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

An Encoder-Decoder Approach to Solve the Inverse Problem

October 7th, 2024

Amirhossein Ghanaatian
Supervisor: Prof. Caragea

Last Session's Review

- Develop an Encoder-Decoder model to solve the inverse problem
- Conditional Variational Auto-Encoders(CVAE)
- 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

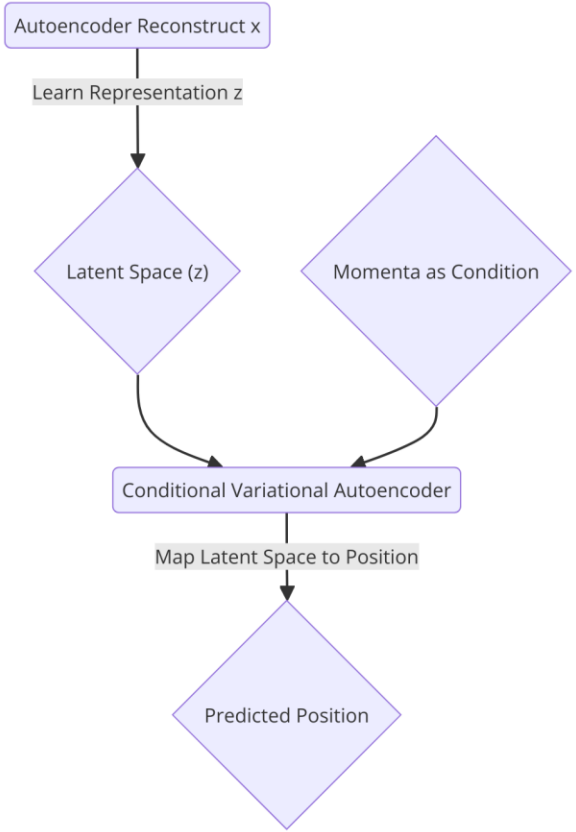
Testing Phase

- 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

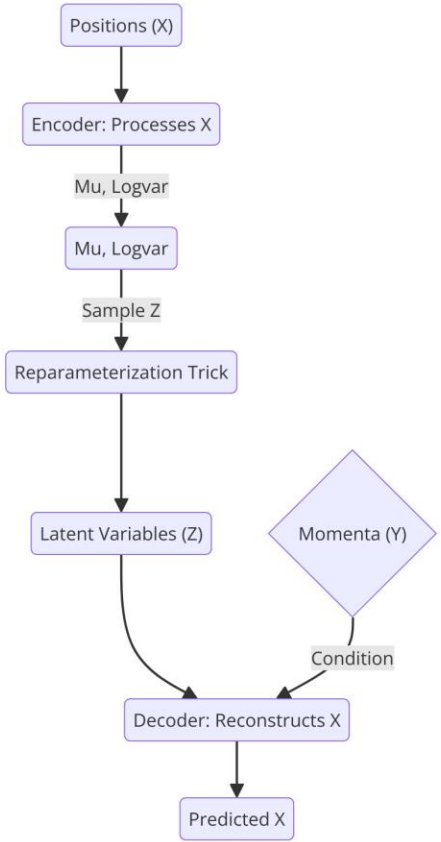
- Encoder is not used during testing (safeguard against data leakage)
- Latent variables are sampled from the learned distribution in the training phase
- Decoder uses sampled latent variables along with test momenta (Y) to predict test positions (X)

Approach 2

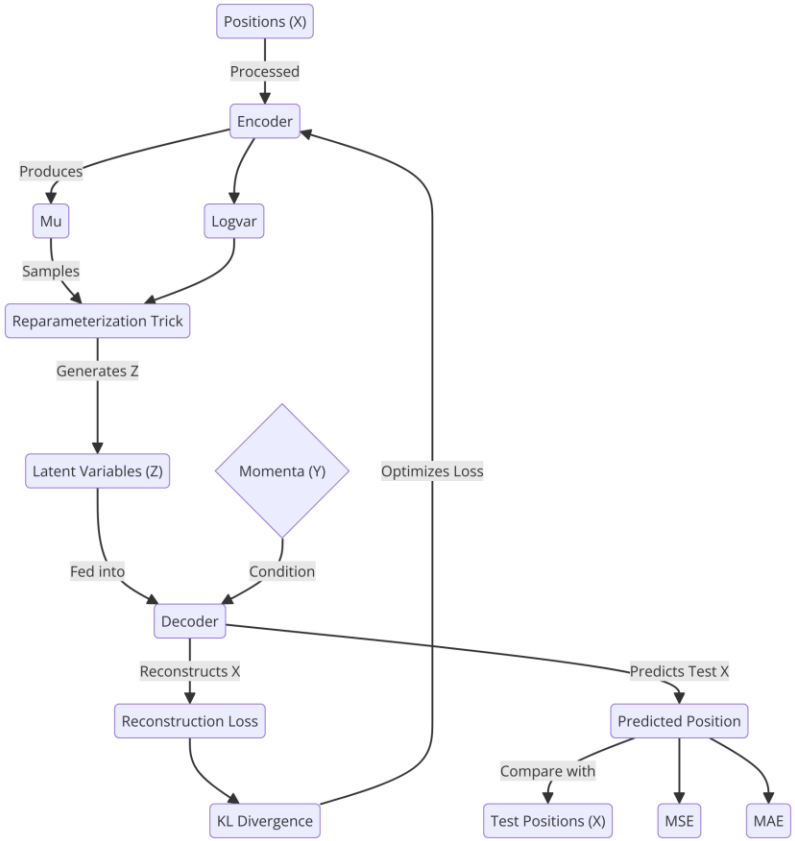
Evaluation

- 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


Positions (X)
Encoder: Processes X
Mu, Logvar
Mu, Logvar
Sample Z
Reparameterization Trick
Latent Variables (Z)
Momenta (Y)
Condition
Decoder: Reconstructs X
Predicted X

