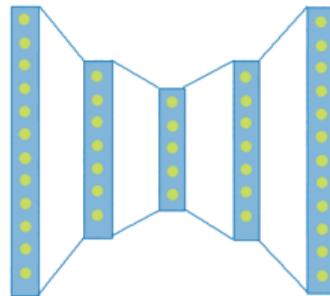




DaSCI

Instituto Andaluz de Investigación en
Data Science and Computational Intelligence



UNIVERSIDAD
DE GRANADA

Autoencoders: An Overview and Applications

David Charte

fdavidcl@ugr.es

SOMACHINE 2020

Contents

Motivation

Definition

Example

Applications

Current situation and future

Representation of data sets

Let us organize our data onto a table:

	variables →			
	A_1	A_2	...	A_n
samples ↓	v_{11}	v_{12}		v_{1n}
	v_{21}	v_{22}		v_{2n}
	v_{31}	v_{32}		v_{3n}
	⋮		⋮	
	v_{k1}	v_{k2}		v_{kn}

Representation of data sets

Let us organize our data onto a table:

	variables →			
	A_1	A_2	...	A_n
samples ↓	v_{11}	v_{12}		v_{1n}
	v_{21}	v_{22}		v_{2n}
	v_{31}	v_{32}		v_{3n}
	⋮		⋮	
	v_{k1}	v_{k2}		v_{kn}

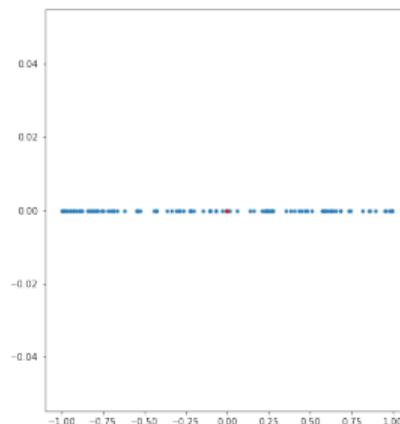
- ◆ Dimension is the number of variables (n). Can it pose a problem?
- ◆ Are variables related?

Curse of dimensionality

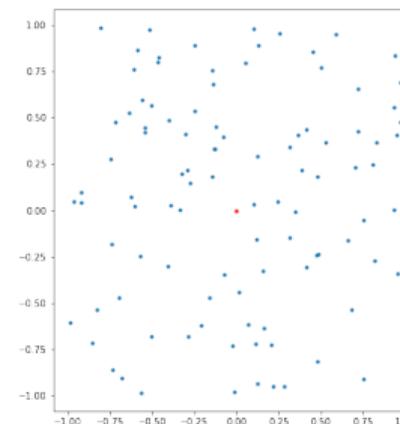
What problems can arise as dimension increases?

Curse of dimensionality

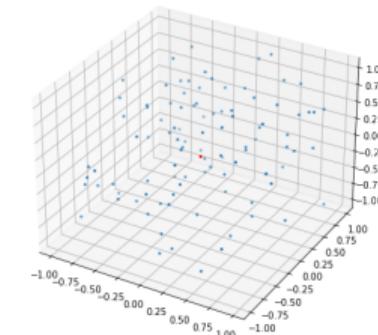
What problems can arise as dimension increases?



1D



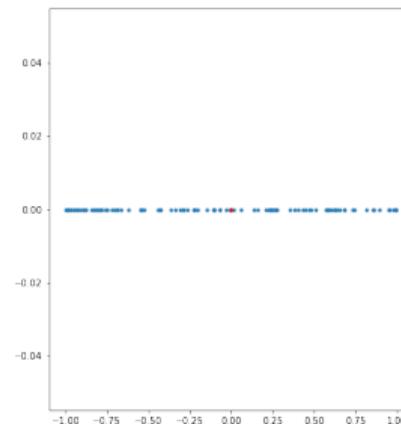
2D



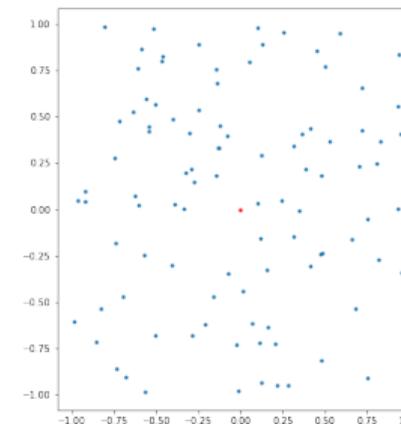
3D

Curse of dimensionality

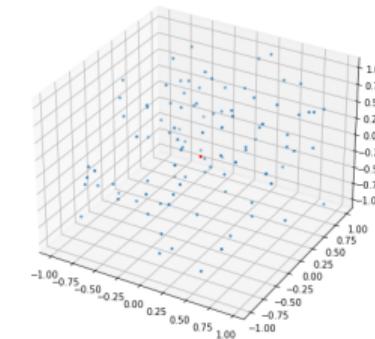
What problems can arise as dimension increases?



1D



2D



3D

Less fraction of the space is covered by the same amount of data

Curse of dimensionality

How do distances behave when dimension is high?

¹C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” in *International conference on database theory*, pp. 420–434, Springer, 2001

Curse of dimensionality

How do distances behave when dimension is high?

\mathcal{F} being a probability distribution of k points. We define the following values:

- ◆ $D_{\max}^p_n = \max\{\|x\|_p : x \text{ is a sample of } \mathcal{F}^n\}$
- ◆ $D_{\min}^p_n = \min\{\|x\|_p : x \text{ is a sample of } \mathcal{F}^n\}$

¹C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” in *International conference on database theory*, pp. 420–434, Springer, 2001

Curse of dimensionality

How do distances behave when dimension is high?

\mathcal{F} being a probability distribution of k points. We define the following values:

- ◆ $D_{\max}^p_n = \max\{\|x\|_p : x \text{ is a sample of } \mathcal{F}^n\}$
- ◆ $D_{\min}^p_n = \min\{\|x\|_p : x \text{ is a sample of } \mathcal{F}^n\}$

Theorem ¹

There exists a positive constant C_p depending solely on p and verifying:

$$C_p \leq \lim_{n \rightarrow +\infty} E \left[\frac{D_{\max}^p_n - D_{\min}^p_n}{n^{\frac{1}{p} - \frac{1}{2}}} \right] \leq (k - 1)C_p$$

¹C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” in *International conference on database theory*, pp. 420–434, Springer, 2001

Consequences

As $n \rightarrow \infty$, the difference between the maximum and the minimum distances to the reference point:

- ◆ tends to ∞ in the case of the Manhattan distance ($p = 1$)
- ◆ tends to a constant for the Euclidean distance ($p = 2$)
- ◆ tends to 0 for any p-distance $p > 2$

