# Image to LaTeX

**DE-LE-PE Team:**
Antonina Kurdyukova
Dmitrii Baluev
Matvey Skripkin
Nikita Ligostaev
Nikolay Kalmykov

**Deep Learning 2023**

# Problem Statement

implement Deep Learning approaches for the problem of image-to-markup generation (image math equation to LaTeX code).

$$\mathcal{L} = \bar{\psi}(i\gamma^{\mu}D_{\mu} - m)\psi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu},$$

$$\downarrow$$

```
${\mathcal {L}}={\bar {\psi }}(i\gamma ^{\mu }D_{\mu }-m)\psi -{\frac
{1}{4}}F_{\mu \nu }F^{\mu \nu}$
```

# Motivation

Solutions already exists, but with ”free” limitations.

– **Mathpix:** 20 snips are free, then 5$/month;

– **yhshin/latex-ocr**: slow processing (up to minutes or errors), and low quality)

$$g(\mathbf{x}) = \det\ \nabla_{\mathbf{x}} \boldsymbol{\mathcal{F}}(\mathbf{x}, \lambda) = 0.$$

$$g() = \mathrm{dec\,et} \nabla_{\times} \mathcal{P}\,(x, \lambda) = 0.$$

$$\mathcal{L} = \bar{\psi}(i\gamma^\mu D_\mu - m)\psi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu},$$

$$=$$

$$vec\psi\,(\dot{\nu}\gamma^\mu \mathcal{D}_\ell -)\,\psi - \tfrac{1}{4}\overset{F}{\omega}\,F$$

[1] https://mathpix.com/
[2] https://huggingface.co/spaces/yhshin/latex-ocr

# General Description

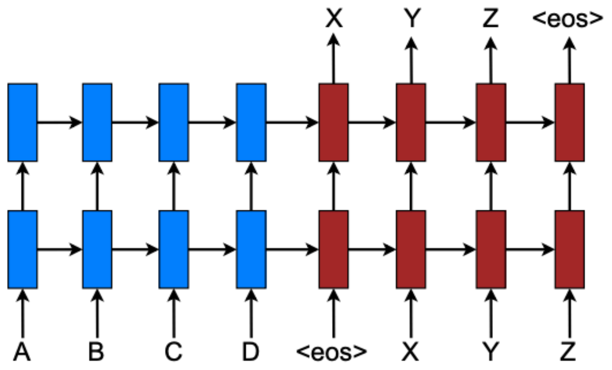
Image to LaTeX problem is basically image captioning technique that involves:

– **looking at an image** → Computer Vision (CV)

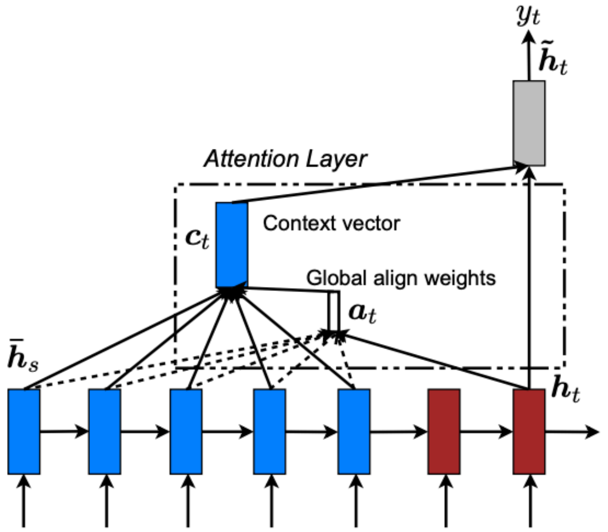
– **generating LaTeX** → Natural Language Processing (NLP)

# Literature Review

1) multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality → deep LSTM to decode the target sequence from the vector [1].




2) the introduction of attention by [2, 3] eventually established a new standard in Machine Translation systems, allowing impressive performance like zero-shot translation like in [4].

