GRIPex AI CEI Project: Reconstructing Atomic Positions from Coulomb Explosion Data

An Encoder-Decoder Approach to Solve the Inverse Problem

September 30th, 2024

Amirhossein Ghanaatian
Supervisor: Prof. Caragea

- Develop an Encoder-Decoder model to solve the inverse problem
- Input: Final momenta of atoms
- Condition: Some features made of momenta
- Output: Predicted initial positions of atoms before Coulomb explosion
- Inverse problem: Final momentum → Initial atomic positions

Approach 1

Auto-encoder

- Build an auto-encoder to:
  - Reconstruct x
  - Find a representation as an inference (latent layer or probability distribution of positions)

Conditional Variational Autoencoder (CVAE)
- Use the latent representation (probability) from the auto-encoder as the input of CVAE
- Add momenta as the condition of the model
- CVAE learns to generate the position based on the latent representation and the momenta condition

Approach 1

Training Phase

- Learn the representation of position in the training phase using the auto-encoder

- Train the CVAE using the latent representation as input and momenta as the condition

- CVAE learns to map the latent space to the position space conditioned on the momenta
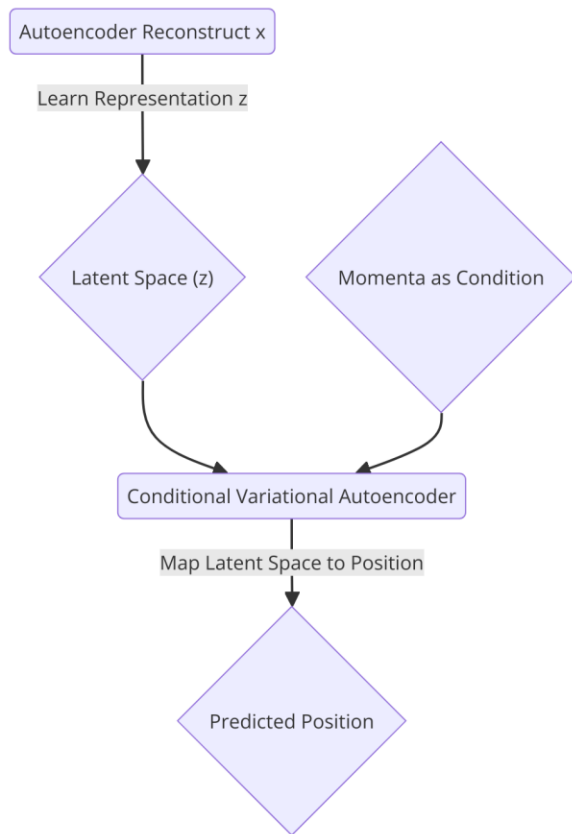
Testing Phase

- Use the probability of z from the auto-encoder developed in the training phase

- Feed the latent representation and momenta from the test set into the trained CVAE

- CVAE generates the predicted position based on the learned mapping

Approach 1

1. Condition: momenta Input: learned representation from the auto-encoder in the training phase Pay attention to the definition of input and condition in a CVAE

2. In the testing phase: Use the learned representation (z) from the training phase Do NOT use the test's position to learn a new representation

Approach 1

Autoencoder Reconstruct x

Learn Representation z

Latent Space (z)

Momenta as Condition

Conditional Variational Autoencoder

Map Latent Space to Position

Predicted Position

Approach 2

Model Initialization

- CVAE model defined with an encoder and decoder
- Encoder processes positions (X) to produce latent variables (Z) via mu and log(var)
- Decoder reconstructs positions (X) from latent variable Z and conditions on momenta (Y)

Approach 2

Training Phase

- Training loop trains model using reconstruction loss and KL divergence (required for CVAEs)

- Encoder processes positions (X) to produce latent distribution parameters (mu, logvar)

- Z is sampled using the reparameterization trick

- Decoder reconstructs positions using latent Z and momenta (Y), optimizing the loss function

