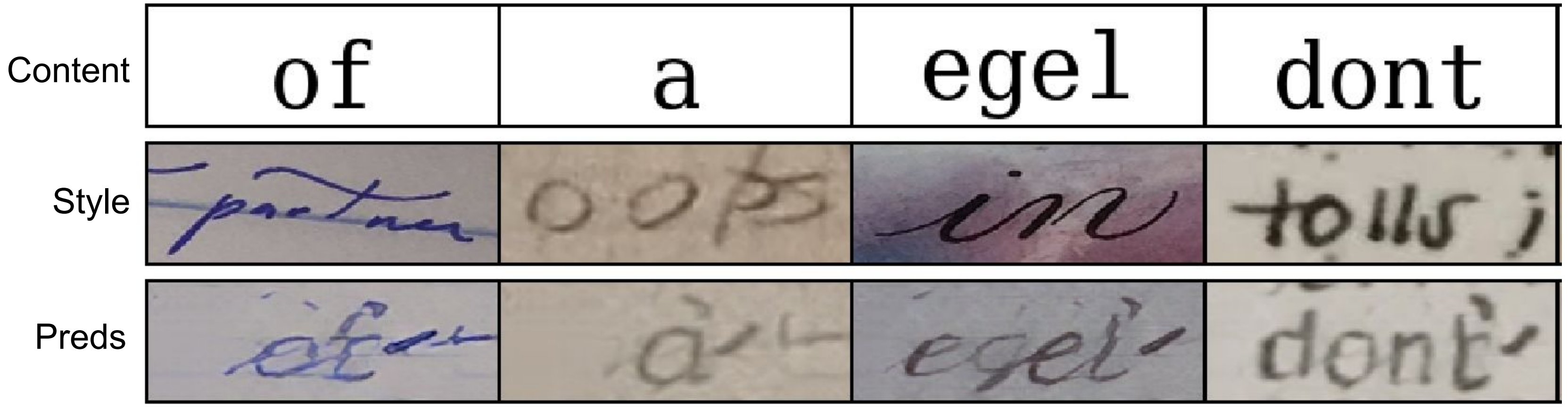
Sequential training – style first

- We decided to modify train process - in the beginning we trained style autoencoder (slide 19), then we continue training optimizing both loss functions
- In results pretrained style is blurred



Sequential training – content first

- As we did in previous experiment we trained content autoencoder first and then continue training optimizing both loss functions
- Finally suitable results – style and content are transferred



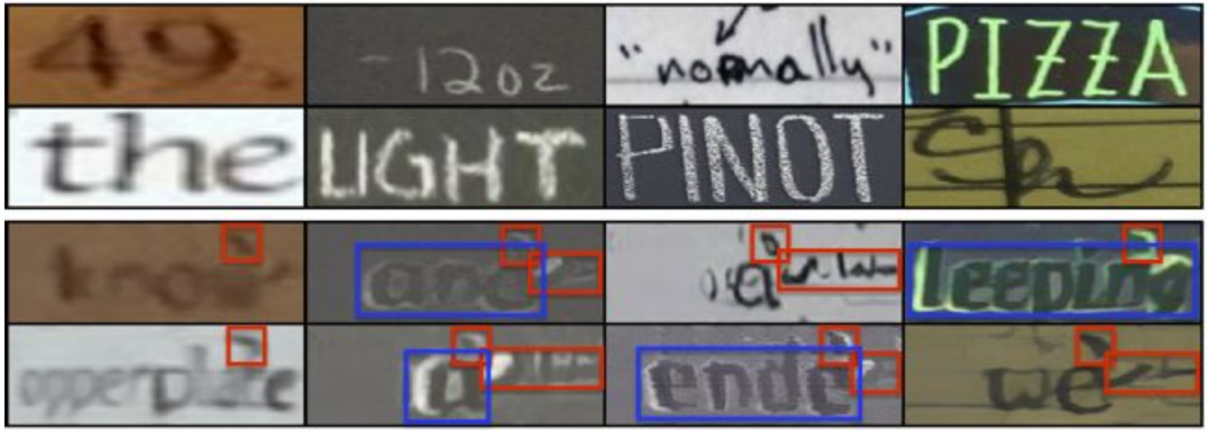
Tuning

Bugs

- We succeed in style transfer, but the quality is low
- Red frames – artefacts made by network
- Blue frames – text generated with the wrong color and extra text border in cases of bright text on the style picture

