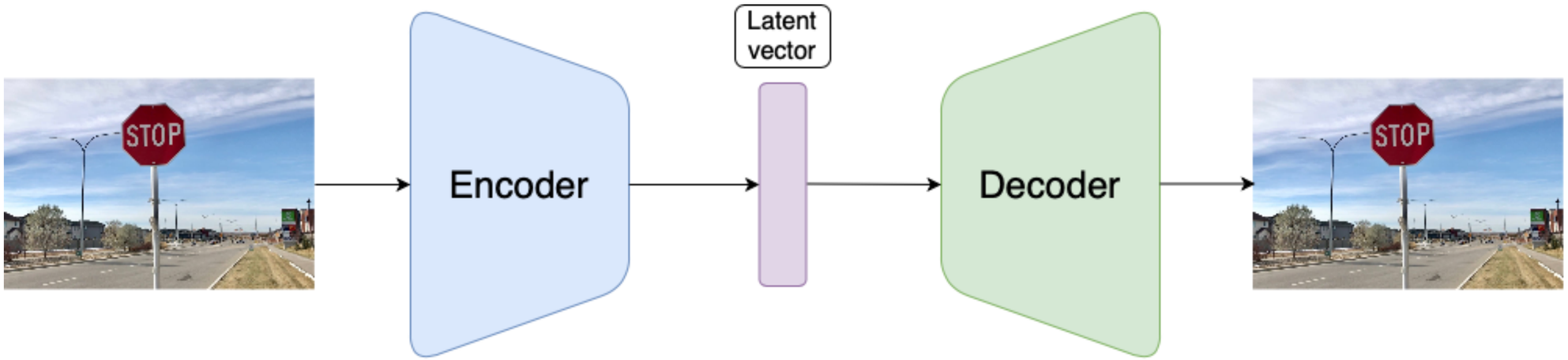
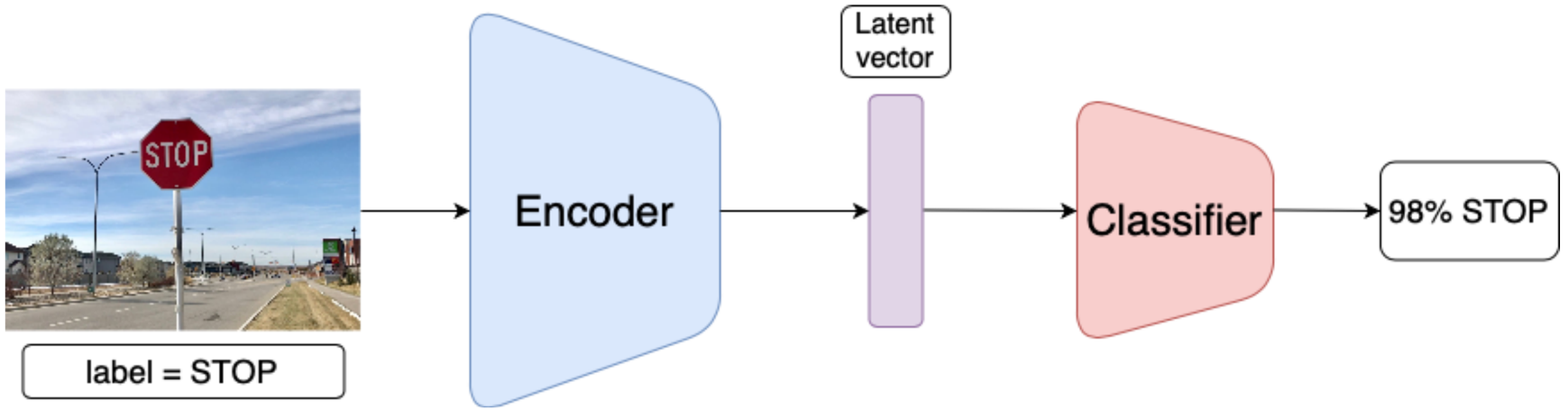
WOJCIECH PRATKOWIECKI

EVALUATION OF SUITABILITY OF CONDITIONAL GENERATIVE MODELS FOR CLASSIFICATION

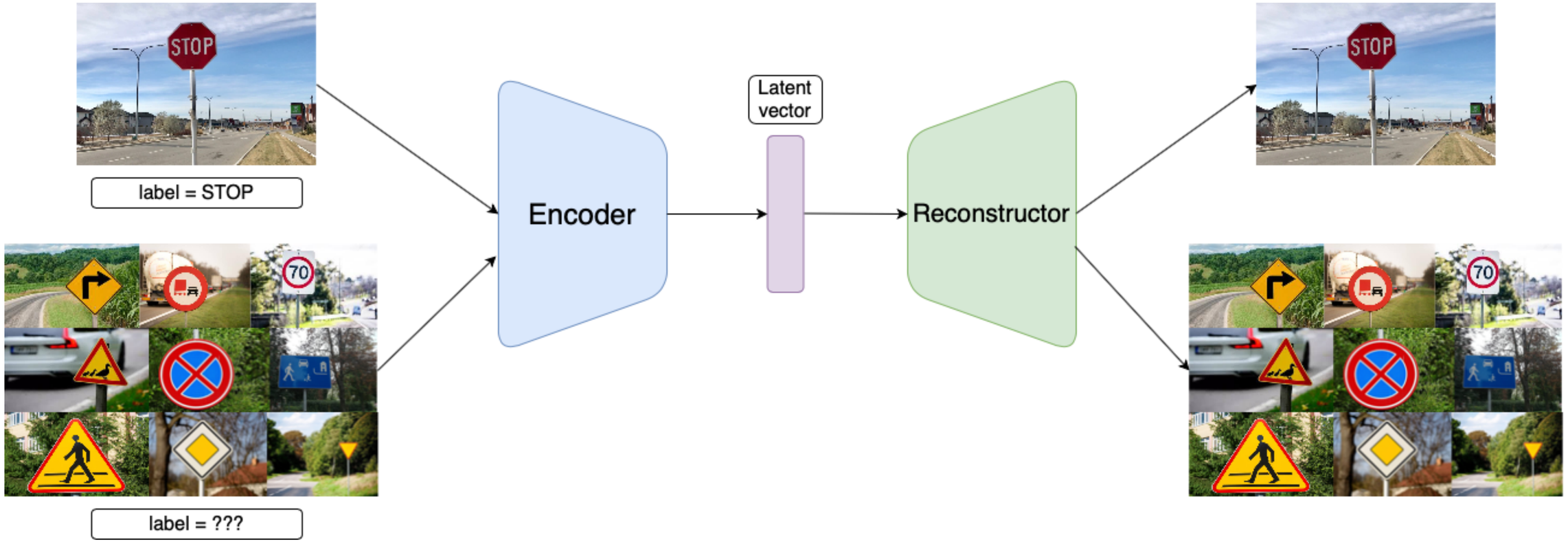
SEMI-SUPERVISED LEARNING



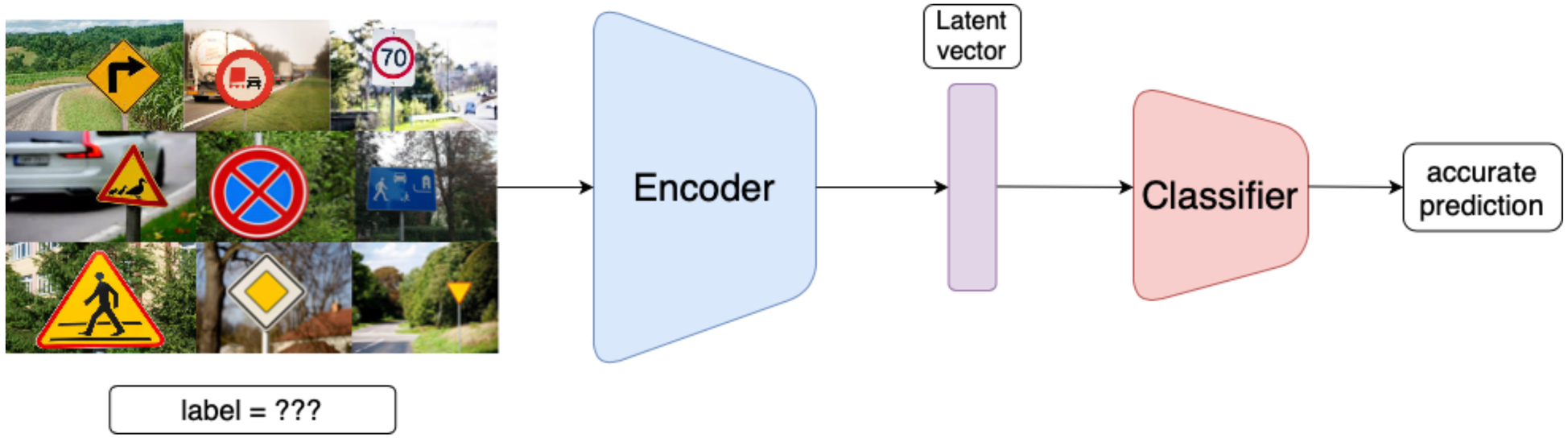
SEMI-SUPERVISED LEARNING



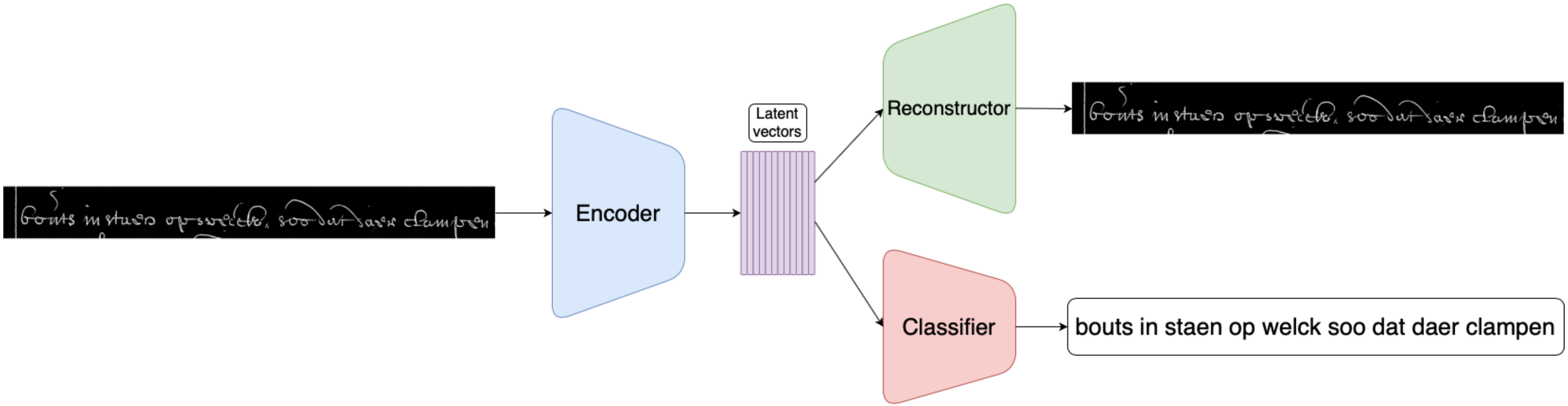
SEMI-SUPERVISED LEARNING



SEMI-SUPERVISED LEARNING



SCRIBBLELENS CLASSIFICATION WITH SEMI-SUPERVISED LEARNING

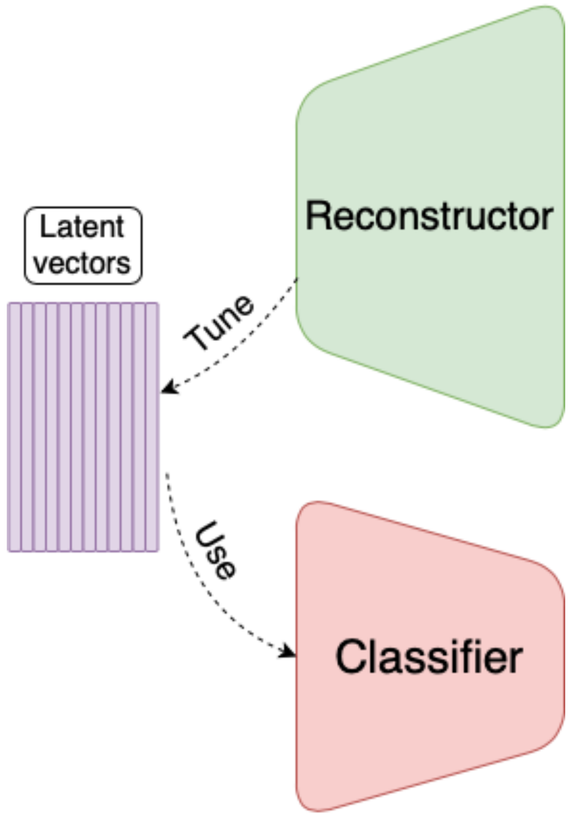


RECONSTRUCTOR'S ROLE

- Encoder E with parameters Θ_E
- Reconstructor R with parameters Θ_R
- line image l
- Reconstructor's loss function:

$$L_R(l, R(E(l; \Theta_E); \Theta_R))$$

RECONSTRUCTOR'S ROLE



RECONSTRUCTOR'S ROLE

- Preferred latent representation:
  - Easy to classify
  - Latent vectors related to represented characters
  - Could resemble one-hot vectors of letters written on the image
- We would like a reconstructor to demand such a latent representation
- Desired reconstructor would prefer line's transcript over any other conditioning
- We may pass the conditioning manually and train a reconstructor separately from any encoder

RECONSTRUCTOR AS A CLASSIFIER

RECONSTRUCTOR AS A CLASSIFIER

- Reconstructor is a generative model. It captures:

$$P(X \mid Y)$$

  - X - ScribbleLens line
  - Y - Conditioning. We may use sequence of one-hot vectors

RECONSTRUCTOR AS A CLASSIFIER

- Reconstructor is a generative model. It captures:

$$P(X \mid Y)$$

- Bayes' theorem:

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)}$$

RECONSTRUCTOR AS A CLASSIFIER

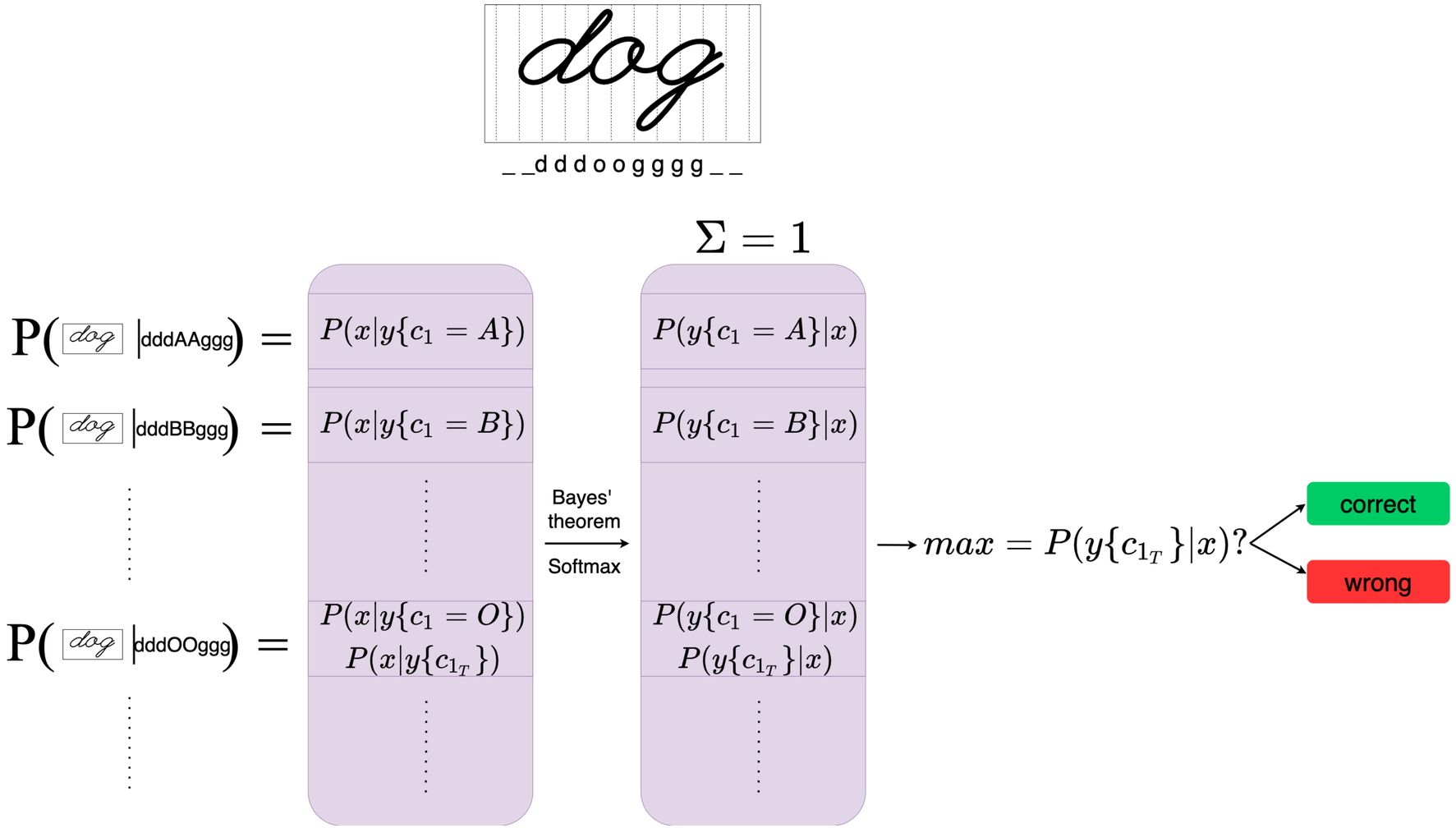
- Bayes' theorem:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

- If we assume that each transcript is equally likely:

$$P(Y|X) = P(X|Y) \cdot \alpha$$

RECONSTRUCTOR AS A CLASSIFIER



RECONSTRUCTORS COMPARISON

- For each reconstructor R we evaluated it on ScribbleLens lines containing 300 characters
- For each letter we computed the most likely character ($P(y\{c_{i_j}\} \mid x)$)
- Reconstructor's accuracy = how many of 300 letters classified correctly

RECONSTRUCTORS COMPARISON

- We compared different reconstructor architectures this way

Model	Accuracy	Average probability of the true character	Average probability of the most likely character when classified correctly	Average probability of the most likely character when classified wrongly
PixelCNN	59%	55%	86%	53%
PixelCNN Large	55%	53%	88%	57%
PixelCNN Lookahead various number of skipped columns	26.6%	24.2%	83.1%	71.9%
PixelCNN Lookahead fixed number of skipped columns	47%	43%	85%	52%
PixelCNN right-to-left	48%	42%	81%	52%
WaveNet	39%	39.2%	94%	81.4%
WaveNet Large	32%	32%	92%	82%
WaveNet lookahead various number of skipped columns	21%	20.9%	94.5%	86.7%
WaveNet noisy labels	11.5%	—	—	—
Bidirectional: PixelCNN + PixelCNN	**71%**	68%	94%	66%

RECONSTRUCTORS COMPARISON

- Most accurate model: Bidirectional PixelCNN (71%)