Approach 2

Latent Representation Saving
- After training, latent parameters (mu, logvar) are saved
- Saved parameters will be used for sampling during testing phase
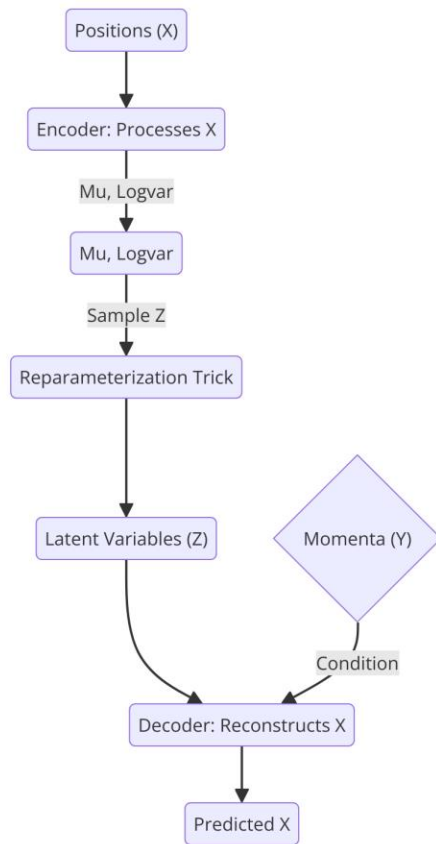
Approach 2

Testing Phase

- Encoder is not used during testing (safeguard against data leakage)
- Latent variables are sampled from the learned distribution in the training phase
- Decoder uses sampled latent variables along with test momenta (Y) to predict test positions (X)
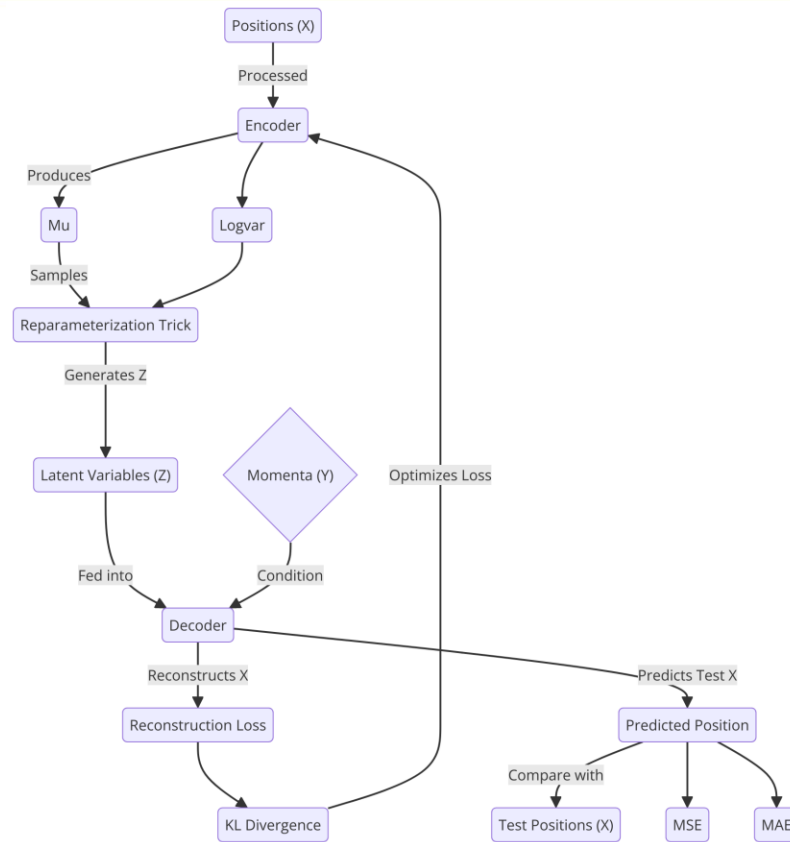
Approach 2

Evaluation

- Evaluation metrics (MSE, MAE) computed by comparing predicted positions with actual test positions
- Results are saved and visualized as specified
- Latent space (Z) is only sampled from the learned distribution in the training phase
- Ensures no test positions (X) are used to influence the prediction

Approach 2

Results

Dataset	Approach1		Approach2	
	MSE	MAE	MSE	MAE
generated_cos3d_check	1.0499	0.8151	1.0559	0.8179
random_cos3d_10000	0.1518	0.3107	0.1549	0.322
cei_traning_orient_1	0.6814	0.4914	0.677	0.4922