IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# UE H : Projects on recent advances in machine learning

Santiago AGUDELO
Gonzalo QUINTANA
Gustavo RODRIGUES DOS REIS

# PLAN

1. Introduction
2. Approche paramétrique
3. GANs et concepts utilisés
4. Génération des données avec des GANS
5. Conclusions

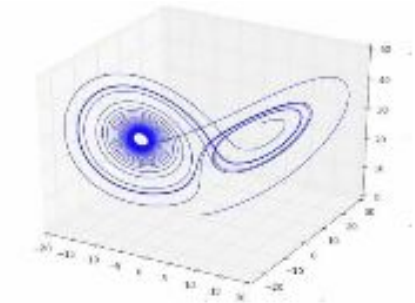
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# INTRODUCTION

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Système non linéaire $\dot{X} = f(X)$
- On suppose que cette fonction peut être exprimée comme une somme finie de fonctions non linéaires (sparsité).
- **Modèle de Lorenz :** des petites variations dans les paramètres du modèle entraînent des variations significatives dans la réponse du système.

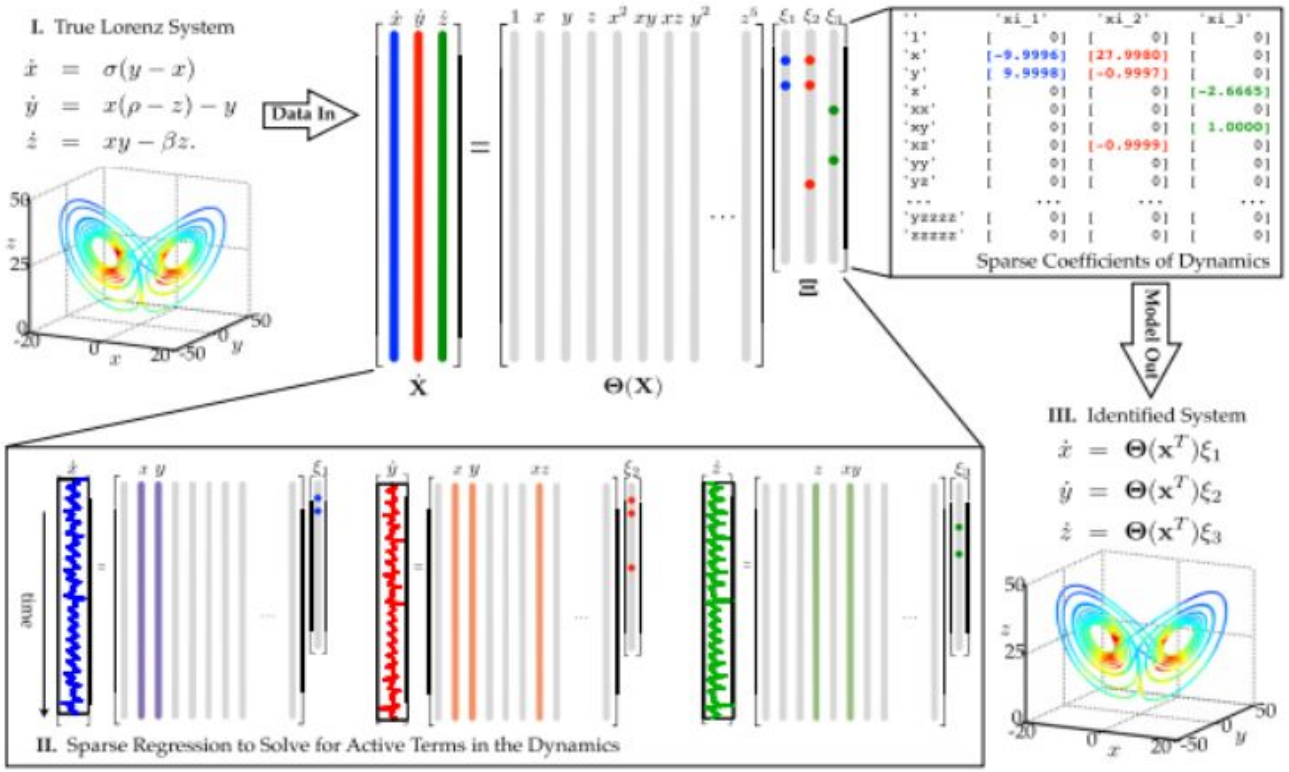
$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$
$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$