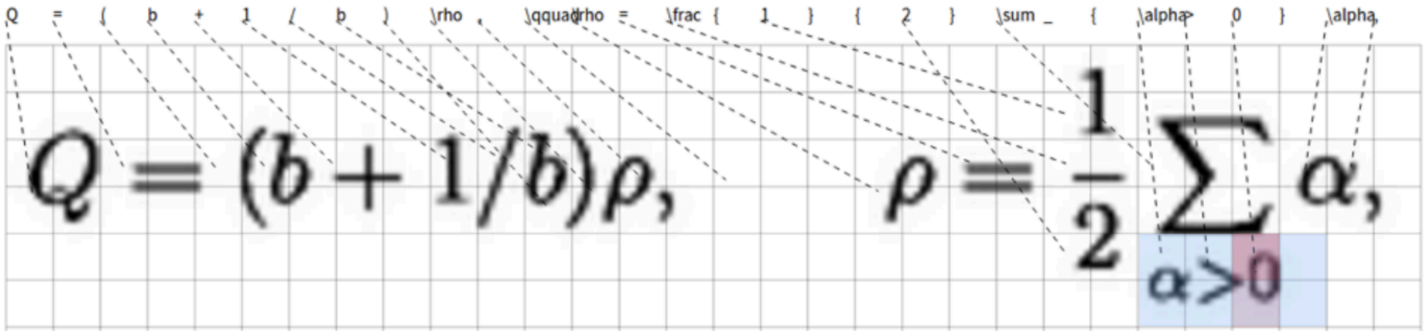


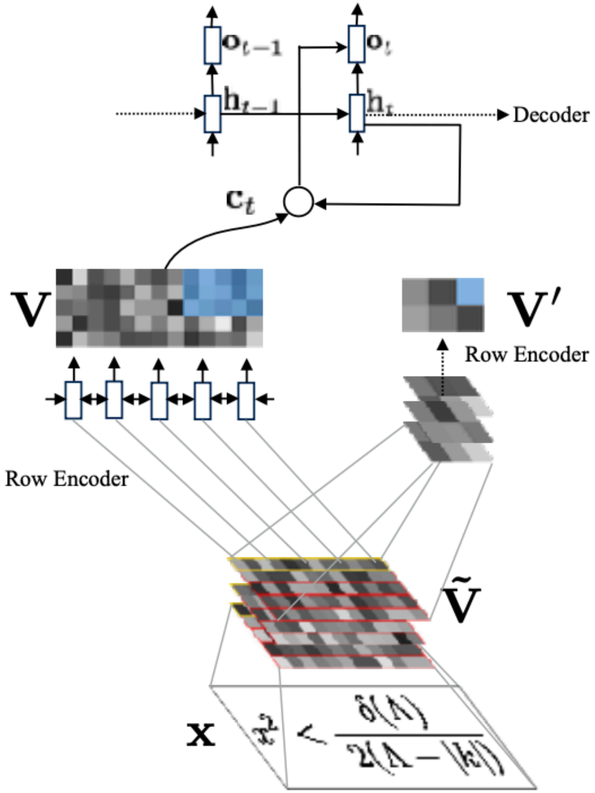


[1] I. Sutskever et al "Sequence to Sequence Learning with Neural Networks", 2014, https://arxiv.org/pdf/1409.3215.pdf
[2] D. Bahdanau et al "Neural Machine Translation by Jointly Learning to Align and Translate", 2014, https://arxiv.org/abs/1409.0473
[3] Minh-Thang Luong el all "Effective Approaches to Attention-based Neural Machine Translation", 2015, https://arxiv.org/pdf/1508.04025.pdf
[4] Johnson et all "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation", 2017, https://arxiv.org/pdf/1611.04558.pdf

# Literature Review

3) Deng's model [5] incorporates a multi-layer convolutional network over the image with an attention-based recurrent neural network decoder.