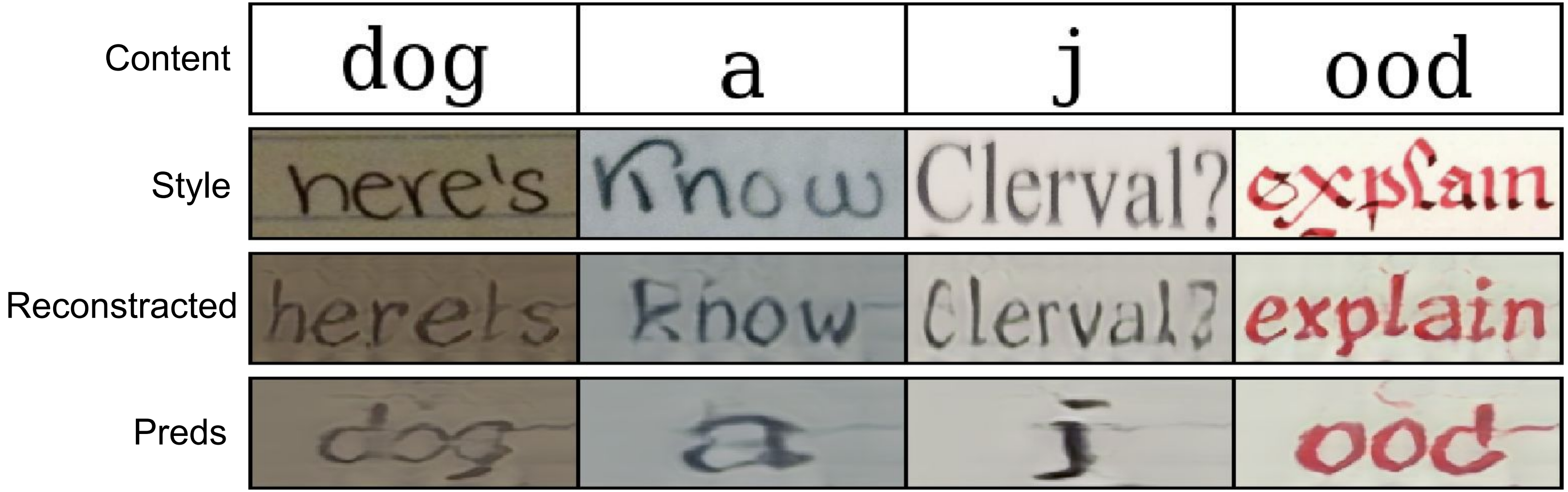
Style

Preds

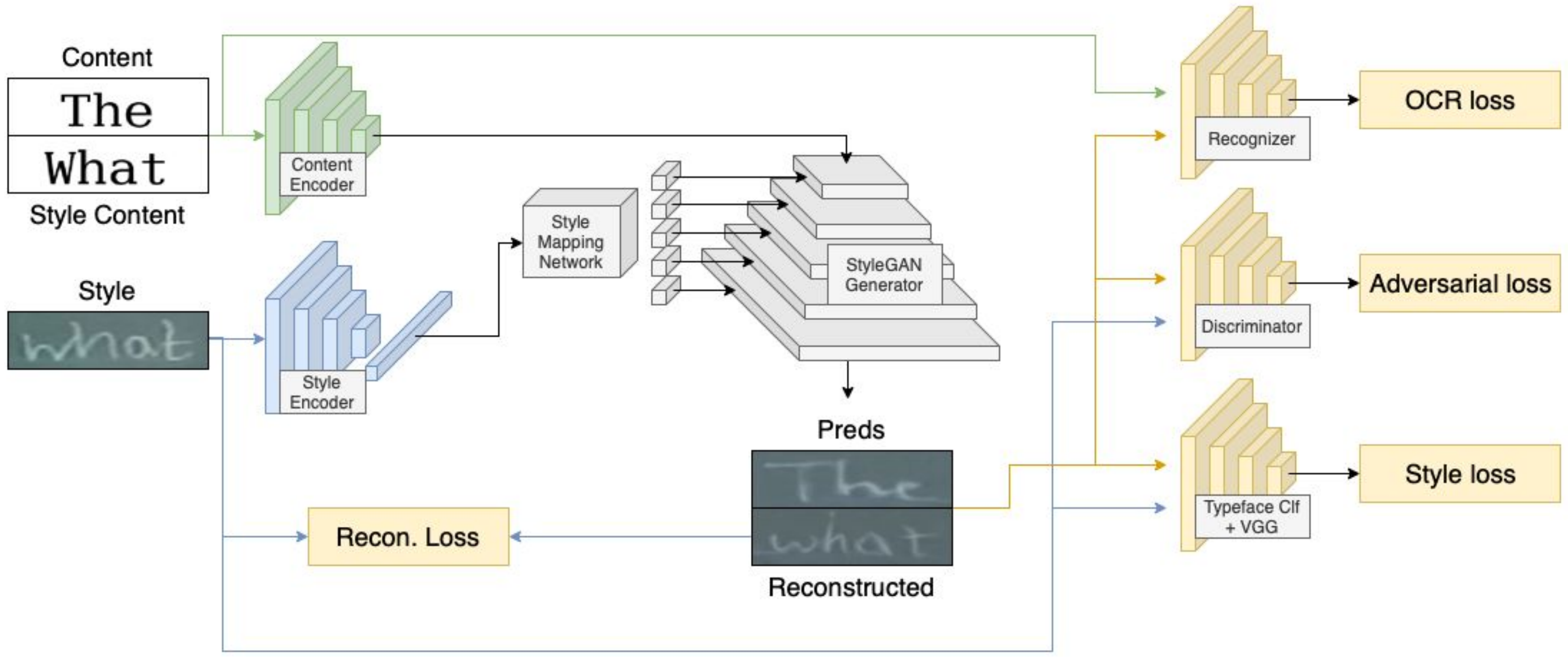


Artefacts

- To make train process more stable and get rid of artefacts we add reconstruction and cycle loss
- Result pictures almost don’t contain artefacts