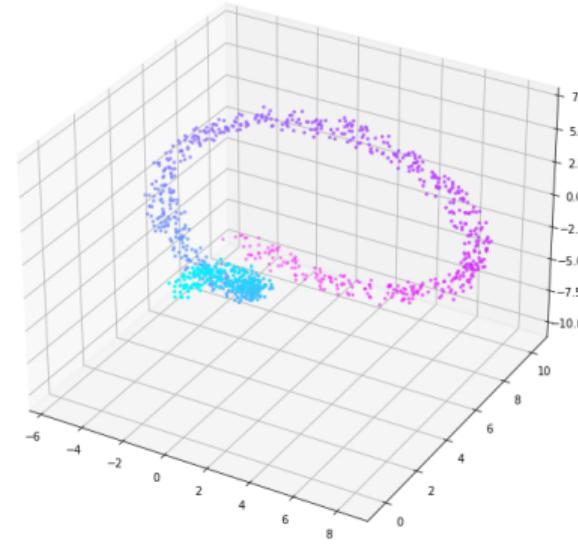
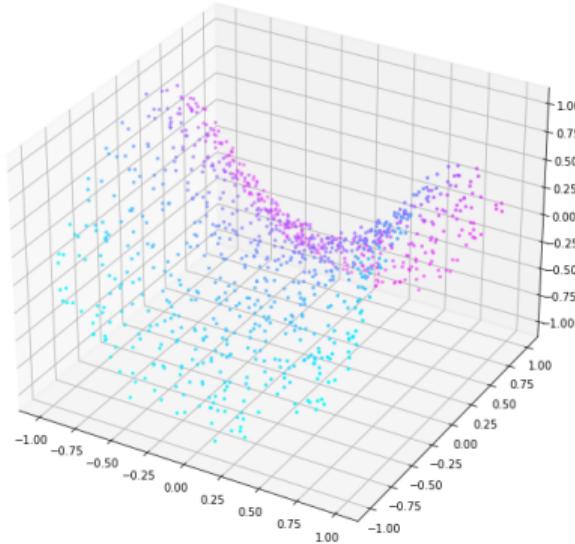
Consequences

As $n \rightarrow \infty$, the difference between the maximum and the minimum distances to the reference point:

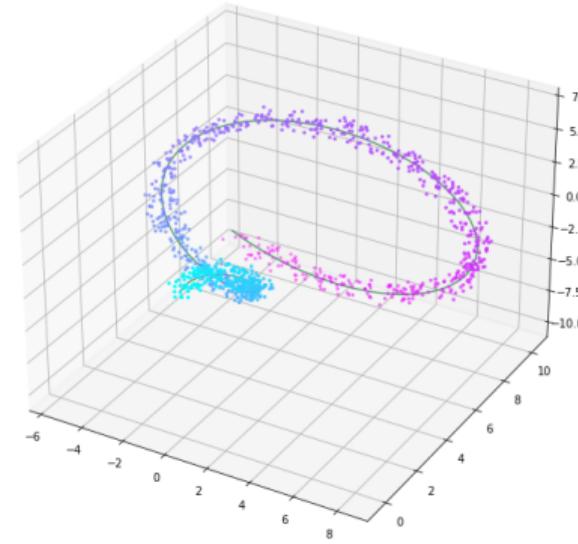
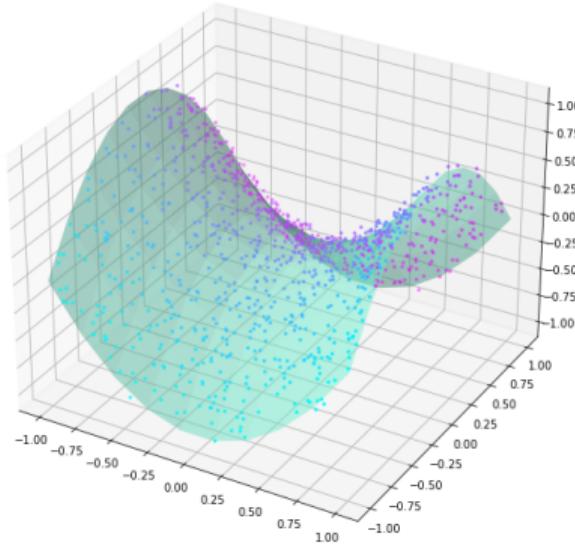
- ◆ tends to ∞ in the case of the Manhattan distance ($p = 1$)
- ◆ tends to a constant for the Euclidean distance ($p = 2$)
- ◆ tends to 0 for any p-distance $p > 2$

Most distance metrics lose significance as dimension increases

Data in a manifold



Data in a manifold



Data is usually not distributed uniformly, but may be found on a manifold



Alternative representations for problem solving

A better encoding of data samples can simplify:

- ◆ Supervised classification
- ◆ Unsupervised classification/Semantic hashing
- ◆ Anomaly detection
- ◆ Artificial data generation
- ◆ Data compression



Alternative representations for problem solving

A better encoding of data samples can simplify:

- ◆ Supervised classification
- ◆ Unsupervised classification/Semantic hashing
- ◆ Anomaly detection
- ◆ Artificial data generation
- ◆ Data compression

Learning a representation of instances can serve as a tool to solve other problems

This task can be called feature learning, feature extraction, representation learning...

Unsupervised problem

Supervised tasks involve knowing the solution of a (training) set of instances.

Unsupervised problem

Supervised tasks involve knowing the solution of a (training) set of instances.

In our case, the solution is generally **unknown** even for training data.

Unsupervised problem

Supervised tasks involve knowing the solution of a (training) set of instances.

In our case, the solution is generally **unknown** even for training data.

The best codifications are not known during learning

Contents

Motivation

Definition

Example

Applications

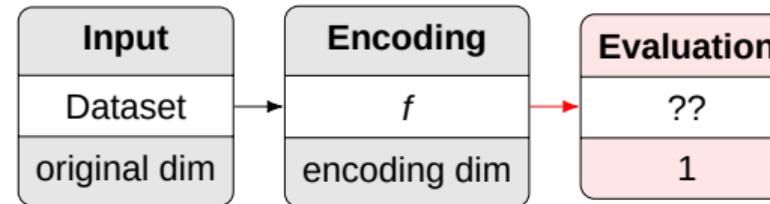
Current situation and future

Input and output

Our encoding learning problem consists in finding a mapping $f : X \rightarrow Z$

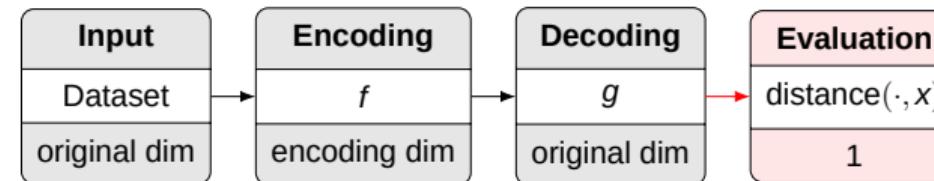
- ◆ The **input** is each sample $x \in X$
- ◆ The **output** is the encoding $z \in Z$

What is the objective function?



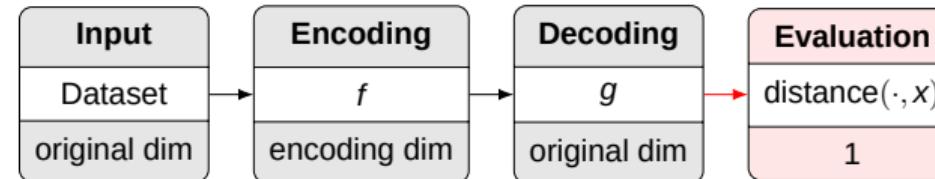
Supervising the output

Since the only information we have is the input data, use it to provide a fitness score for the encoding: find another mapping $g : Z \rightarrow X$ so that $\|g(f(x)) - x\|$ is minimized.



Supervising the output

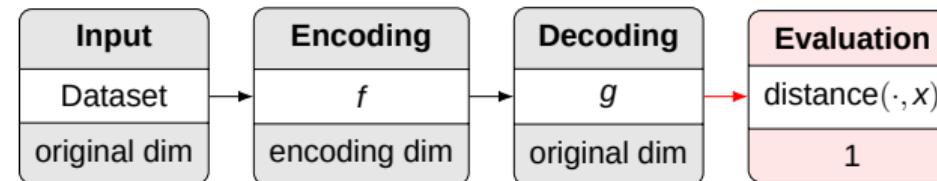
Since the only information we have is the input data, use it to provide a fitness score for the encoding: find another mapping $g : Z \rightarrow X$ so that $\|g(f(x)) - x\|$ is minimized.



- ◆ Each code preserves significant individual information (position within the manifold)
- ◆ Common information (manifold structure) is held by the network weights

Supervising the output

Since the only information we have is the input data, use it to provide a fitness score for the encoding: find another mapping $g : Z \rightarrow X$ so that $\|g(f(x)) - x\|$ is minimized.



- ◆ Each code preserves significant individual information (position within the manifold)
- ◆ Common information (manifold structure) is held by the network weights

An **encoder-decoder architecture** can use the inputs as an evaluation tool

Defining f and g

Are f and g any function? No, but they **may approximate any in $C(I^n)$.**

Theorem (universal approximation)²

There is a single hidden layer feedforward network that approximates any measurable function to any desired degree of accuracy on some compact set K .

²K. Hornik, M. Stinchcombe, H. White, et al., “Multilayer feedforward networks are universal approximators.,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989

Defining f and g

Are f and g any function? No, but they **may approximate any** in $C(I^n)$.

Theorem (universal approximation)²

There is a single hidden layer feedforward network that approximates any measurable function to any desired degree of accuracy on some compact set K .

This result can be extended to:

- ◆ multi-layer networks
- ◆ other activation functions

²K. Hornik, M. Stinchcombe, H. White, et al., “Multilayer feedforward networks are universal approximators.,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989

Defining f and g

$f = f_k \circ \dots \circ f_1$, where $f_i(x) = \sigma_i(W_i x + b_i)$

$g = g_{k'} \circ \dots \circ g_1$, where $g_i(x) = \sigma'_i(W'_i x + b'_i)$

Defining f and g

$f = f_k \circ \cdots \circ f_1$, where $f_i(x) = \sigma_i(W_i x + b_i)$

$g = g_{k'} \circ \cdots \circ g_1$, where $g_i(x) = \sigma'_i(W'_i x + b'_i)$

- ◆ W_i, W'_i, b_i, b'_i are **learnable parameters**

Defining f and g

$f = f_k \circ \dots \circ f_1$, where $f_i(x) = \sigma_i(W_i x + b_i)$

$g = g_{k'} \circ \dots \circ g_1$, where $g_i(x) = \sigma'_i(W'_i x + b'_i)$

- ◆ W_i, W'_i, b_i, b'_i are learnable parameters
- ◆ σ_i, σ'_i are **activation (nonlinear) functions**

Defining f and g

$f = f_k \circ \dots \circ f_1$, where $f_i(x) = \sigma_i(W_i x + b_i)$

$g = g_{k'} \circ \dots \circ g_1$, where $g_i(x) = \sigma'_i(W'_i x + b'_i)$

- ◆ W_i, W'_i, b_i, b'_i are learnable parameters
- ◆ σ_i, σ'_i are activation (nonlinear) functions

The encoder and decoder are compositions of matrix products and nonlinearities

(This is good for parallel computation!)

Defining f and g

$f = f_k \circ \dots \circ f_1$, where $f_i(x) = \sigma_i(W_i x + b_i)$

$g = g_{k'} \circ \dots \circ g_1$, where $g_i(x) = \sigma'_i(W'_i x + b'_i)$

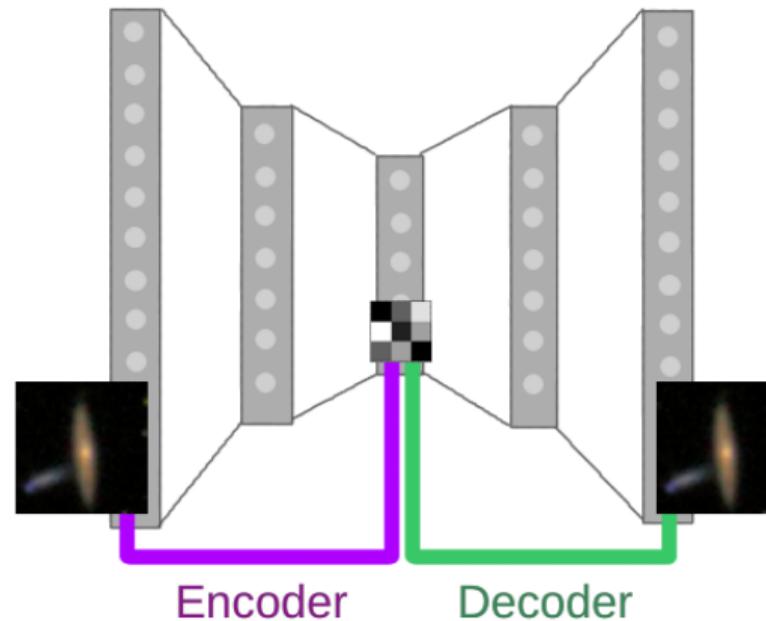
- ◆ W_i, W'_i, b_i, b'_i are learnable parameters
- ◆ σ_i, σ'_i are activation (nonlinear) functions

The encoder and decoder are compositions of matrix products and nonlinearities

(This is good for parallel computation!)