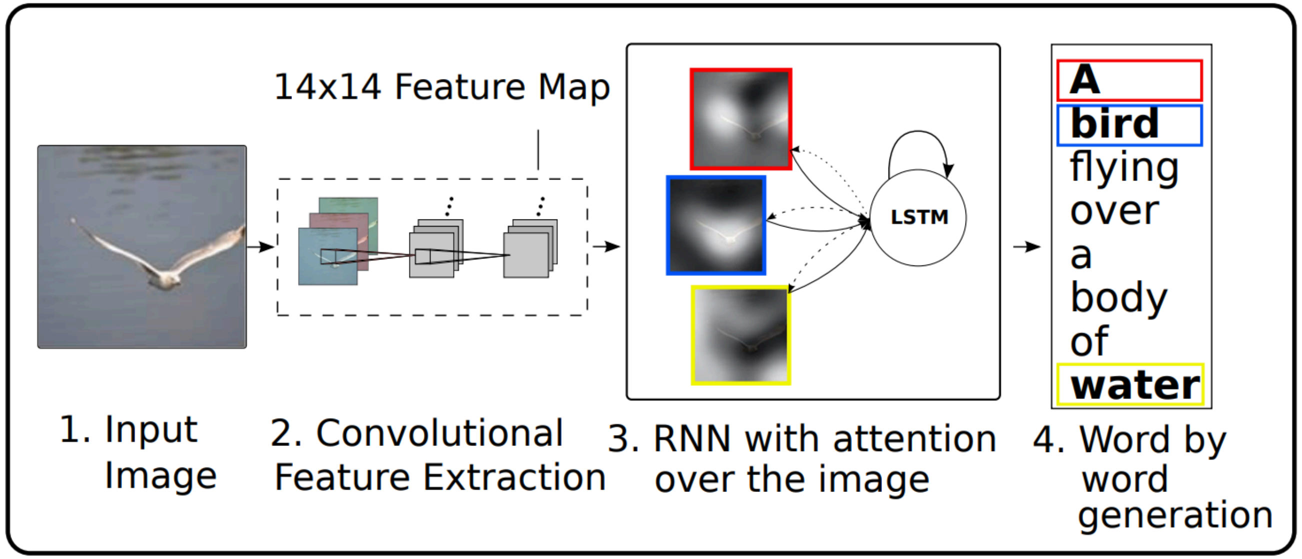




[5] Deng et al "Image-to-Markup Generation with Coarse-to-Fine Attention", 2017, https://arxiv.org/pdf/1609.04938.pdf

# Literature Review

Combining sequence-to-sequence with Image Captioning techniques, [6] encode the image in a fixed size vector with a CNN, and then decoding the vector step by step, generating at each step a new word of the caption and feeding it as an input to the next step.

An attention mechanism was added, enabling the decoder at each time step to look and attend at the encoded image, and compute a representation of this image with respect to the current state of the decoder




[6] Xu et al "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", 2016, https://arxiv.org/pdf/1502.03044.pdf

# Our Model

**ViT Encoder (from the scratch):**

– patching 16x16;

– embedding on the outcome.

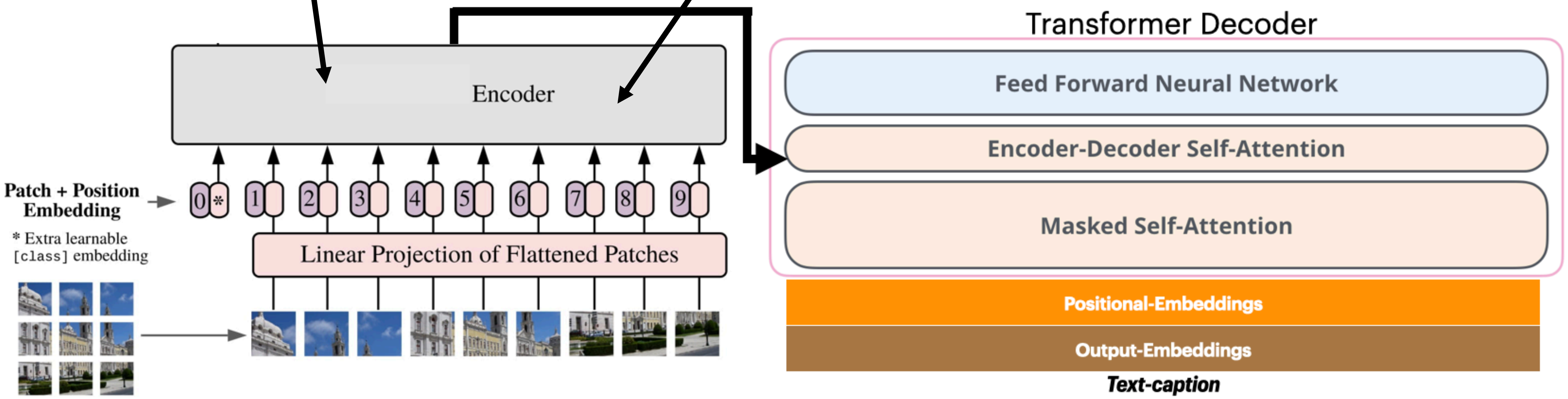
**CNN Encoder (resnet101):**

– pretrained;

– the first layer changed (as one-channel image);

– no dense layers and the end;

– added final convolutional layer.

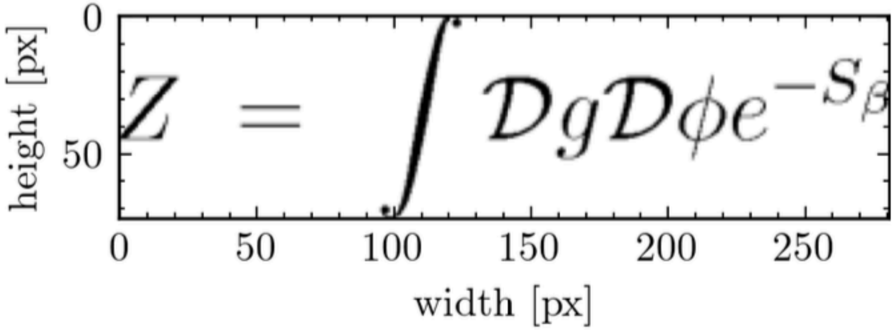


# Dataset

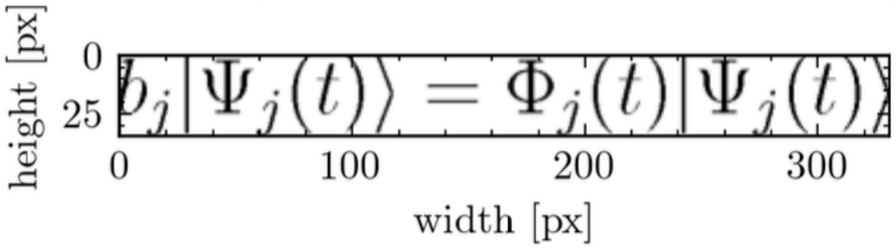
**I2L-140K** dataset was used.

Dataset contains a total of 154,944 LaTeX images and formulas. LaTeX formulas length is varied from 1 to 2177 symbols.

```
Z ~ = ~ \\int { \\cal D } g { \\cal
D } \\phi e ^ { -S _ { \\beta } }
```




```
b _ { j } | \\Psi _ { j } ( t )
\\rangle = \\Phi _ { j } ( t ) |
\\Psi _ { j } ( t ) \\rangle
```

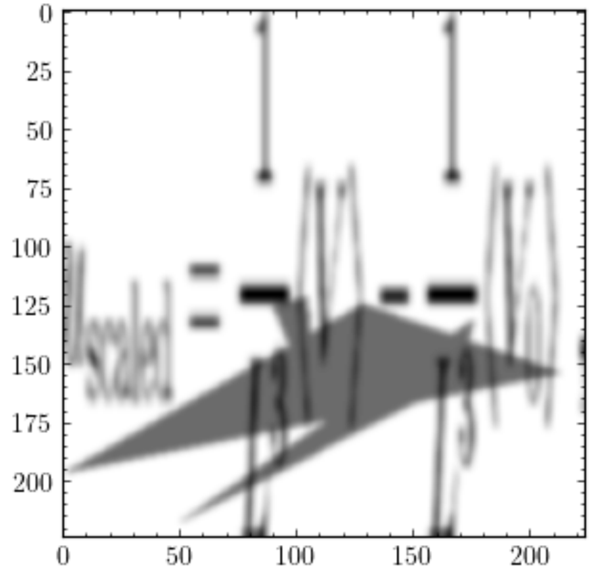


# Dataset Processing

The dataset was partially augmented using:

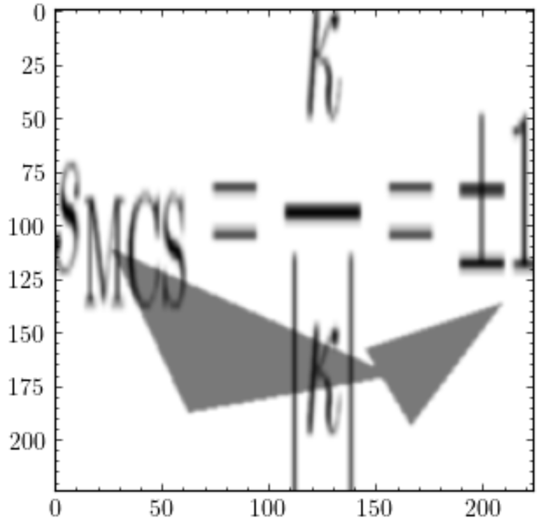
- Resize;
- RandomBrightnessContrast;
- RandomShadow;
- GaussianBlur.

$$u_{\mathrm{scaled}} = \frac{1}{L^3}\langle V \rangle - \frac{1}{L^3}\langle V_0 \rangle \ ,$$

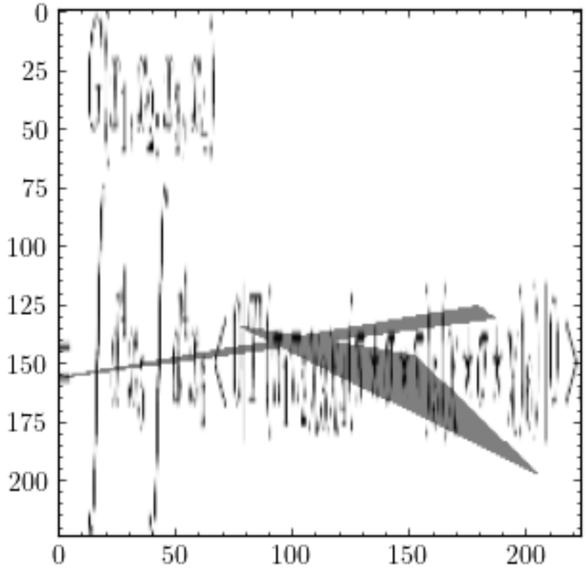


# Dataset Processing

$$s_{\mathrm{MCS}} = \frac{\kappa}{|\kappa|} = \pm 1$$




$$\begin{aligned} & G(x_1, x_2, x_3, x_4) \\ = \ & \int d^4x_5 \int d^4x_6 < 0|T\left(\pi_1\pi_2\chi_3\chi_4(\pi \star \sigma \star \pi)_5(\chi \star \sigma \star \chi)_6\right)|0 >, \end{aligned}$$



# Experiments

Also, two optimizers were tested: Adamax and Adam. Surprisingly, Adam with lr=1e-4 provided a bit better results.

To check the model correctness, we start with small subsets.

| Subset (items in train) | Epochs | Result |
|---|---|---|
| 50 | 300 | Overfitting |
| 500 | 300 | Overfitting |
| ... | ... | ... |

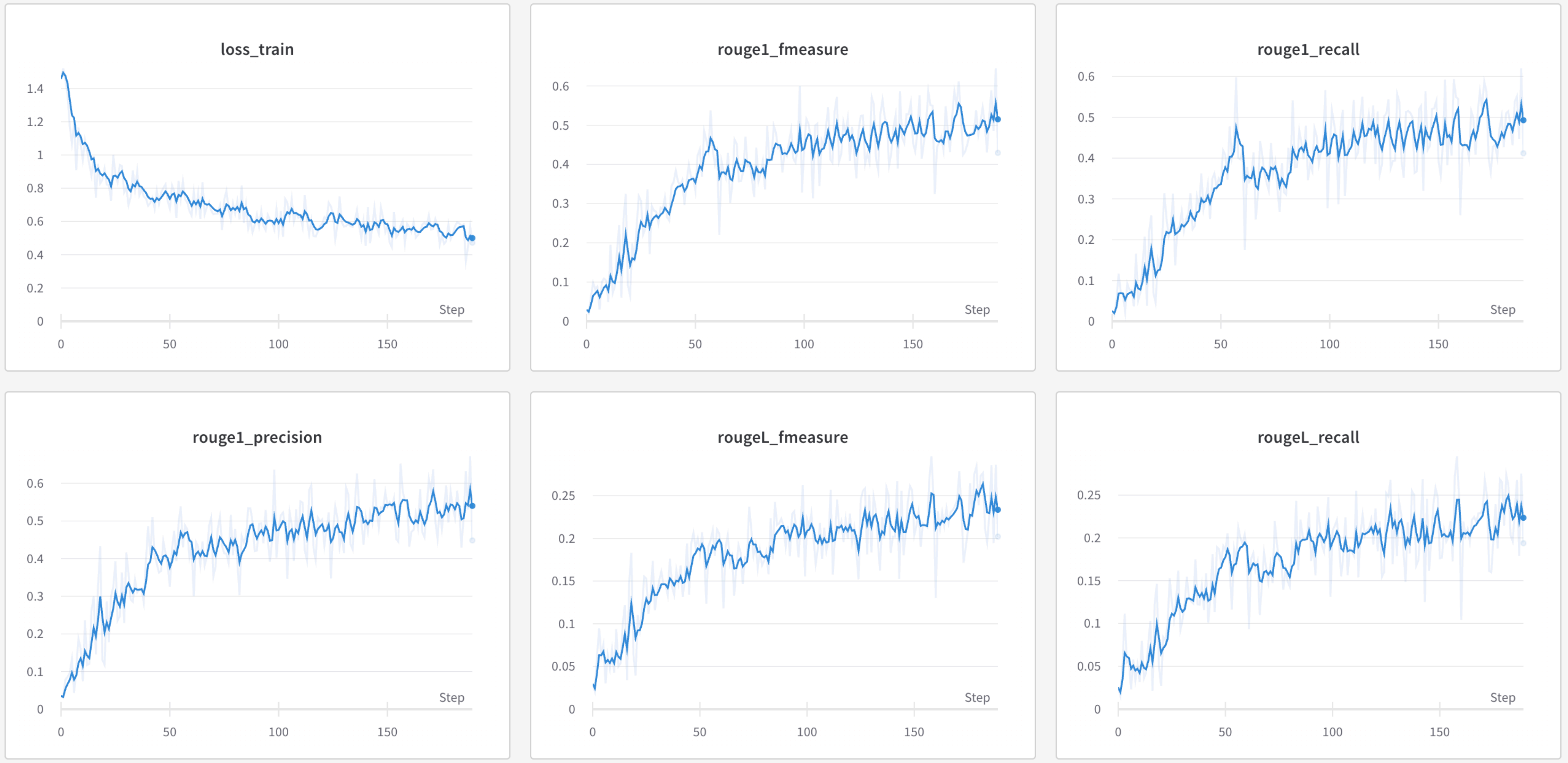
# Experiments

Started with whole dataset, but 1 epoch took ~ 1 hour. After the 1st epoch the cross entropy loss was ~2 providing awful results.

To check the model correctness, we start with small subsets and testing different hyperparameters.

| Hyperparameter Name | Value |
|---|---|
| Embedding Size | 512 |
| | 1024 |
| | 2048 |
| Hidden Size | 512 |
| | 1024 |
| | 2048 |
| Number of Heads | 8 |
| | 64 |
| | 128 |
| Number of Blocks | 3 |
| | 5 |
| | 6 |

# Experiments: CNN decoder




batch size = 20; number of epochs = 2.

# Experiments: ViT decoder

– NVIDIA A100-SXM4-40GB;
– batch size = 64;
– number of epochs = 6;

– Train Loss: 0.206
– Test Loss: 0.514

# Results: test set

original:

$$\vec{P}_{2T} = \xi' \vec{k}_T + \overrightarrow{\rho'} - \frac{(\vec{P}_1 \overrightarrow{\rho'})}{P_1^2} \vec{P}_1,$$

generated:

$$\vec{P}_{TT} = \xi \vec{k}_T + \vec{\rho} - \frac{(\vec{P}_T \vec{\rho})}{P_{11}^2} \vec{P}_1,$$

original:

$$\frac{d^2(x-x_0)}{dt^2} + 2\gamma \frac{d(x-x_0)}{dt} + \nu_0^2(x - x_0) = 0,$$

generated:

$$\frac{d^2(x-x_1)}{dt^2} + 2\gamma \frac{d(x-x_1)}{dt} + \sigma(x - x_0) = 0,$$

original:

$$\begin{pmatrix} A_\mu \\ Z_\mu \end{pmatrix} = \begin{pmatrix} c_W & s_W \\ -s_W & c_W \end{pmatrix} \begin{pmatrix} B_\mu \\ W_\mu^3 \end{pmatrix}.$$

generated:

$$\begin{pmatrix} A_\mu \\ Z_\mu \end{pmatrix} = \begin{pmatrix} \alpha_W & s_W \\ -m_W & h_W \end{pmatrix} \begin{pmatrix} \beta_\mu \\ \beta_W \end{pmatrix}.$$

original:

$$\Delta\langle\phi_q^2\rangle = \frac{1}{4\pi} \mathrm{l\,n} \frac{2c^2}{3\phi_c^2 - c^2 + 3\Delta\langle\phi_q^2\rangle}.$$

generated:

$$\Delta\Delta\phi_q^2 = \frac{1}{4\pi} \mathrm{l\,n} \frac{2\nu^2}{3\pi^2 - (\nu+\Delta)(\Delta q)^2}.$$

# Team Member's Contribution

**Antonina Kurdyukova** (20% of work)

- Project idea inspiration
- Preparing the report
- GitHub repo description

**Dmitrii Baluev** (20% of work)

- Preparing the presentation slides
- Final project speech
- Perform experiments

**Matvey Skripkin** (20% of work)

- Assemble and train the model
- Perform experiments

**Nikita Logitsaev** (20% of work)

- Preparing the dataset
- Data preprocessing and augmentation
- Dataset and data preprocessing description

**Nikolay Kalmykov** (20% of work)

- Trying CNN-based approach
- Perform experiments

# Sources

**Transformer weights (ViT):**
https://drive.google.com/file/d/17wLr29AGupBcCSTeOuIoi_zHQ3BrJytA/view?usp=share_link

**Model parameters (ViT):**
https://drive.google.com/file/d/1jsCX7nVeAtS9zW0MUB2H942FApZVhJUN/view?usp=share_link

**GitHub:**
https://github.com/barracuda049/imgtolatex

# Thx.

**DE-LE-PE Team:**
Antonina Kurdyukova
Dmitrii Baluev
Matvey Skripkin
Nikita Ligostaev
Nikolay Kalmykov