More Details



CVAE 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

Conditional Invertible Neural Network (CINN)

- Goal: Reconstruct atomic positions given their momenta as a condition
  Uses a Conditional Invertible Neural Network (CINN) architecture
- Implements encoder-decoder structure with conditional network

- Positions (x) as input and Momenta (y) as a condition
- In Testing phase, input is latent representation (z)
- Encoder: Maps positions to latent space distribution (mu, logvar)
- Conditional Network: Maps momenta to conditional vector (c)
- CINN: 8 coupling layers for invertible transformations
- Decoder: Maps transformed latent vector to reconstructed positions

CINN Architecture

1. Encode positions (x) to obtain mu and logvar
2. Sample latent vector (z) using reparameterization trick
3. Generate conditional vector (c) from momenta (y)
4. Pass z and c through CINN in reverse direction
5. Decode transformed latent vector to get reconstructed positions

TRAIN AND TESTING

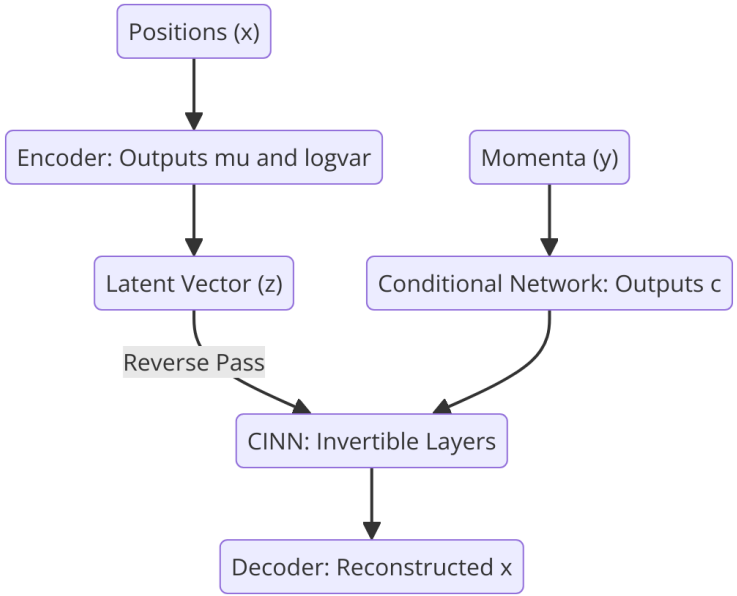
Training Process

- Loss: Reconstruction (MSE) + KL divergence (weighted)
- Latent representations collected and saved during training

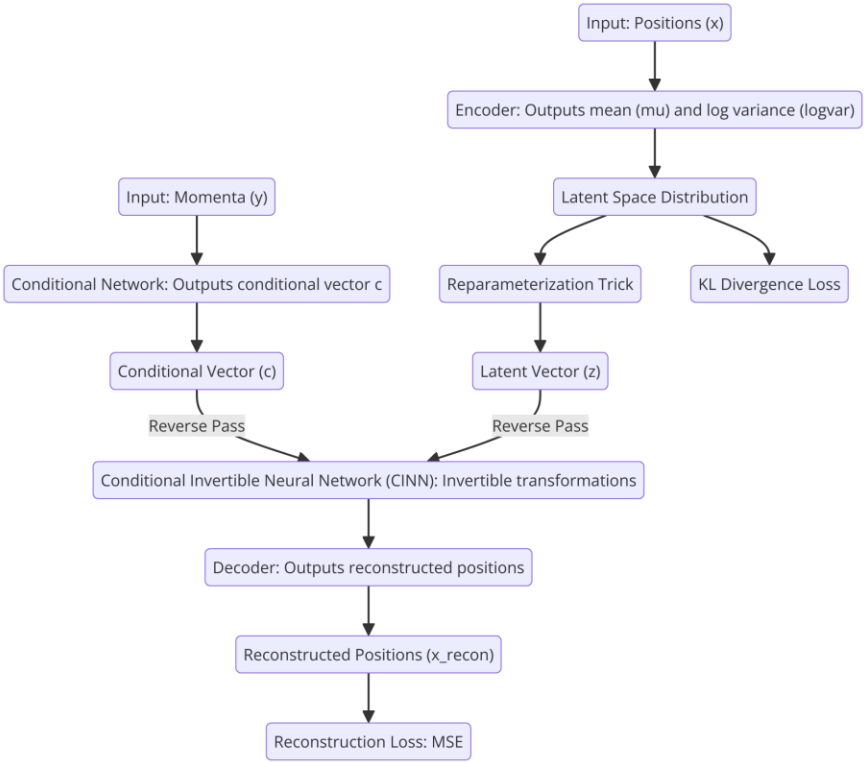
Testing Process

- Model evaluated on test set split
- Reconstruction performance measured using MSE
- Using saved latent representations
- Sample based on (z)

CINN


Positions (x)
Encoder: Outputs mu and logvar
Momenta (y)
Latent Vector (z)
Conditional Network: Outputs c
Reverse Pass
CINN: Invertible Layers
Decoder: Reconstructed x

More Detailed ...



CINN Results

Dataset	MSE
generated_cos3d_check	0.7481
random_cos3d_10000	0.4281
cei_traning_orient_1	1.0069