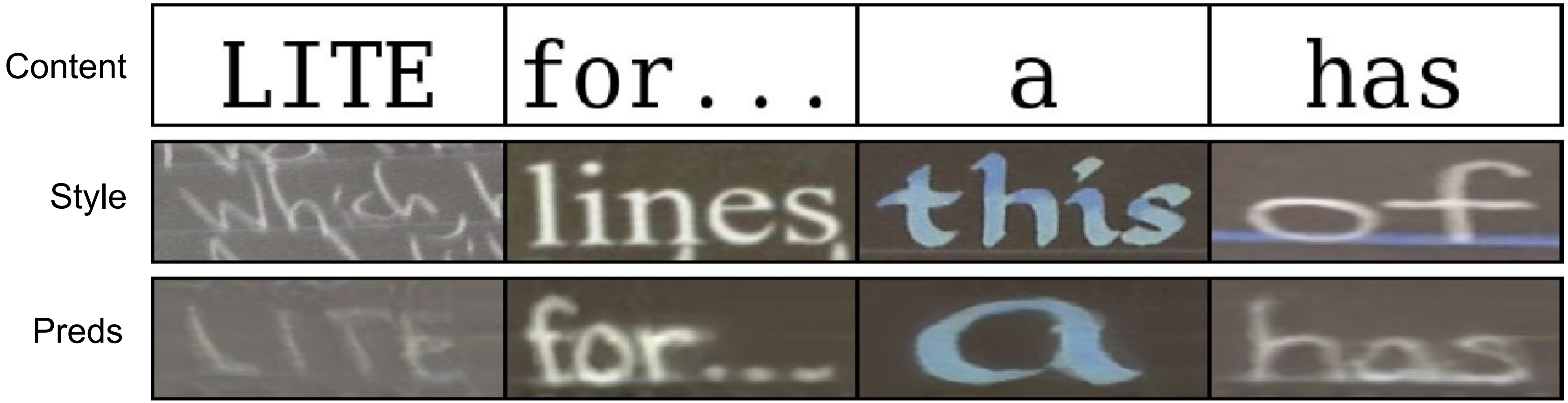


Final architecture



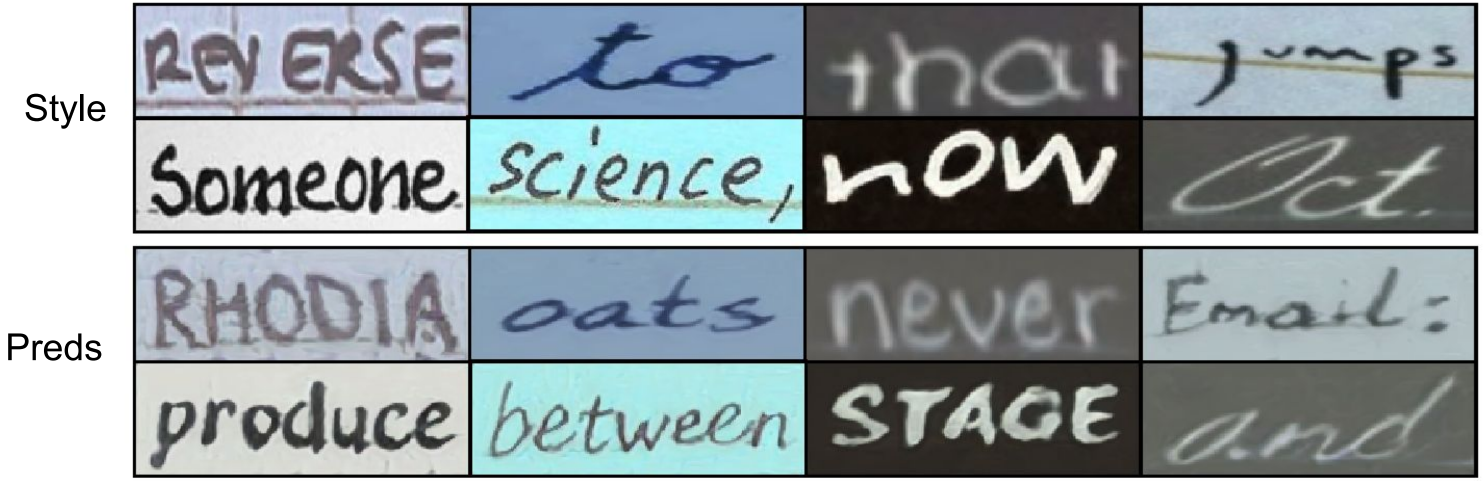
Light font issues

- In train data most of the pictures contains dark text on bright background, that’s why network prefers to generate dark text
- To decrease disbalance we invert some pictures in train process