(What about convolutions? They can be expressed as matrix products)

Autoencoder architecture



Contents

Motivation

Definition

Example

Applications

Current situation and future

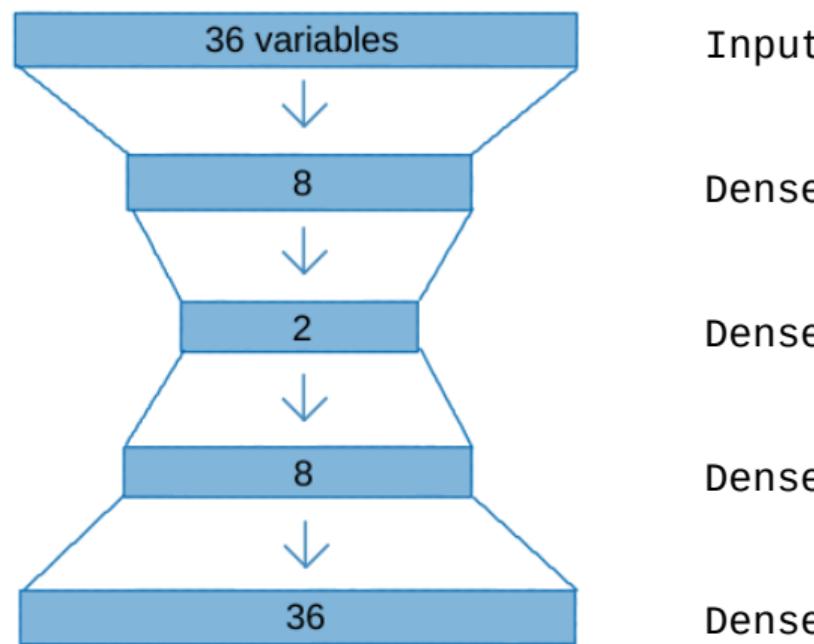
Example: satellite imagery

Objective: group pixels of an image according to the **type of land** they contain.
 For each pixel we have multi-spectral values for the neighboring 8 pixels and itself:
 in total, 9 pixels \times 4 bands = **36 features**.

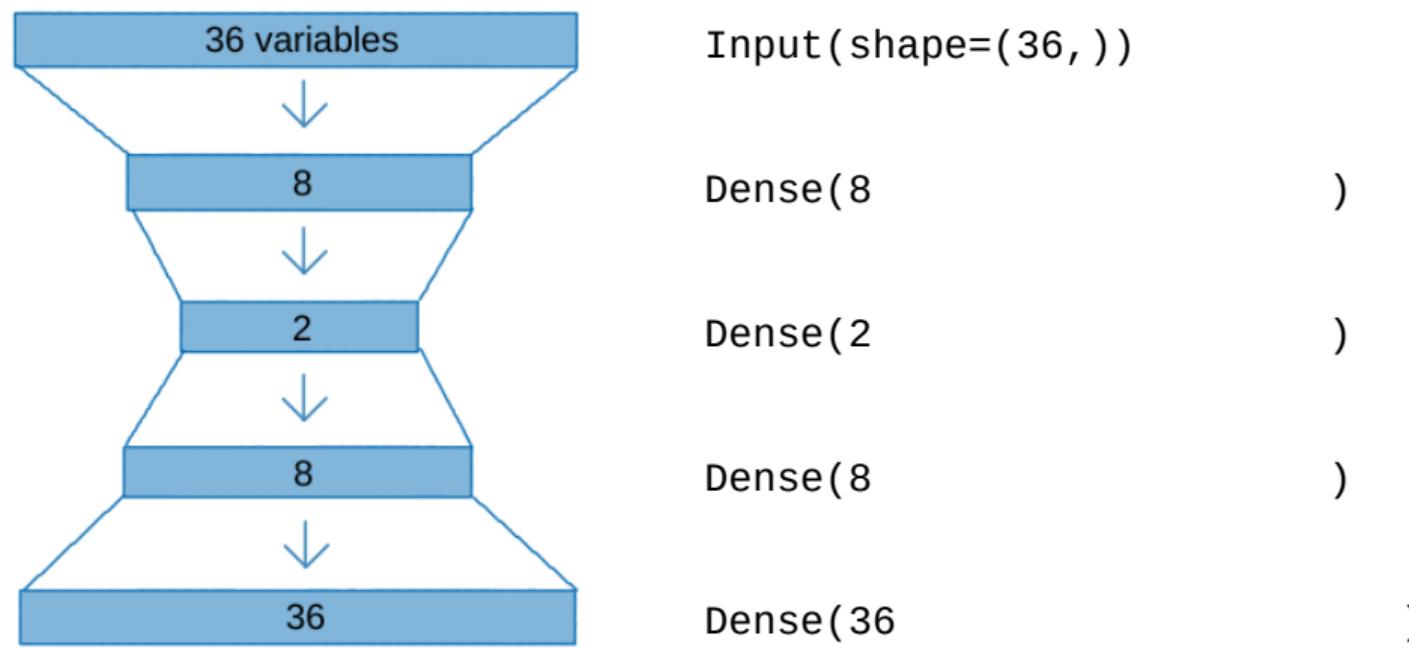
	Aattr	Battr	Cattr	Dattr	Eattr	Fattr	A1attr	B2attr	C3attr	D4attr	E5attr
1	0.117596	1.241362	1.184036	0.815302	-0.158561	1.256483	1.193546	0.818486	-0.141965	0.879481	0.670010
2	-1.205362	-1.249654	-0.077532	0.444886	-0.895959	-0.447579	-0.786760	-0.554203	-0.364672	0.092157	-0.051291
3	0.779075	0.148811	0.042617	-0.243030	0.800057	0.164136	0.053370	-0.448612	0.154978	-0.345245	-0.712483
4	1.146564	0.585831	0.342991	0.021553	0.947536	0.601074	0.353416	0.026550	1.788164	1.010702	0.910444
5	-0.764376	-1.162250	-0.137607	0.180303	-0.969698	-1.146681	-0.126658	0.184937	-0.735851	-1.132569	-0.111399
6	-0.470385	-0.725230	-1.158877	-1.036778	-0.232300	-0.534967	-1.396853	-1.029365	-0.216201	-0.170284	-1.133242
7	-1.425855	-1.337058	-1.279026	-0.719279	-1.190918	-1.321457	-1.506872	-0.712591	-1.403973	-1.307530	-1.253459
8	-0.911371	-1.249654	-1.939848	-1.513027	-1.190918	-1.496232	-2.106965	-1.715710	-0.884323	-1.220050	-1.914651
9	-1.719846	-2.036290	1.184036	2.349882	-1.707096	-2.239028	1.193546	2.560745	-1.700916	-2.138594	1.210986
10	-0.764376	-0.856336	0.042617	0.497802	-0.748479	-1.102987	0.293407	0.924077	-0.735851	-1.088829	0.309360
11	0.779075	0.323619	-0.498055	-0.825112	0.505098	0.164136	-0.486714	-0.659795	0.154978	0.004676	-0.712483
12	1.146564	0.585831	0.102692	-0.137197	1.168756	0.601074	0.653462	-0.131838	0.971571	0.835741	0.670010
13	0.999568	0.848043	0.342991	0.180303	1.021276	0.863238	0.653462	0.026550	1.045807	0.704520	0.129035
14	-0.176394	1.197660	0.883663	0.603635	-0.158561	0.994319	1.193546	0.765690	-0.141965	1.010702	0.910444
15	1.367057	0.804341	0.403065	0.233219	1.684934	1.387565	0.713472	0.448916	1.713929	1.404364	0.970552
16	-0.249892	1.285064	1.063887	0.921135	0.062659	1.300177	1.373574	1.082464	0.080742	1.316883	1.391311
17	-0.249892	1.285064	1.063887	0.709468	0.062659	1.125401	1.373574	0.924077	-0.216201	1.141922	1.391311
18	-0.470385	0.848043	0.643364	0.391969	-0.748479	0.688462	0.653462	0.396120	-0.735851	0.529559	0.670010

Let us study a simple solution using Keras and an autoencoder.

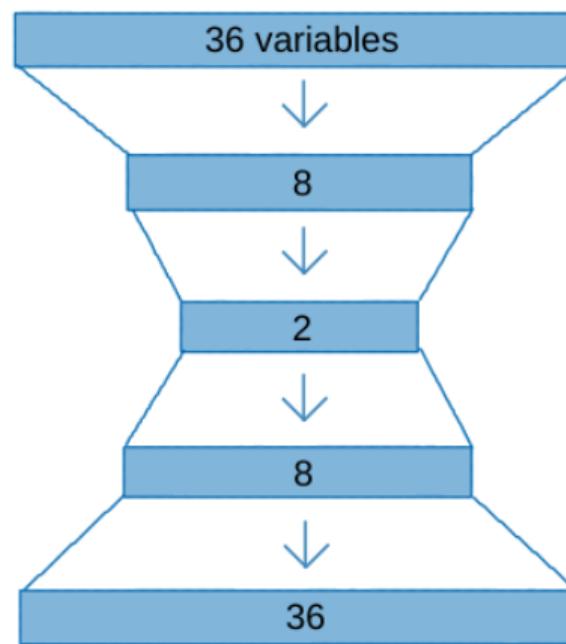
Example: satellite imagery



Example: satellite imagery



Example: satellite imagery



Input(shape=(36,))

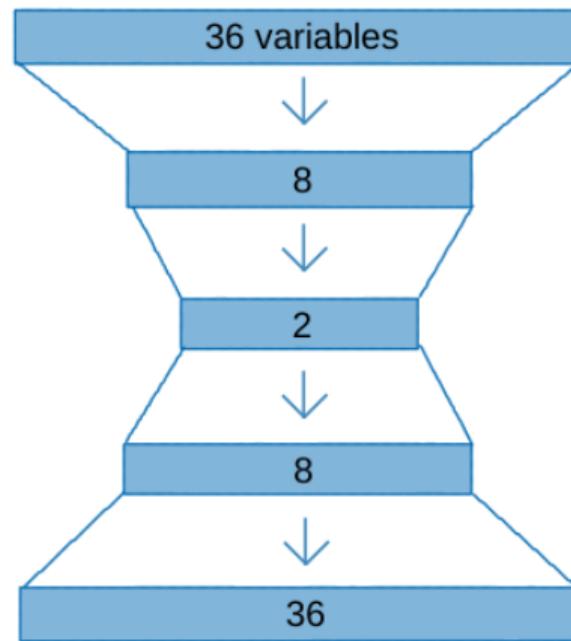
Dense(8, activation='selu')

Dense(2, activation='selu')

Dense(8, activation='selu')

Dense(36, activation='linear')

Example: satellite imagery



```
il = Input(shape=(36,))
```

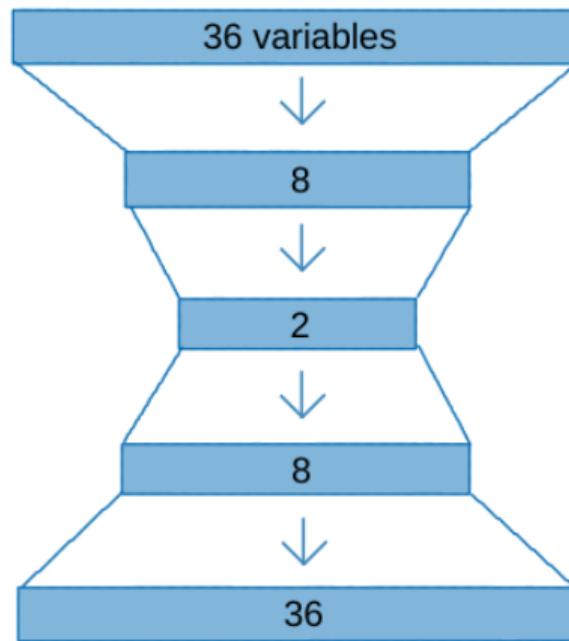
```
encoder = Sequential([  
    Dense(8, activation='selu'),
```

```
    Dense(2, activation='selu')])
```

```
decoder = Sequential([  
    Dense(8, activation='selu'),
```

```
    Dense(36, activation='linear')])
```

Example: satellite imagery



```
il = Input(shape=(36,))

encoder = Sequential([
Dense(8, activation='selu'),
Dense(2, activation='selu')])

decoder = Sequential([
Dense(8, activation='selu'),
Dense(36, activation='linear')])

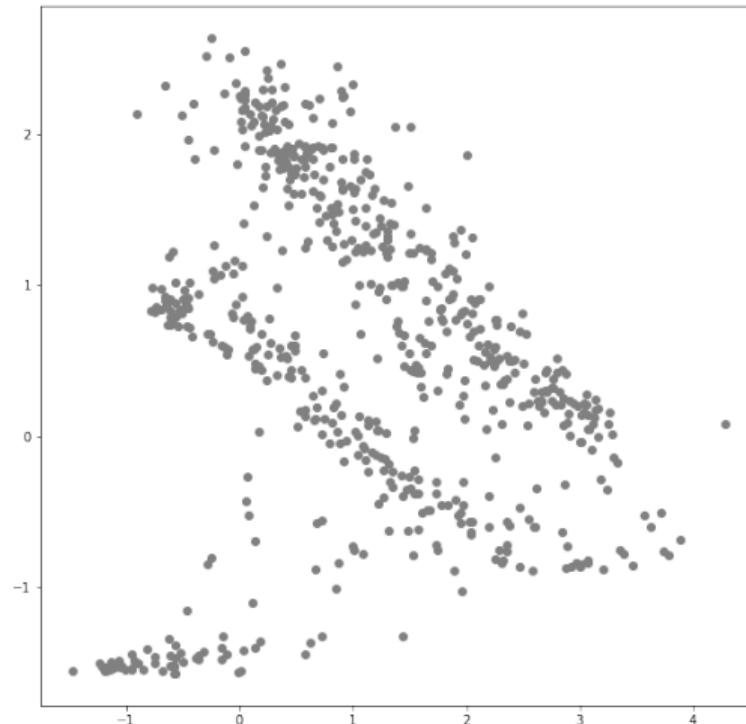
autoencoder =
Model(il, decoder(encoder(il)))
```

Test encodings

After training and selecting a model, we can obtain codes for other samples.

Codes can be used for visualization and clustering.

The (SGD-based) optimizer can produce different results in different runs.

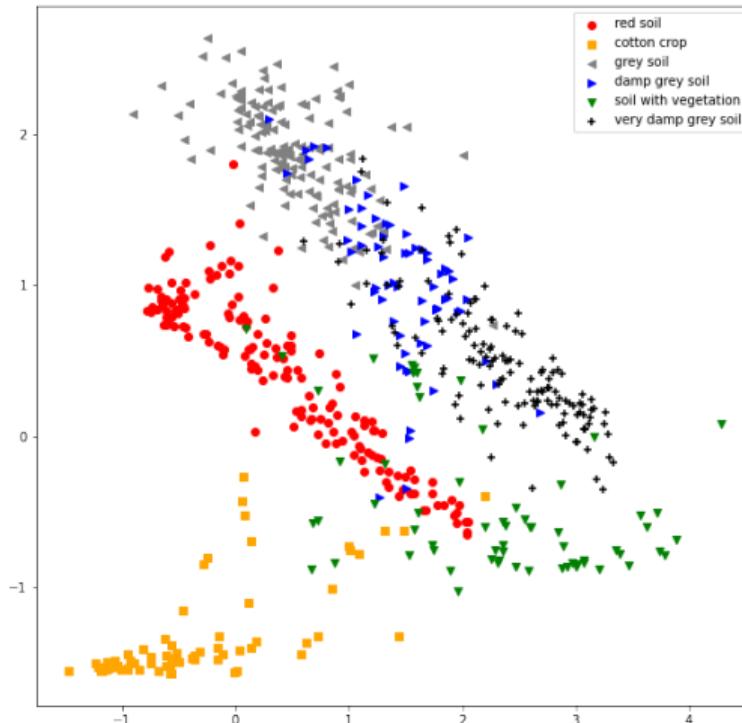


Test encodings

After training and selecting a model, we can obtain codes for other samples.

Codes can be used for visualization and clustering.

The (SGD-based) optimizer can produce different results in different runs.



Contents

Motivation

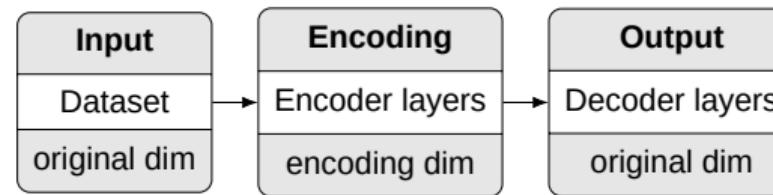
Definition

Example

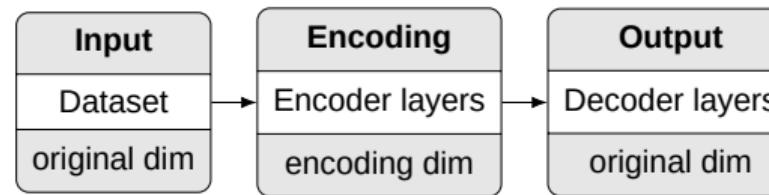
Applications

Current situation and future

Reminder: general structure



Reminder: general structure



We need 3 Keras models:

- ◆ encoder: layers up to encoding → encode data
`Model(data_input, encoding)`
- ◆ decoder: layers from encoding → generate new data
`Model(code_input, reconstruction)`
- ◆ autoencoder: composition of models → train/reconstruct
`Model(data_input, decoder(encoder(data_input)))`

Noise reduction: strategy

◆ Objective

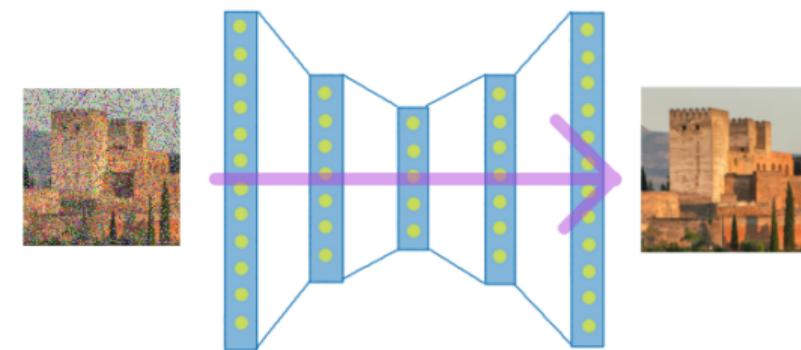
Improve manifold modeling and reduce noise

◆ Method

Input noisy data and evaluate over clean data

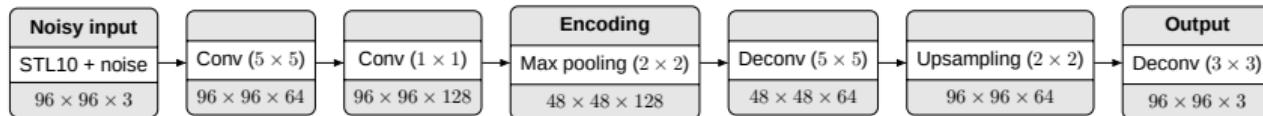
◆ Reference

Vincent et al. *Extracting and composing robust features with denoising autoencoders*. In Proceedings of the 25th international conference on Machine learning (pp. 1096-1103).



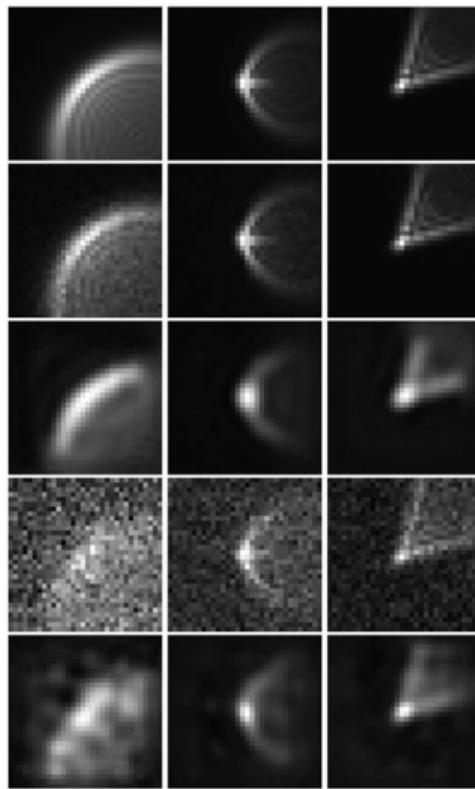
Convolutional denoising autoencoders for noise reduction

Case study: **denoising a 10-class object classification dataset**



Convolutional denoising autoencoders for noise reduction

Figure 6.



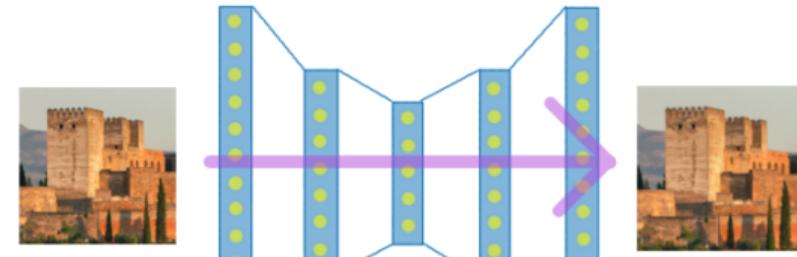
From the literature: **point spread function modelling**

Jia, P., Li, X., Li, Z., Wang, W., & Cai, D. (2020). Point spread function modelling for wide-field small-aperture telescopes with a denoising autoencoder. *Monthly Notices of the Royal Astronomical Society*, 493(1), 651-660.

Anomaly detection: strategy

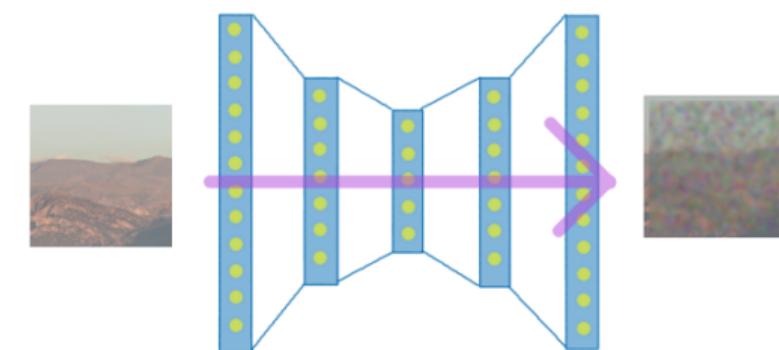
◆ Conjecture

An autoencoder trained on *normal* data cannot reconstruct anomalous data



◆ Method

Anomaly score = reconstruction error



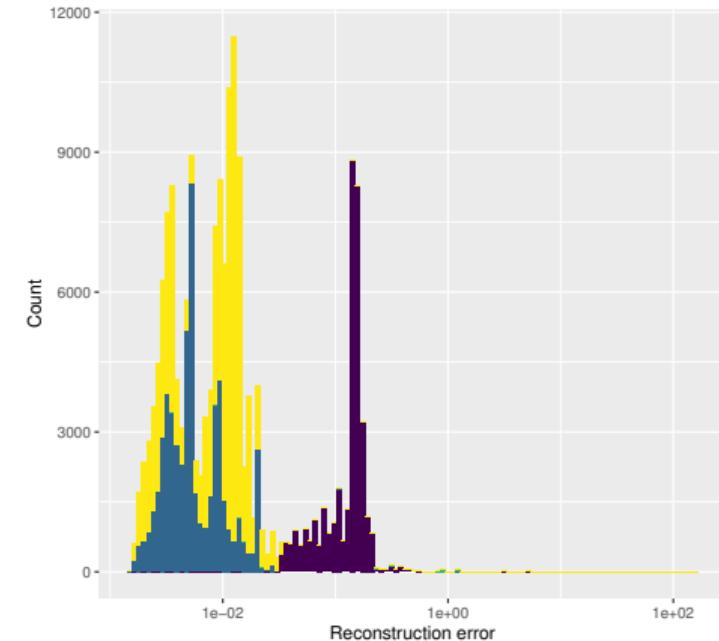
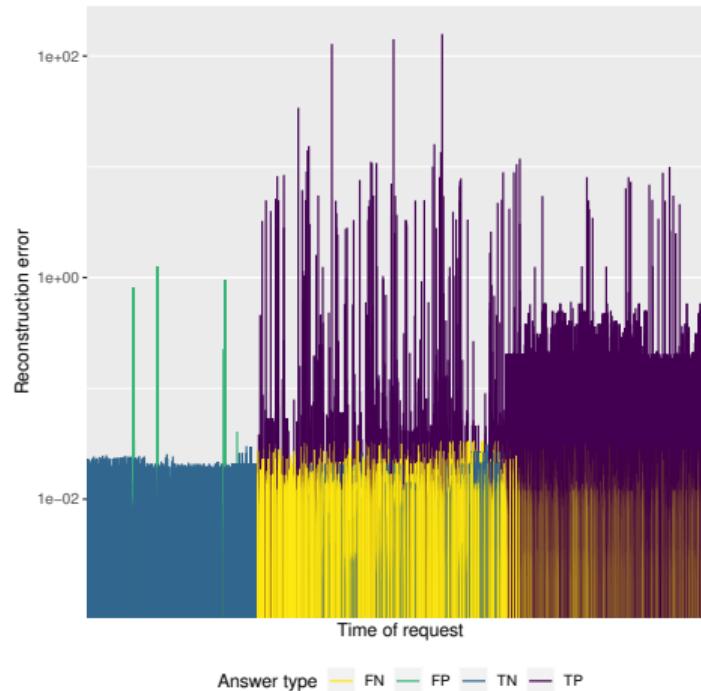
◆ Reference

Sakurada et al. *Anomaly detection using autoencoders with nonlinear dimensionality reduction*. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (pp. 4-11).



Denoising autoencoders for anomaly detection

Case study: attack detection in networks



Denoising autoencoders for anomaly detection

From the literature: **gravitational wave detection**

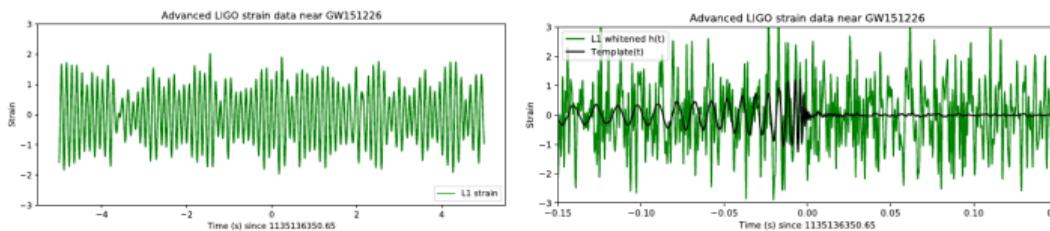


Fig. 1. (left) Example of a time series containing a GWs signal immersed in LIGO noise (ID GW151226). Since data are dominated by noise, it is not possible to see the signal without some signal processing. (right) Whitened time series containing a GWs signal (ID GW151226) and the best matching template identified in the LIGO search pipeline.

Corizzo, R., Ceci, M., Zdravevski, E., & Japkowicz, N. (2020). Scalable auto-encoders for gravitational waves detection from time series data. *Expert Systems with Applications*, 113378.

Instance generation: strategy

◆ Objective

Use a trained decoder to generate plausible unseen data

◆ Method

Learn probability distributions instead of codes

◆ Reference

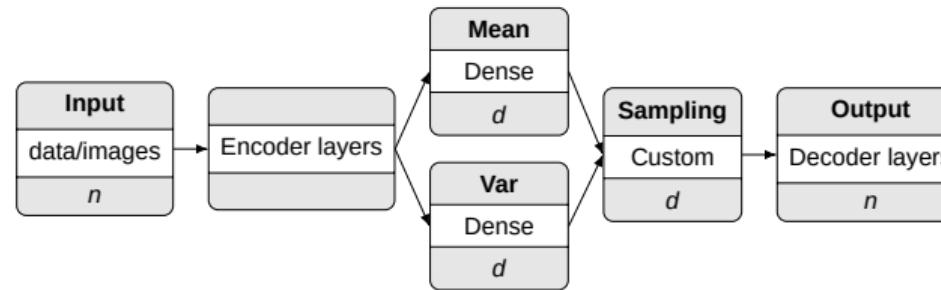
Sakurada et al. "Anomaly detection using autoencoders with nonlinear dimensionality reduction." In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (pp. 4-11).



ThisPersonDoesNotExist.com

Generative autoencoders for instance generation

Basic variational autoencoder structure:



Reparametrization trick: estimate a sample from $\mathcal{N}(\mu, \Sigma)$ using one from $\mathcal{N}(0, I)$.
With 1 variable: $z' = \mu + \sigma z$

Generative autoencoders for instance generation

From the literature: **galaxy deblending**

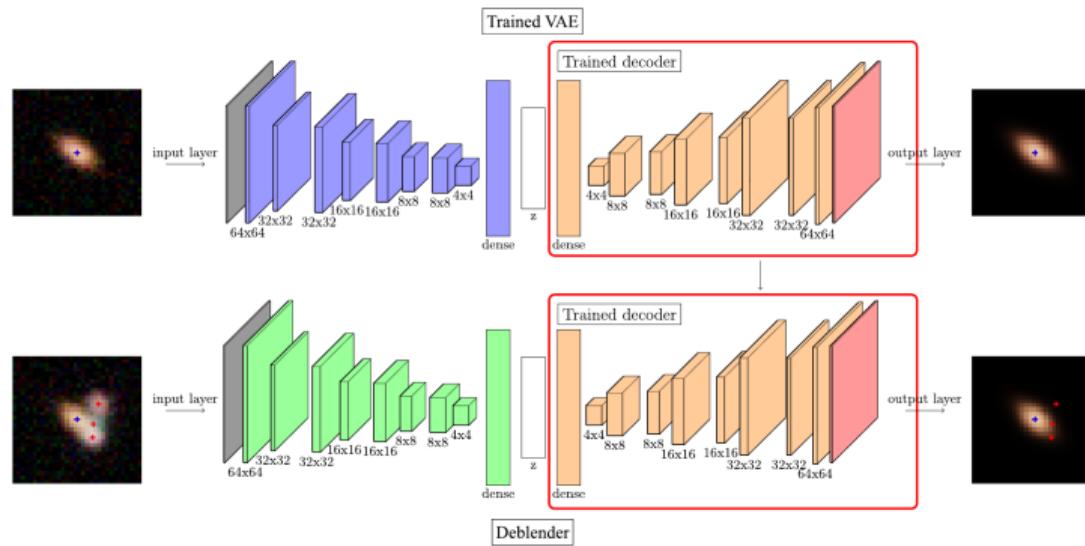


Figure 3. Architectures of the VAE and the deblender. The weights of the VAE's trained decoder are loaded and fixed in the deblender network before training.

B. Arcelin et al., "Deblending galaxies with variational autoencoders: A joint multiband, multi-instrument approach," Monthly Notices of the Royal Astronomical Society, vol. 500, no. 1, pp. 531–547, 2020

Contents

Motivation

Definition

Example

Applications

Current situation and future

Software packages

Deep learning libraries:

- ◆ Tensorflow/Keras
- ◆ PyTorch

Generic implementations of autoencoders are scarce:

- ◆ Ruta (for R) [https:// ruta.software](https://ruta.software)
- ◆ autoencoder (Python)

Many specific implementations for concrete applications

What's next?

- ◆ Many variants of generative autoencoders (VAE, Adv. AE)
- ◆ **Transformers**
A. Vaswani et al., "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008, 2017

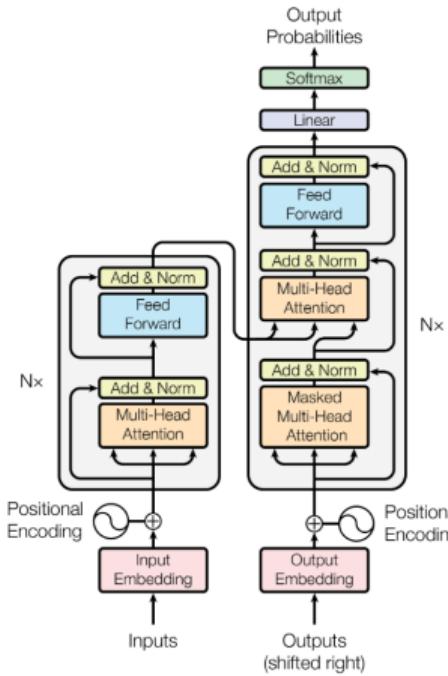
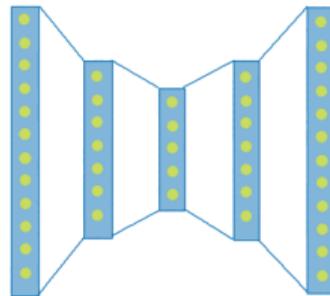


Figure 1: The Transformer - model architecture.



DaSCI

Instituto Andaluz de Investigación en
Data Science and Computational Intelligence



UNIVERSIDAD
DE GRANADA

Autoencoders: An Overview and Applications

David Charte

fdavidcl@ugr.es

SOMACHINE 2020



INSTITUTO DE
ASTROFÍSICA DE
ANDALUCÍA

EXCELENCIA
SEVERO
OCHOA

CSIC