# Introduction

## Définitions






**Source** : *Discovering governing equations from data by sparse identification of nonlinear dynamical systems* (2015).**Steven Brunton, Joshua Proctor and Nathan Kutz**. Disponible dans: https://www.pnas.org/content/pnas/113/15/3932.full.pdf

# Considérations sur la discrétisation des dérivées

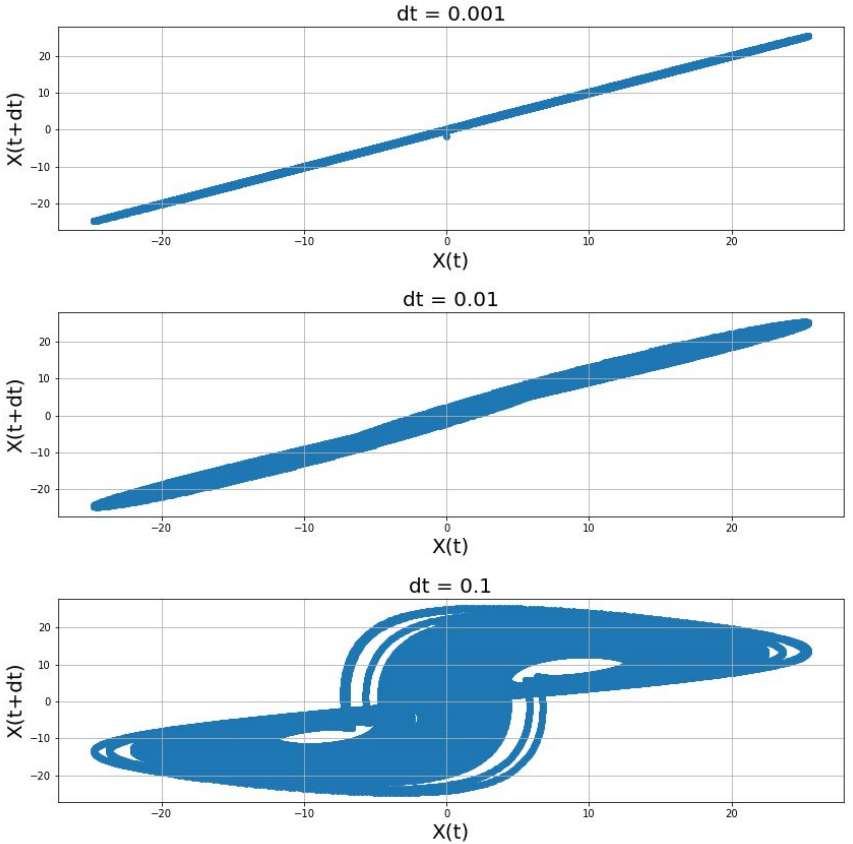


- Pour que l'approximation de premier ordre

$$X(t+\delta t) = X(t) + \delta t f'(X(t))$$

puisse être utilisée, il faut prendre δt suffisamment petit.

**Intérêt de l'approche GAN :** possibilité d'utiliser des pas temporels plus grands.



# Exposant de Lyapunov



- On s'intéresse à quantifier le comportement chaotique des systèmes.
- Soient $X_1(t), X_2(t)$ deux trajectoires calculées à partir de deux conditions initiales différentes proches, et soit $\delta(t) = X_1(t) - X_2(t)$
- On s'intéresse à l'exposant λ qui satisfait :

$$\|\delta(t)\| = \|\delta(0)\| e^{\lambda t}$$

- Pour le modèle de Lorenz, λ=0.9056 (valeur théorique).
- Une valeur positive de λ implique un comportement chaotique, car deux trajectoires partant des points rapprochés divergent exponentiellement dans le temps.



**Source** : *Lyapunov exponents in constrained and unconstrained ordinary differential equations* (2003)**.Michael Hartl**.
Disponible dans: https://arxiv.org/pdf/physics/0303077.pdf

# APPROCHE PARAMÉTRIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Régression parcimonieuse

## Méthodes LASSO



- Le but est de résoudre le problème LASSO, c'est à dire, de minimiser

$$J(\Xi) = ||\dot{X} - \Theta\Xi||_2^2 + \lambda||\Xi||_1$$

- Critères de reconstruction et de sparsité.
- Différents algorithms existent pour résoudre ce problème :

  - **IST** (iterative shrinkage and thresholding)
  - **ADMM** (alternate direction of multipliers method)
  - **LARS** (least angle regression)
  - **etc**

# Régression parcimonieuse

## D'autres approches d'estimation paramétrique



- **Régression de Ridge:** on cherche à minimiser $J(\Xi) - ||\dot{X} - \Theta\Xi||_2^2 + \lambda||\Xi||_2^2$.
- **Régression contrainte de Galerkin :** on suppose qu'on connaît en avance l'un des paramètres et on résout un problème d'optimisation avec contraintes.
- **LOESS :** régression non paramétrique qui combine plusieurs méthodes de régression au sein d'un méta-modèle reposant sur la méthode des k plus proches voisins.
- Dans la littérature, on trouve qu'une approche combinant les algorithmes de **Jaya et Powell** peut atteindre une précision dans le calcul de paramètres allant jusqu'à 10e-23.

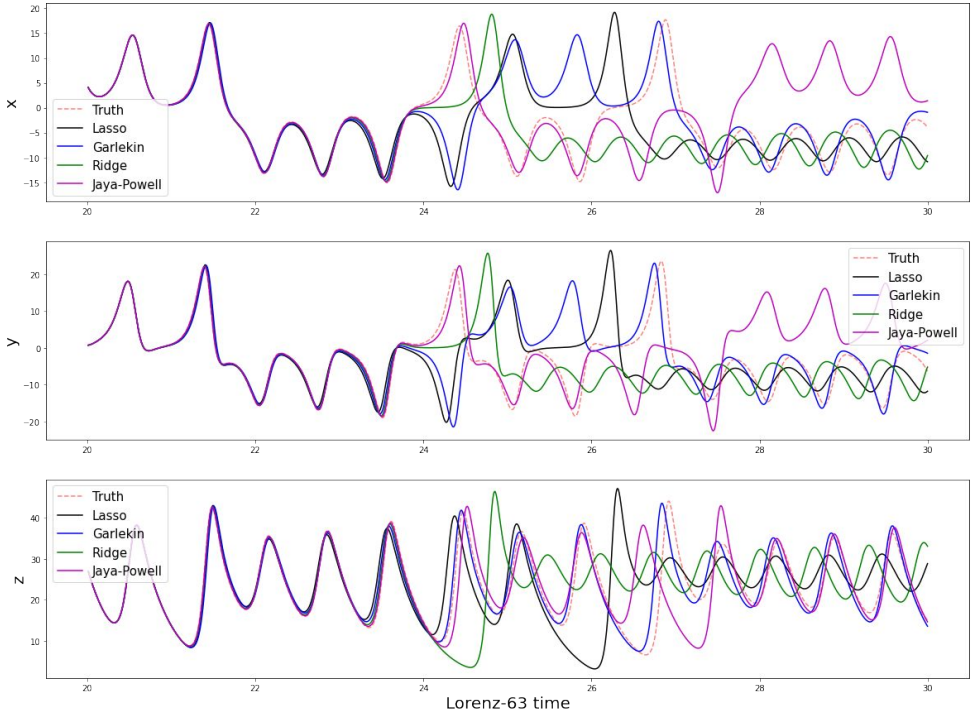


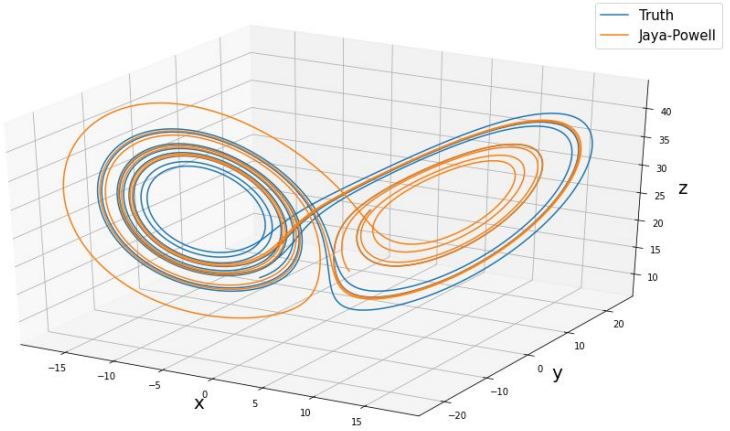
**Source** : *Parameter Estimation of Lorenz Chaotic System Based on a Hybrid Jaya-Powell Algorithm* (2019).**Li Zhuang, Longpeng Cao, Yong Wu, Yonghong Zhong, Lili Zhangzhong, Wengang Zheng and Long Wang** . Disponible dans: https://ieeexplore.ieee.org/abstract/document/8963912

# Résultats : approche paramétrique

dt = 0.001






| Méthode | RMSE paramètres | $\lambda$ estimé |
|---|---|---|
| Données | - | 0.9626 |
| LASSO | 0.3394 | 0.7354 |
| Galerkin | 0.0550 | 0.8829 |
| Ridge | 0.0833 | 0.8518 |
| Jaya-Powell | 0.0989 | 0.8376 |

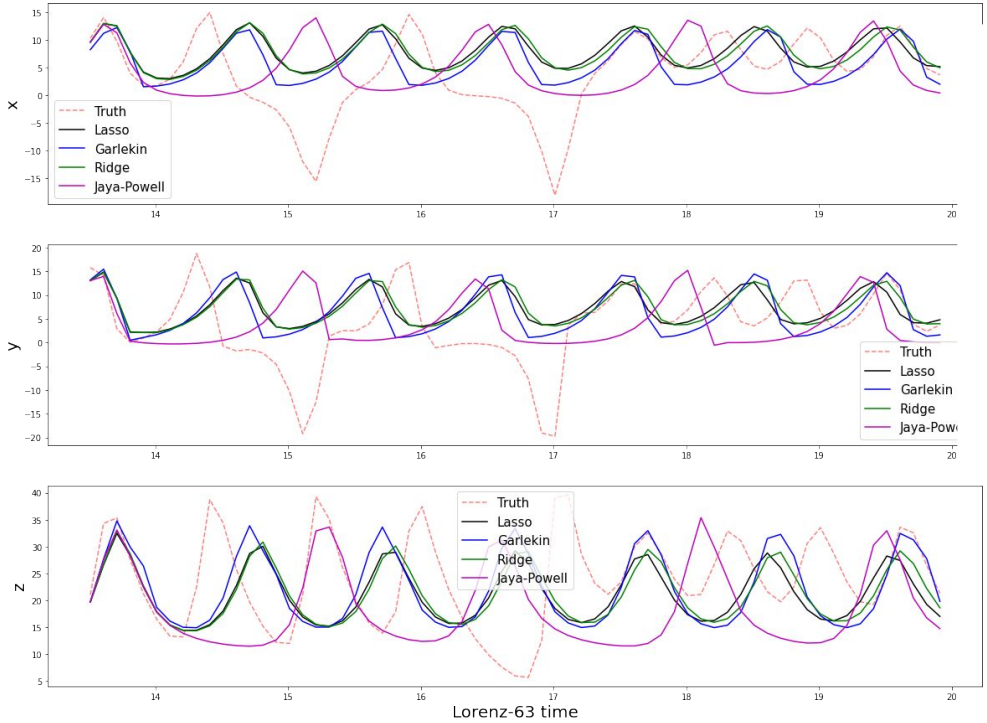
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

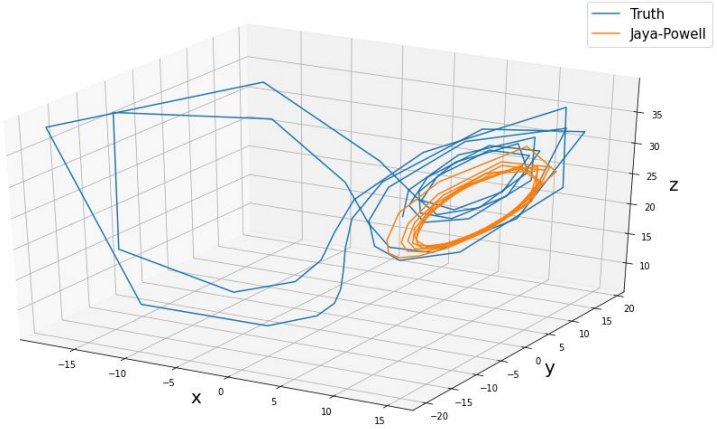
**Source** : *Lyapunov exponent of the Lorenz system* (2015).**Chebfun**.
Disponible dans:
http://www.chebfun.org/examples/ode-nonlin/LyapunovExponents.html