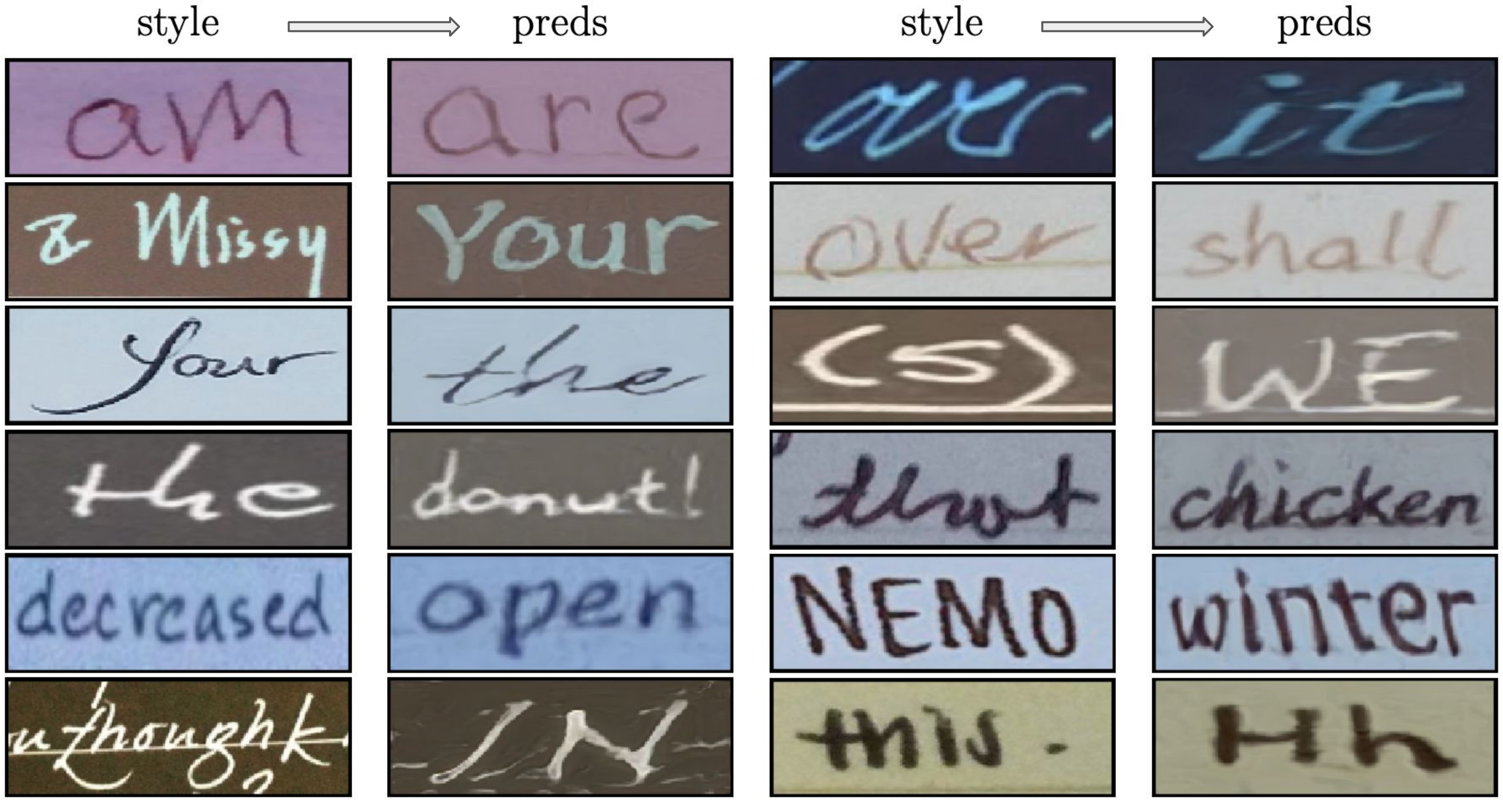
Состязательная функция потерь

- Add adversarial loss to make output pictures more real
- As a result, pictures became sharper

