Continuous generative neural networks

Giovanni S. Alberti
MaLGa – Machine Learning Genoa Center
Department of Mathematics
University of Genoa

joint with Matteo Santacesaria and Silvia Sciutto (MaLGa, University of Genoa)

CGNNs in one sentence

Continuous Generative Neural Networks (CGNNs) are a machine learning architecture that represent elements in infinite-dimensional function spaces and provide Lipschitz stability for inverse problems.

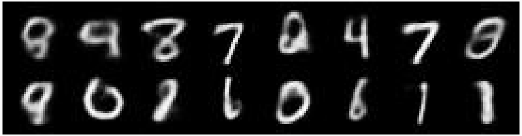
CGNNs in one formula

A CGNN $G\colon \mathbb{R}^{40} \to L^2((0,1)^2)$ generating a 40-dim manifold of handwritten digits.

CGNNs in one formula

A CGNN $\mathrm{G}\colon \mathbb{R}^{40} \to \mathrm{L}^2((0,1)^2)$ generating a 40-dim manifold of handwritten digits.

$$z \sim (\mathcal{N}(0,1))^{40} \xmapsto{\text{16 times}}$$



Outline

Generative models, inverse problems and stability

Continuous Generative Neural Networks

Generative models for inverse problems

X, Y Hilbert spaces, $\mathcal{F}\colon X \to Y$ possibly nonlinear

Generative models for inverse problems

X, Y Hilbert spaces, $\mathcal{F}\colon X \to Y$ possibly nonlinear

Key point: X and Y are typically infinite-dimensional function spaces

Generative models for inverse problems

X, Y Hilbert spaces, $\mathcal{F}\colon X \to Y$ possibly nonlinear

Key point: X and Y are typically infinite-dimensional function spaces

Classical reconstruction

Given $y = \mathcal{F}(x^\dagger) + \varepsilon$, determine $x^\dagger$ by solving

$$\arg\min_{x \in X} \{\|\mathcal{F}(x) - y\|^2 + R(x)\}.$$

Generative models for inverse problems

X, Y Hilbert spaces, $\mathcal{F}\colon X \to Y$ possibly nonlinear

Key point: X and Y are typically infinite-dimensional function spaces

Classical reconstruction

$$\text{Given } y = \mathcal{F}(x^\dagger) + \varepsilon, \text{ determine } x^\dagger \text{ by solving}$$

$$\arg\min_{x \in X} \{\|\mathcal{F}(x) - y\|^2 + R(x)\}.$$

With a generative model

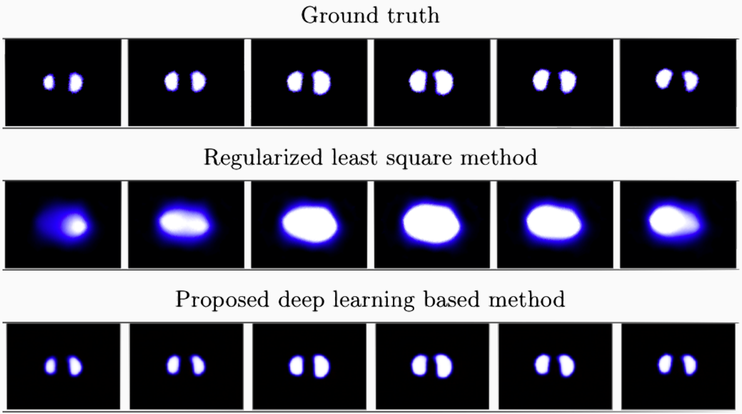
$$\text{Determine } x^\dagger = G(z^\dagger) \text{ by solving}$$

$$\arg\min_{z \in Z} \{\|\mathcal{F}(G(z)) - y\|^2\},$$

where $G\colon Z \to X$ is a generator.

UniGe | MaLGa

Example with electrical impedance tomography[1]



Classical approach:

$$y = F(x)$$

With Generative models:

$$y = F(G(z))$$

[1] Seo-Kim-Jargal-Lee-Harrach, A learning-based method for solving ill-posed nonlinear inverse problems: a simulation study of Lung EIT, 2019

A general Lipschitz stability result[2]

Theorem (Alberti-Arroyo-S. 2022)

If

- $M \subseteq X$ **finite-dimensional manifold**,
- and $\mathcal{F}|_M$ and $\mathcal{F}'(x)|_{T_x M}$, for $x \in M$, are injective,

[2] G.S. Alberti, A. Arroyo, M. Santacesaria, Inverse Problems on Low Dimensional Manifolds, 2022

A general Lipschitz stability result[2]

Theorem (Alberti-Arroyo-S. 2022)

If

- $M \subseteq X$ **finite-dimensional manifold**,
- and $\mathcal{F}|_M$ and $\mathcal{F}'(x)|_{T_x M}$, for $x \in M$, are injective,

then

$$\|x_1 - x_2\|_X \leqslant C\|\mathcal{F}(x_1) - \mathcal{F}(x_2)\|_Y, \qquad x_1, x_2 \in M.$$

UniGe | MaLGa

[2] G.S. Alberti, A. Arroyo, M. Santacesaria, Inverse Problems on Low Dimensional Manifolds, 2022

Learning the manifold M

The lower the dimension of the finite dimensional manifold M, the better the stability.

Learning the manifold M

The lower the dimension of the finite dimensional manifold M, the better the stability. M is often unknown.

Learning the manifold M

The lower the dimension of the finite dimensional manifold M, the better the stability. M is often unknown.

We can learn and approximate M as $M \approx \mathrm{Im}(G)$, for a generator $G\colon Z \to X$, with $\dim M \approx \dim Z$.

Learning the manifold M

The lower the dimension of the finite dimensional manifold M, the better the stability. M is often unknown.

We can learn and approximate M as $M \approx \mathrm{Im}(G)$, for a generator $G\colon Z \to X$, with $\dim M \approx \dim Z$.

Pros: higher stability, more accurate modeling, less computations.

Cons: missing theory!

Outline

Generative models, inverse problems and stability

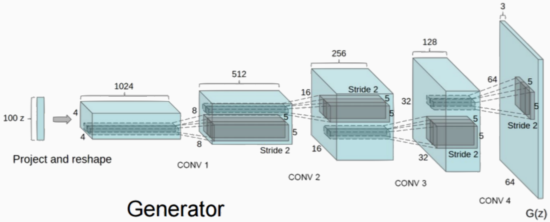
Continuous Generative Neural Networks

How to build a CGNN: the discrete case

Many architectures: fully connected, convolutional, transformers, etc.

How to build a CGNN: the discrete case

Many architectures: fully connected, convolutional, transformers, etc.



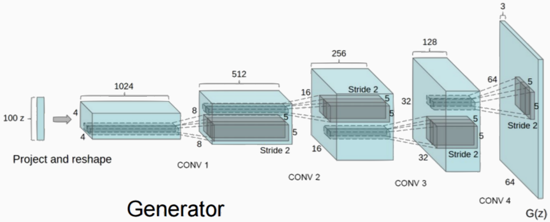
Fully Connected layer

$$y = \sigma(Fx + b)$$

UniGe | MaLGa

How to build a CGNN: the discrete case

Many architectures: fully connected, convolutional, transformers, etc.



Fully Connected layer

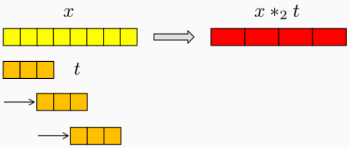
$$y = \sigma(Fx + b)$$

Convolutional layer (filters t_i^k)

$$(f_{out})_k = \sigma\left(\sum_{i=1}^{c} (f_{in})_i *_s t_i^k + b^k\right)$$

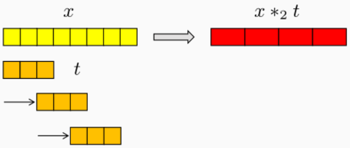
Strided continuous convolution

Discrete case



Strided continuous convolution

Discrete case



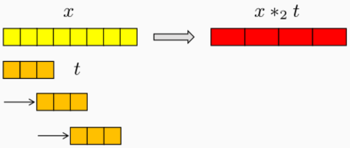
Continuous case

$$V_j \xrightarrow[\text{conv.}]{\Psi} L^2([0,1]) \xrightarrow[\text{proj.}]{P_{V_{j-1}}} V_{j-1},$$

- Ψ is a continuous convolution
- $\cdots \subseteq V_{j-1} \subseteq V_j \subseteq V_{j+1} \subseteq \cdots$: scale spaces of a wavelet multiresolution analysis

Strided continuous convolution

Discrete case



Continuous case

$$V_j \xrightarrow[\text{conv.}]{\Psi} L^2([0,1]) \xrightarrow[\text{proj.}]{P_{V_{j-1}}} V_{j-1},$$

- Ψ is a continuous convolution
- $\cdots \subseteq V_{j-1} \subseteq V_j \subseteq V_{j+1} \subseteq \cdots$: scale spaces of a wavelet multiresolution analysis

Easy implementation. Discrete convolution (almost) for the wavelet coefficients

Discrete and Continuous Generator structure in 1D [3]

Discrete Generator

$$G : \mathbb{R}^\eta \xrightarrow[\text{f.c.}]{F\,\cdot\,+b} (\mathbb{R}^{\alpha_1})^{c_1} \xrightarrow[\text{non lin.}]{\sigma} (\mathbb{R}^{\alpha_1})^{c_1}$$

$$\xrightarrow[\text{conv.}]{\Psi_2} (\mathbb{R}^{\alpha_2})^{c_2} \xrightarrow[\text{non lin.}]{\sigma} (\mathbb{R}^{\alpha_2})^{c_2} \dots$$

$$\dots \xrightarrow[\text{conv.}]{\Psi_L} \mathbb{R}^{\alpha_L} \xrightarrow[\text{non lin.}]{\sigma} \mathbb{R}^{\alpha_L}$$

$$c_1 > c_2 > \dots > c_L$$

$$\alpha_1 < \alpha_2 < \dots < \alpha_L$$

[3] J. Bruna, S. Mallat, Invariant Scattering Convolution Networks, 2012
N. Kovachki et al., Neural Operator: Learning Maps Between Function Spaces, 2021
A. Habring, M. Holler, A generative variational model for inverse problems in imaging, 2022
A.E. Khorashadizadeh, et al., FunkNN: Neural Interpolation for Functional Generation, 2022

Discrete and Continuous Generator structure in 1D [3]

Discrete Generator

$$G : \mathbb{R}^{\eta} \xrightarrow[\text{f.c.}]{F\,\cdot\,+b} (\mathbb{R}^{\alpha_1})^{c_1} \xrightarrow[\text{non lin.}]{\sigma} (\mathbb{R}^{\alpha_1})^{c_1}$$

$$\xrightarrow[\text{conv.}]{\Psi_2} (\mathbb{R}^{\alpha_2})^{c_2} \xrightarrow[\text{non lin.}]{\sigma} (\mathbb{R}^{\alpha_2})^{c_2} \dots$$

$$\dots \xrightarrow[\text{conv.}]{\Psi_L} \mathbb{R}^{\alpha_L} \xrightarrow[\text{non lin.}]{\sigma} \mathbb{R}^{\alpha_L}$$

$$c_1 > c_2 > \dots > c_L$$

$$\alpha_1 < \alpha_2 < \dots < \alpha_L$$

Continuous Generator

$$G : \mathbb{R}^{\eta} \xrightarrow[\text{f.c.}]{F\,\cdot\,+b} (V_{j_1})^{c_1} \xrightarrow[\text{non lin.}]{\sigma} (V_{j_1})^{c_1}$$

$$\xrightarrow[\text{conv.}]{\Psi_2} (V_{j_2})^{c_2} \xrightarrow[\text{non lin.}]{\sigma} (V_{j_2})^{c_2} \dots$$

$$\dots \xrightarrow[\text{conv.}]{\Psi_L} V_{j_L} \xrightarrow[\text{non lin.}]{\sigma} V_{j_L}$$

$$c_1 > c_2 > \dots > c_L$$

$$j_1 < j_2 < \dots < j_L$$

---

[3] J. Bruna, S. Mallat, Invariant Scattering Convolution Networks, 2012
N. Kovachki et al., Neural Operator: Learning Maps Between Function Spaces, 2021
A. Habring, M. Holler, A generative variational model for inverse problems in imaging, 2022
A.E. Khorashadizadeh, et al., FunkNN: Neural Interpolation for Functional Generation, 2022

Continuous Generator structure in 1D: Formalization

$$
\begin{aligned}
G : \mathbb{R}^{\eta} &\xrightarrow[\text{f.c.}]{F\ \cdot\ +b} (V_{j_1})^{c_1} \xrightarrow[\text{non lin.}]{\sigma} (L^2([0,1]))^{c_1} \xrightarrow[\text{proj.}]{P_{(V_{j_1})^{c_1}}} (V_{j_1})^{c_1} \\
&\xrightarrow[\text{conv.}]{\Psi_2} (L^2([0,1]))^{c_2} \xrightarrow[\text{proj.}]{P_{(V_{j_2})^{c_2}}} (V_{j_2})^{c_2} \xrightarrow[\text{non lin.}]{\sigma} (L^2([0,1]))^{c_2} \\
&\xrightarrow[\text{proj.}]{P_{(V_{j_2})^{c_2}}} (V_{j_2})^{c_2} \quad \dots \quad \xrightarrow[\text{conv.}]{\Psi_L} L^2([0,1]) \xrightarrow[\text{proj.}]{P_{V_{j_L}}} V_{j_L} \\
&\xrightarrow[\text{non lin.}]{\sigma} L^2([0,1]) \xrightarrow[\text{proj.}]{P_{V_{j_L}}} V_{j_L}
\end{aligned}
$$

Main features

Discretization invariance

[4] Puthawala, Kothari, Lassas, Dokmanić, de Hoop, Globally injective ReLU networks, 2020
Hagemann, Neumayer, Stabilizing invertible neural networks using mixture models, 2021

Main features

Discretization invariance

Injectivity[4]

Theorem (G.S.Alberti, M. S., S. Sciutto, 2022)

Assume F injective, σ injective and linear independence of convolutional filters.
Then G is injective

[4] Puthawala, Kothari, Lassas, Dokmanić, de Hoop, Globally injective ReLU networks, 2020
Hagemann, Neumayer, Stabilizing invertible neural networks using mixture models, 2021

Main features

Discretization invariance

Injectivity[4]

Theorem (G.S.Alberti, M. S., S. Sciutto, 2022)

Assume F injective, σ injective and linear independence of convolutional filters. Then G is injective

Lipschitz stability for inverse problems

Theorem (G.S.Alberti, M. S., S. Sciutto, 2022)

Assume G as above, set $\mathrm{Im}(G) = M$. Then

$$\|x - y\|_X \leqslant C\|\mathcal{F}(x) - \mathcal{F}(y)\|_Y, \qquad x, y \in M.$$

[4] Puthawala, Kothari, Lassas, Dokmanić, de Hoop, Globally injective ReLU networks, 2020
Hagemann, Neumayer, Stabilizing invertible neural networks using mixture models, 2021

Conclusions

Limitations

- Training protocols (we used variational autoencoders)
- Architecture (convolutional; we only used the approximation coefficients)
- Manifold learning (only one chart)

Conclusions

Limitations

- Training protocols (we used variational autoencoders)
- Architecture (convolutional; we only used the approximation coefficients)
- Manifold learning (only one chart)

Future work

- Multiple generators for nontrivial topologies (with J. Hertrich)
- Different architectures
- How does the training affect the reconstruction performances?

[Continuous Generative Neural Networks, preprint arXiv:2205.14627]

UniGe | MaLGa