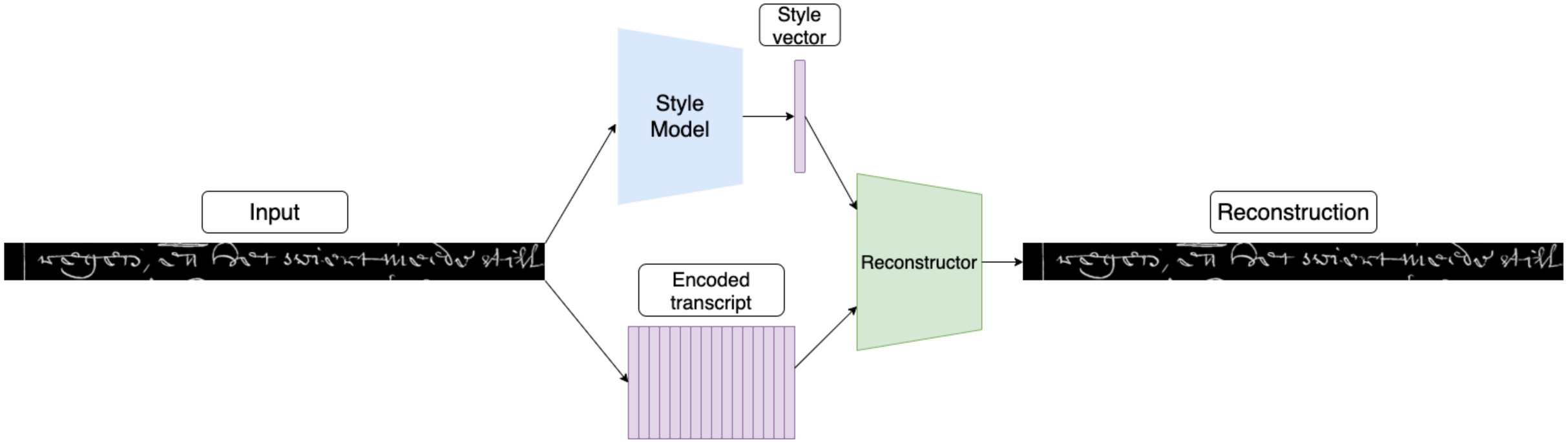
Model	Accuracy	Average probability of the true character	Average probability of the most likely character when classified correctly	Average probability of the most likely character when classified wrongly
PixelCNN	59%	55%	86%	53%
PixelCNN Large	55%	53%	88%	57%
PixelCNN Lookahead various number of skipped columns	26.6%	24.2%	83.1%	71.9%
PixelCNN Lookahead fixed number of skipped columns	47%	43%	85%	52%
PixelCNN right-to-left	48%	42%	81%	52%
WaveNet	39%	39.2%	94%	81.4%
WaveNet Large	32%	32%	92%	82%
WaveNet lookahead various number of skipped columns	21%	20.9%	94.5%	86.7%
WaveNet noisy labels	11.5%	—	—	—
Bidirectional: PixelCNN + PixelCNN	**71%**	68%	94%	66%

STYLE MODELING

STYLE MODELING

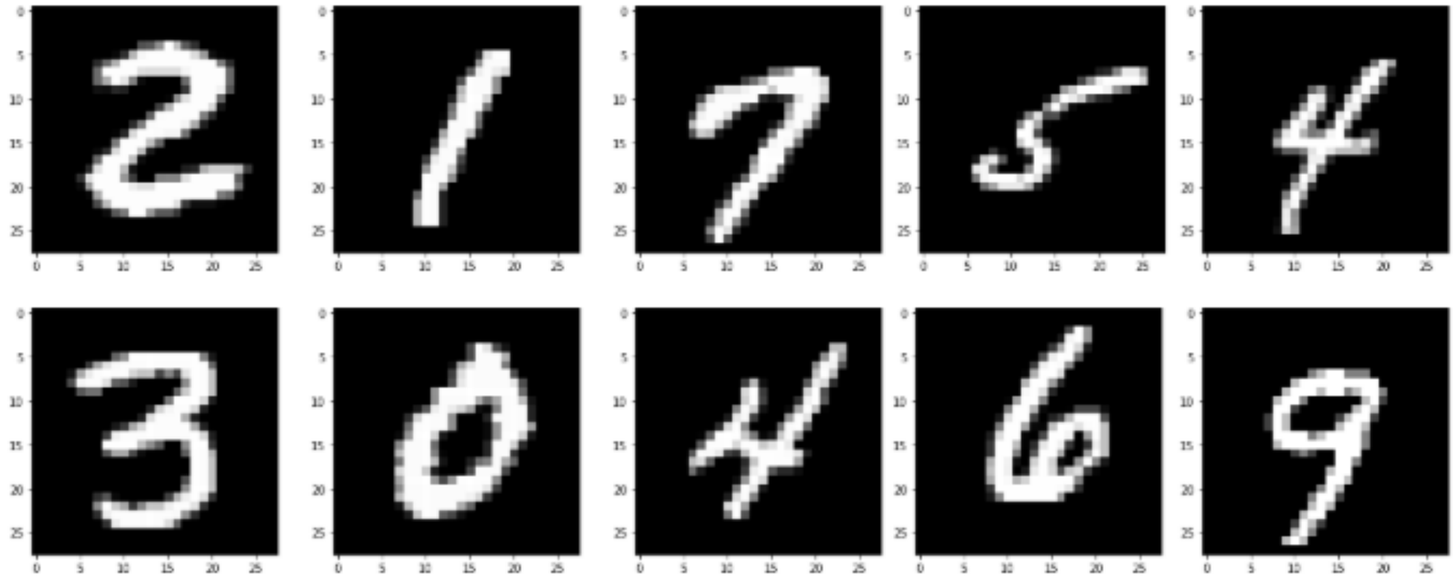
- We would like to improve reconstructor's performance
- A feature that strongly differentiates the pictures is author's handwriting style
- If we provide the style information to the reconstructor, it could demand more characters-related features from the latent representation