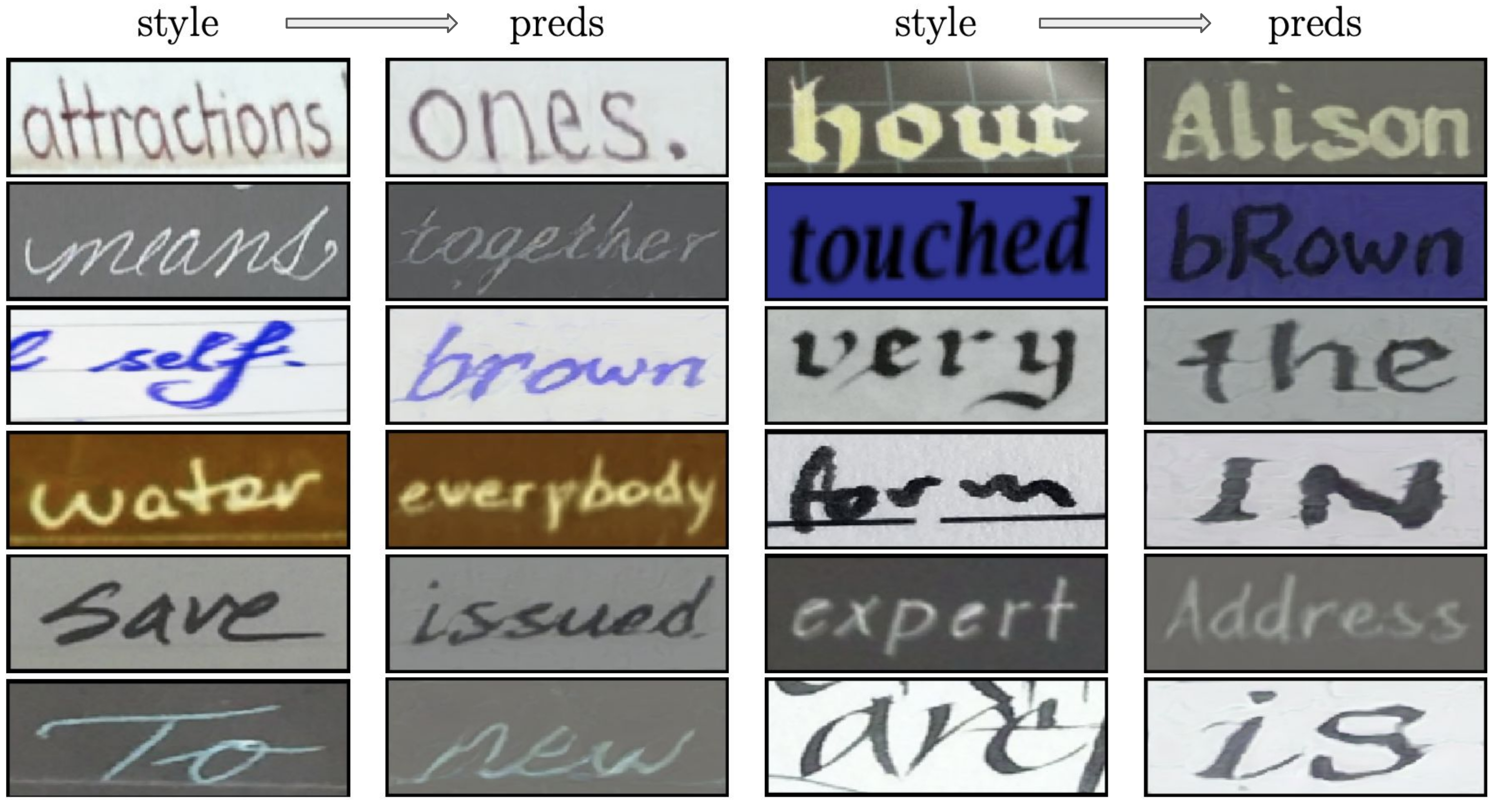


Conclusion

Results


style
preds
style
preds
am
are
& Missy
Your
your
the
the
donut!
decreased
open
thought
IN
it
over
shall
(s)
WE
that
chicken
NEMO
winter
this .
Hh

Results


style
preds
style
preds
attractions
ones.
hour
Alison
means
together
touched
bRown
self
brown
very
the
water
everybody
form
IN
save
issued
expert
Address
To
new
are
is

Conclusion

- We successfully implemented algorithm for realistic style transfer
- Implemented main parts of TextStyleBrush paper
- Published our results and code on Github

The End

Me: *uses machine learning*

Machine: *learns*

Me: