# Conditional Invertible Neural Networks (cINNs)

Amirhossein Ghanaatian

October 17, 2024

# Direct Mapping from Design Parameters to Latent Space

- Instead of training an invertible neural network to predict y and x with additional latent variable z, the cINN transforms x directly to a latent representation z conditioned on the observation y.
- This is achieved by using y as an additional input to each affine coupling layer in both forward and backward processes.

# Conditioning on Target Property y

- The cINN directly maps the design parameters x to a latent space z, conditioned on the target property y.
- Conditioning on y is accomplished by providing y as an extra input into each affine coupling layer of the invertible neural network, in both forward and reverse directions.
- The mapping between x and z depends on (is conditional on) the value of the target property y.
- Key Aspect: Using the target y to condition the bijective mapping between the design space x and latent space z.

# Generating Consistent Designs

- In the cINN model, the design parameters x are directly transformed into a latent representation z, conditioned on the observation of the target property y.
- By conditioning the mapping between x and z on the value of y, the cINN model learns to generate designs that are consistent with the desired target property.

# Training with Maximum Likelihood Loss Function

- The cINN model is trained using a maximum likelihood loss function, as shown in Equation 7 of the paper:

$$L(z) = -\frac{1}{2}\left(\|z\|^2 + \log\left|\det\left(J_{x\to z}\right)\right|\right)$$

  - $z$: Latent representation
  - $\|z\|^2$: Squared norm of z
  - $\det\left(J_{x\to z}\right)$: Determinant of the Jacobian matrix for the mapping from x to z

# Minimizing the Loss Function

- By minimizing this loss function, the cINN model learns to:
  - Generate a latent representation z that follows a standard Gaussian distribution.
  - Ensure that the mapping between x and z is invertible and preserves the information content.

# Inference and Generation of Design Candidates

- During inference, the cINN model can:
  - Generate design candidates by sampling from the latent space z conditioned on the desired target property y.
  - Map the samples back to the design space x using the inverse mapping learned by the model.

# Latent Variables Represent Stochasticity

- Latent Variables ($z$):
  - Represent the stochasticity or randomness inherent in the one-to-many mapping.
  - Allow the model to generate multiple valid structures corresponding to the same material properties by sampling different z values.

# Understanding the Loss Function Components

The loss function in a cINN is derived from the negative log-likelihood of the data under the model and is given by:

$$L(z) = \frac{1}{2}\|z\|^2 - \log\left|\det\left(J_{x\to z}\right)\right|$$

Where:

- $\frac{1}{2}\|z\|^2$: Represents the data fidelity term, penalizing deviations of z from the standard normal distribution. This term is always non-negative since it's based on the squared norm.
- $-\log\left|\det\left(J_{x\to z}\right)\right|$: Accounts for the volume change during the transformation from x to z. This term can be positive or negative depending on the value of the determinant.

# Why the Loss Can Be Negative

- Initial Positive Loss: At the start of training, the model parameters are often initialized randomly, and the network hasn't learned meaningful transformations yet. The Jacobian determinant $\det(J_{x\to z})$ may be close to 1, leading to $\log|\det(J_{x\to z})| \approx 0$. The loss is then dominated by the positive quadratic term $\frac{1}{2}\|z\|^2$, resulting in a positive total loss.
- Transition to Negative Loss: As training progresses, the model learns transformations that better fit the data. The Jacobian determinant can become larger than 1, making $\log|\det(J_{x\to z})|$ positive. Consequently, the term $-\log|\det(J_{x\to z})|$ becomes negative. If this negative value outweighs the positive quadratic term, the overall loss becomes negative.

# Is This Correct Behavior?

Yes, this is generally acceptable and can be expected in training cINNs. Here's why:

- Model Learning Dynamics: The negative loss indicates that the model is successfully learning transformations that increase the likelihood of the observed data under the model. The negative log-likelihood (which is being minimized) decreases, signifying improved model performance.
- Probability Density Interpretation: A negative loss means that the model assigns a higher probability density to the observed data than the initial state did. This is a sign of effective learning.

Interpretation:

- More Negative Loss Indicates Higher Likelihood: A loss of -15 suggests the model assigns a higher likelihood to the data than a loss of -5.

# Is a More Negative Loss Better?

In General, Yes:

- Lower NLL Equals Better Fit: Minimizing the negative log-likelihood means the model better fits the data.
- Comparing -15 and -5: -15 is lower than -5, indicating a better fit to the training data.

However, Be Cautious:

- Overfitting Risk: A significantly negative loss on training data may not generalize to unseen data.
- Validation Performance: Always verify that the model performs well on a validation set.

# How Negative is Too Negative?

Moderate Negative Values (-5 to -20):

- In many cases, negative loss values within this range can be expected, as they indicate a good balance between fitting the data and the model's transformation properties.

Extremely Negative Values:

- If you see losses much lower than -20 or -30, it might signal an issue:
    - Numerical instability: The Jacobian determinant might become very large or small, causing instability in the log-determinant term.
    - Overfitting: The model might be overfitting to the training data, especially if the validation loss does not decrease in a similar fashion.