STYLE MODELING

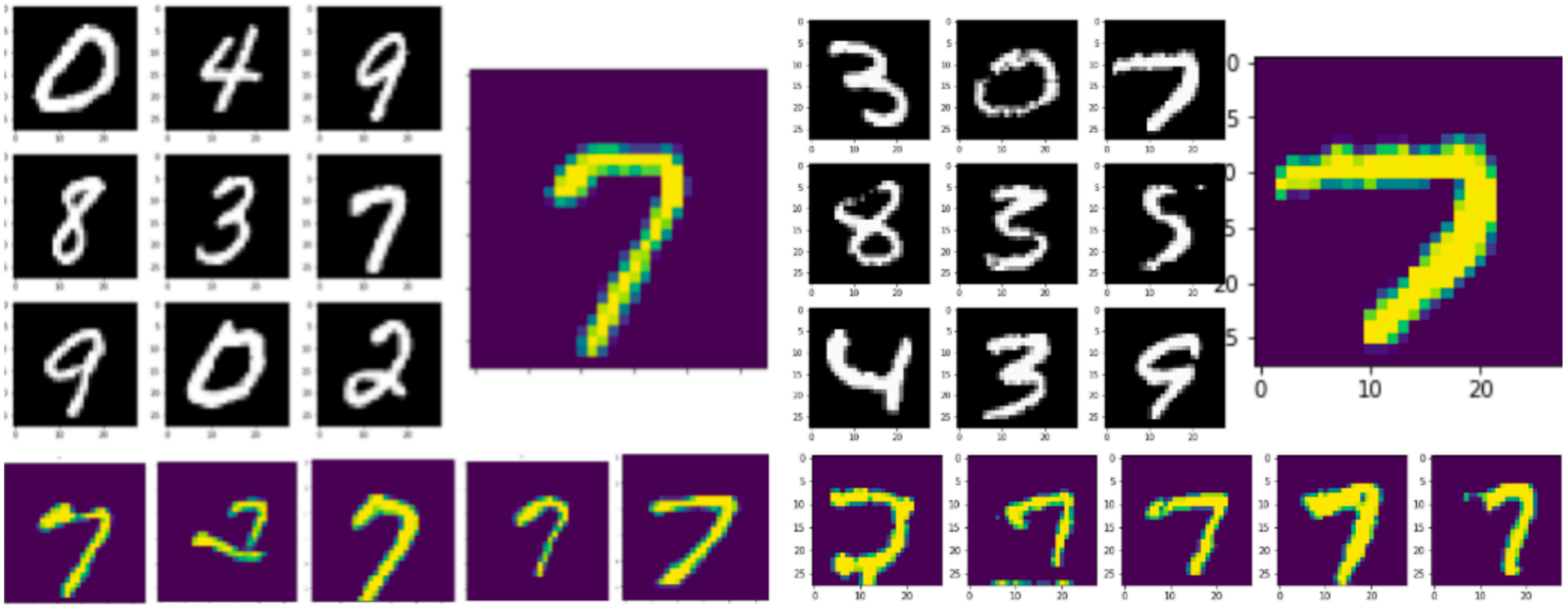


STYLE MODELING

- We can use QMNIST dataset to simplify the computations
  - Pictures of handwritten digits
  - Each sample labeled with digit id and author id



STYLE MODELING



STYLE MODELING

- A style vector is computed as a mean of k vectors
- A style model improves performance in the image reproduction task

Model	Reconstruction loss
PixelCNN without style model	0,2785
PixelCNN with S_{16} style model	0,2681

STYLE MODELING

- We may use the reconstructors to classify the transcript (presented digit)
- Using the style model improves the model's performance

Model	Classification accuracy
PixelCNN without style model	98,03%
PixelCNN with S_4 style model	98,14%
PixelCNN with S_1 style model using all samples to compute the style vector	98,41%

