Autoregressive Generation of Neural Field Weights

Using a transformer based architecture

from Luis Muschal and Luca Fanselau

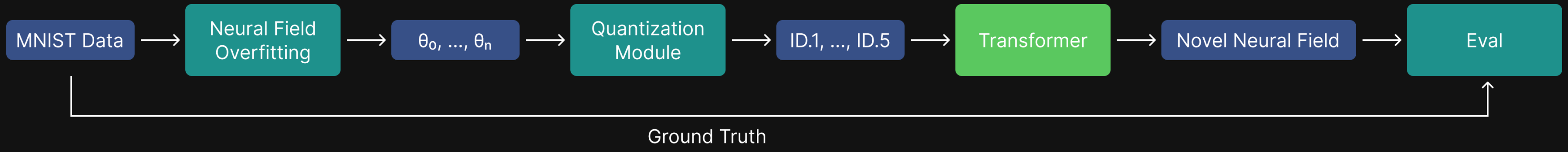
Recap

Weight initialization for neural fields

- **Goal**: Generate novel neural fields using an autoregressive process
- **Problem**: Permutation problem arises when transforming neural fields into sequences
- **Solution**: Condition the weights to decrease structural differences between neural fields
- **First presentation**: Trained a Regression Transformer to generate neural fields, but ran into novelity issues

Experiment

From Regression Transformer to Traditional Transformer



General Procedure:

1. Tokenization of weights using Vector Quantization
2. Training Transformer and tune hyperparameters
3. Optimizing and evaluate inference

Quantization

Tokenization of weights using Vector Quantization

Approach: Continuous Neural Field weights are discretized using Vector Quantization

Procedure:

- Learning Codebook using weights of all MNIST Neural Fields

Quantization

Training of Vector Quantization

Training:

1. Codebook elements randomly initialized
2. **Forward**: Assign Weight to the closest Codebook element
3. **Backward**: Update Codebook elements by minimizing L2-loss
4. **Correction**: Assign rarely used elements to weights
5. goto 2.

Quantization

Special Tokens

Special Tokens:

- "Start of Sequence" Token **SOS**
  - indicating the start of the sequence
- "Conditioning" Token **C**
  - indicating to which number the weights belong

Metrics - Novelty

Introduction

- S_g: Set of **evaluated** generated neural fields
  - Images generated from novel neural fields
- S_r: Set of **evaluated** reference neural fields
  - Images generated from training neural fields
- $D(X, Y)$: Distance between elements $X, Y \in S_g \cup S_r$
  - Here Structural Similarity Index (SSIM) is used

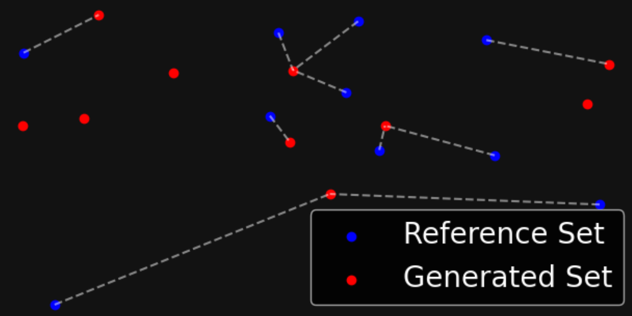
Metrics - Novelty

Minimum Matching Distance (MMD)

- S_g: Set of **evaluated** generated neural fields
  - Images generated from novel neural fields
- S_r: Set of **evaluated** reference neural fields
  - Images generated from training neural fields
- $D(X, Y)$: Distance between elements $X, Y \in S_g \cup S_r$
  - Here Structural Similarity Index (SSIM) is used

$$\mathrm{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X, Y)$$

- Average distance between reference images and their closest neighbor in the generated set
- Lower is better



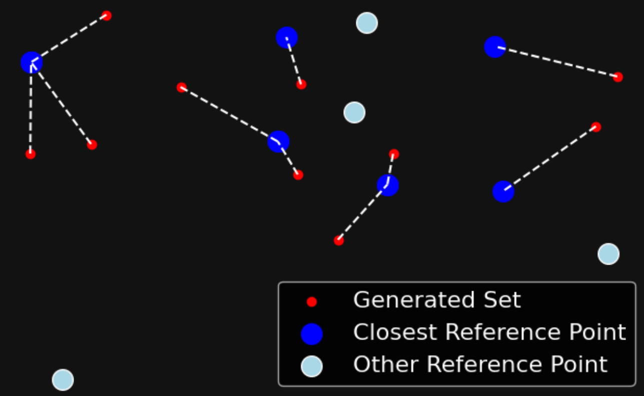
Metrics - Novelty

Coverage

- S_g: Set of **evaluated** generated neural fields
  - Images generated from novel neural fields
- S_r: Set of **evaluated** reference neural fields
  - Images generated from training neural fields
- $D(X,Y)$: Distance between elements $X, Y \in S_g \cup S_r$
  - Here Structural Similarity Index (SSIM) is used

$$\mathrm{COV}(S_g, S_r) = \frac{|\{\arg\min_{Y \in S_r} D(X,Y) | X \in S_g\}|}{|S_r|}$$

- Coverage of the reference by the generated set
- Higher is better



Metrics - Novelty

1-Nearest Neighbor Accuracy (1-NNA)

- S_g: Set of **evaluated** generated neural fields
  - Images generated from novel neural fields
- S_r: Set of **evaluated** reference neural fields
  - Images generated from training neural fields
- $D(X, Y)$: Distance between elements $X, Y \in S_g \cup S_r$
  - Here Structural Similarity Index (SSIM) is used

$$1 - \mathrm{NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{1}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{1}[N_Y \in S_r]}{|S_g| + |S_r|}$$

$$N_X = \operatorname*{argmin}_{Y \in S_r \cup S_g} D(X, Y)$$

- 50% is the optimal value
- Sum of the elements of S_g and S_r that are closest neighbors in their respective sets
- Divided by the total number of elements in S_g and S_r

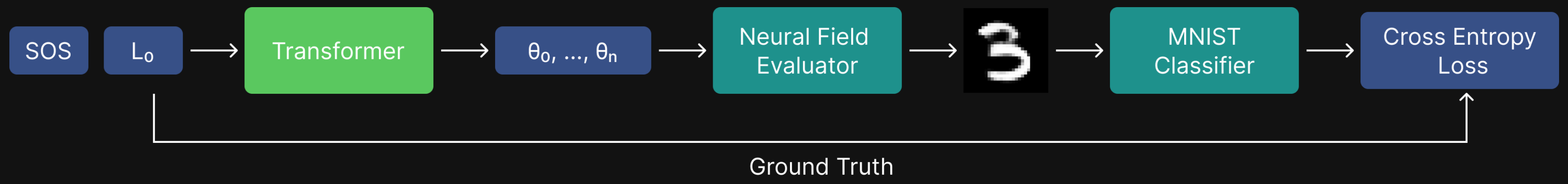
Metrics - Image Fidelity

MNIST Classifier Score

Proxy metric: Generate neural fields which lead to *understandable* digits

Procedure:

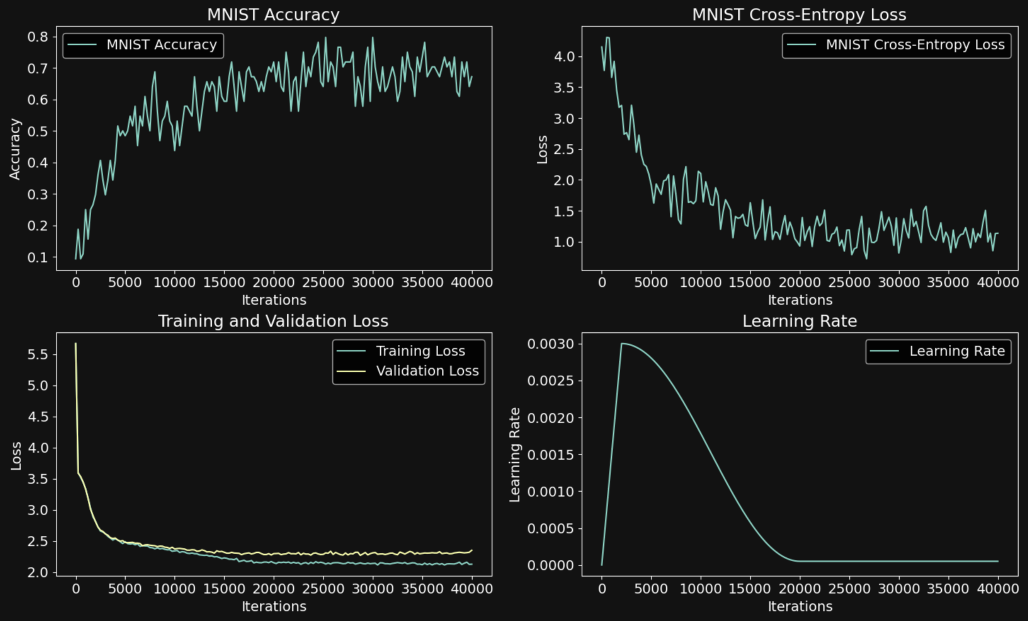
- Train a classifier on MNIST dataset
- Generate a novel neural field using a conditioning token
- Use the data pair (neural field, digit) to get a score from the classifier



Train a Transformer

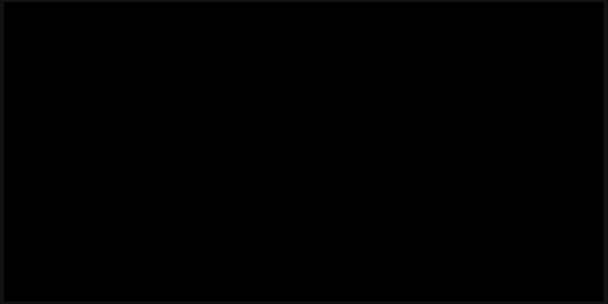
Hyperparameters

Hyperparameter Training	
Learning Rate	3e-3
Iterations	40000
Batch Size	64
Hyperparameter Transformer	
Embedding Size	240
Numbers of Heads	12
Numbers of Attention Blocks	12
Vocabulary Size	256
Context Length	562



Preliminary Results

Autoregressive Generation and Initial Results



Tuning Inference Parameters

Determining top-k, temperature

- **Top-k**: Reduces number of considered tokens
- **Temperature**: Smooths the distribution of the logits

$$L \hat{=} \text{Logits}$$
$$T \hat{=} \text{Temperature}$$

$$\text{Softmax}(L) = \frac{\exp(L/T)}{\sum_i \exp(L_i/T)}$$

Results

For all conditioning tokens

Results for all conditioning tokens

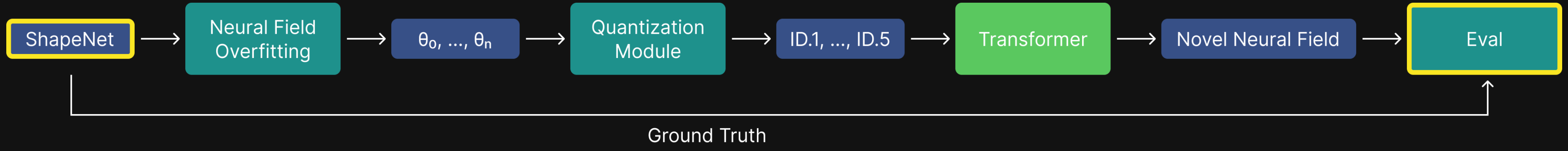
Results for all conditioning tokens

Results for all conditioning tokens for $T = 0.8$ and $\text{top-k} = 3$

$$\mathrm{COV}(S_g, S_r) = 0.2773 \quad 1 - \mathrm{NNA}(S_g, S_r) = 0.84 \quad \mathrm{MMD}(S_g, S_r) = 0.2162$$

Outlook

From MNIST to ShapeNet



Challenges:

- Neural Fields for ShapeNet have an increased complexity

Solution:

- Map a vector of neural field weights to token

Future:

- Qualitative comparison to State-of-the-Art methods

Thank you for your attention!

We hope you enjoyed our presentation and are looking forward to your questions.