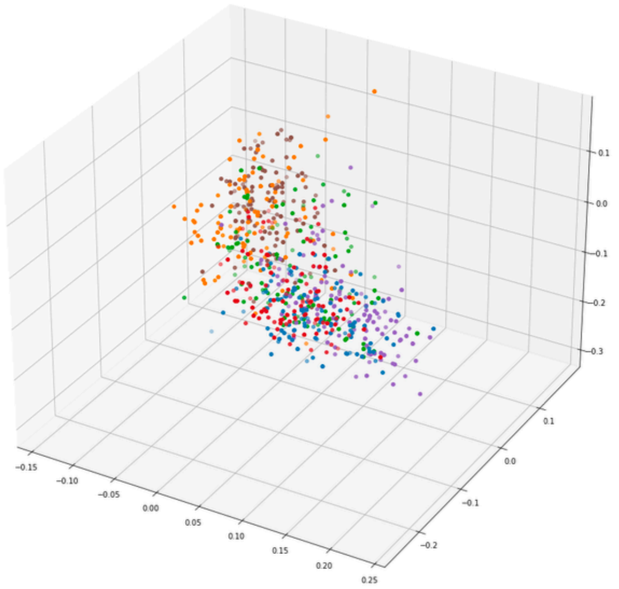
STYLE MODELING

- We could notice a significant classification accuracy drop when “fake” author was used to compute a style vector

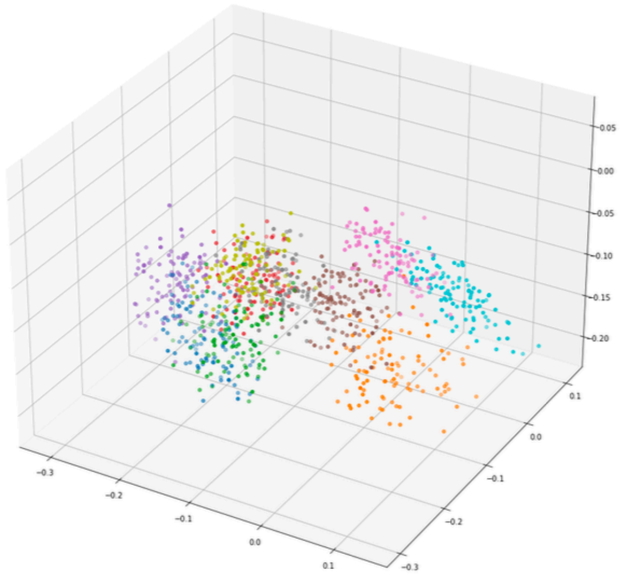
Model	Classification accuracy	Classification accuracy when “fake author” style vector was used
PixelCNN with S_1 style model	98,11%	96,28%
PixelCNN with S_8 style model	98,18%	89,17%
PixelCNN with S_{32} style model	98,1%	85,81%

STYLE MODELING

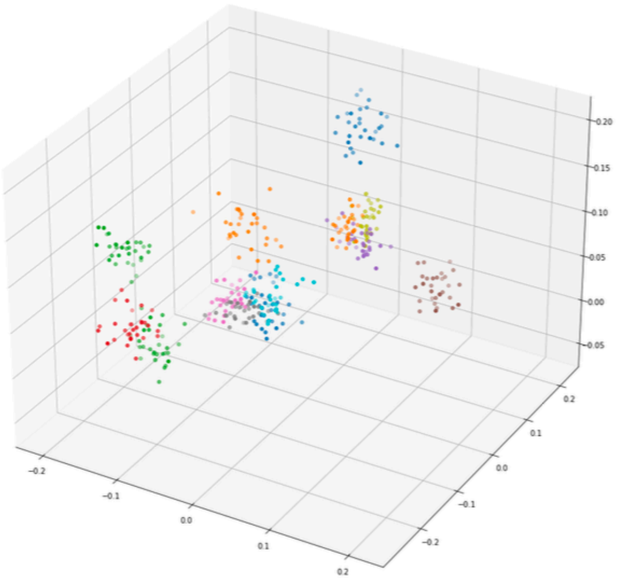
- By extending the style model with a layer mapping the output to a 3 dimensional vector we can visualize the style vectors



(a) S_1^3



(b) S_8^3



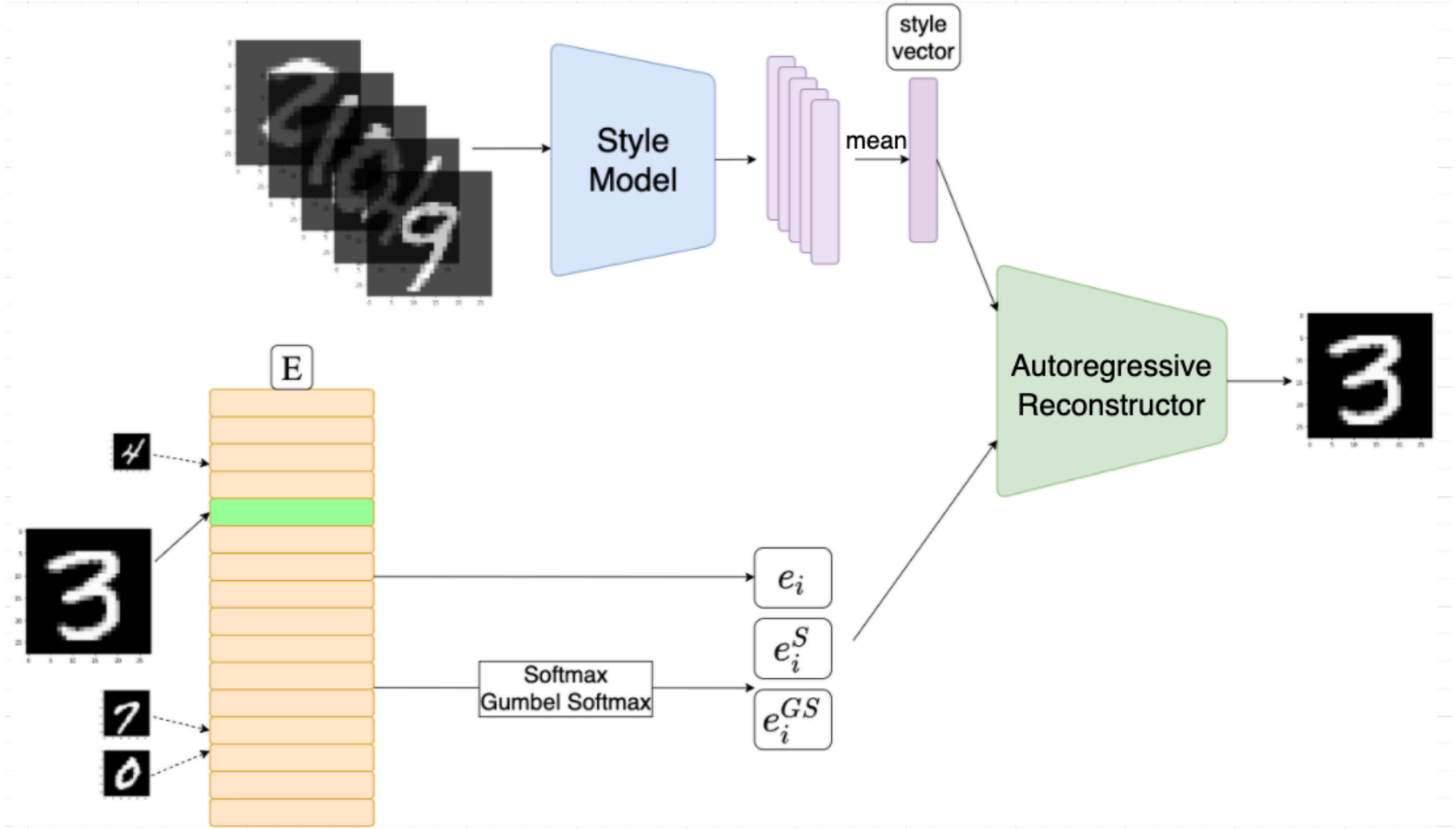
(c) S_{32}^3

LEARNED EMBEDDINGS

LEARNED EMBEDDINGS

- So far we used manually passed conditioning in form of one-hot vectors
- We may wonder, whether a reconstructor would indeed produce such conditioning, if it was able to tune these values
- We can allow it to assign an individual vector e for each data sample, that would be used instead of one-hot vectors during the training
- If we were able to predict a digit presented on image assigned to embedding vector e, the e contains label-related features

LEARNED EMBEDDINGS



LEARNED EMBEDDINGS

- e vectors without significantly increase the reconstruction accuracy

Model	Reconstruction loss
PixelCNN without conditioning	0,2936
StylePixelCNN conditioned on one-hot vector	0,2811
StylePixelCNN conditioned on $e \in \mathbb{R}^{128}$ embedding	0,2362

LEARNED EMBEDDINGS

- However we were not able to predict the digit presented on the corresponding image. Label-related features were not stored in the embedding vectors.

Embedding type	Embedding classification accuracy
$e \in \mathbb{R}^{10}$	27%
$e \in \mathbb{R}^{10}$ with Softmax	28%
$e \in \mathbb{R}^{10}$ with Gumbel Softmax	12%

LEARNED EMBEDDINGS

- We may initialize the embeddings as one-hot vectors and use a reconstructor trained on such conditioning.
- A reconstructor is allowed to modify the conditioning after the training.

LEARNED EMBEDDINGS

- Embeddings are easy to classify this time. Also reconstruction result is much better.
- This shows That label-related features are beneficial for the reconstructor, however it is unable to model it in the embeddings.

Metric	Score
reconstruction loss (nats/pix) with Softmaxed one-hot conditioning	0,2821
reconstruction loss (nats/pix) with learned Softmaxed embeddings initialized as one-hot vectors	0,2640
Embeddings e_i classification	99,99%

SUMMARY

- We evaluated the reconstructor in isolation from any encoder to get a better understanding of its performance.
- We proposed an approach based on the Bayes' theorem to compare different reconstructors.
- Introduced style model improved the reconstructor's performance. We might notice, that the model indeed captures author's style.
- Replacing conditioning vectors with learned embeddings showed the reconstructor's behaviour when it is able to tune latent vectors.

THANK YOU FOR ATTENTION 👨‍🎓