# Résultats : approche paramétrique

dt = 0.1





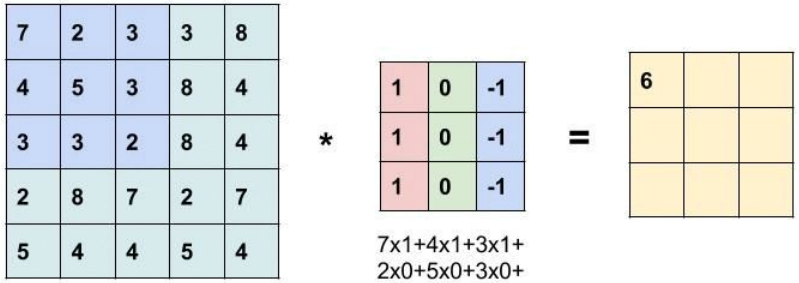

| Méthode | RMSE paramètres | $\lambda$ estimé |
|---|---|---|
| Données | - | 0.9193 |
| LASSO | 6.6577 | -0.4516 |
| Galerkin | 5.8982 | -0.4866 |
| Ridge | 5.9576 | -0.4810 |
| Jaya-Powell | 8.3441 | -0.4736 |

# GANs ET CONCEPTS UTILISÉS

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

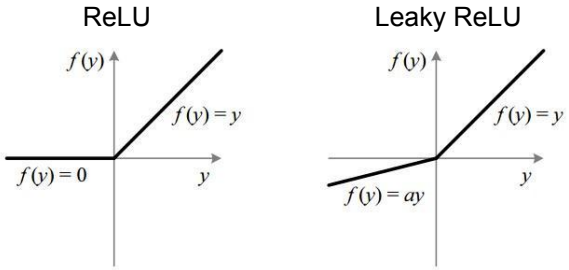
# Couche Convolutionnel (*Convolutional Layer*)






7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

Source:
https://medium.com/datadriveninvestor/convolutional-neural-networks-3b241a5da51e

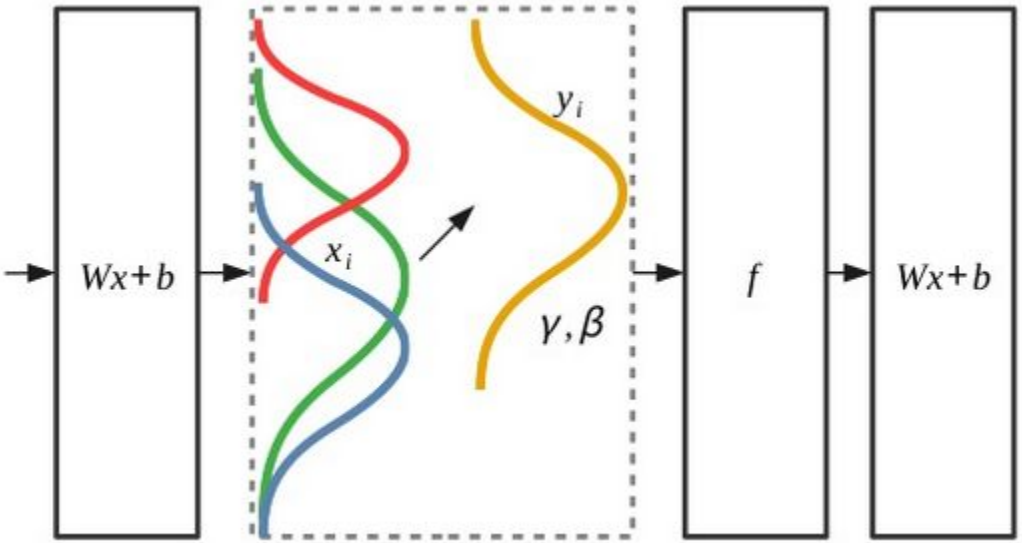



- Dans le contexte des architectures d'apprentissage profonde les couches convolutionnels sont souvent utilisées quand les données peuvent être représentés comme des matrices (e.g. des images).

- Ce type de couche peut être divisé en deux parties:
  - Une partie linéaire: L'opération de convolution entre les données d'entrée et le/les kernels de chaque couche.
  - Une partie non-linéaire: Une opération non-linéaire réalisé element par element. Des exemples de ces fonctions sont données à gauche.

# Normalisation pour batch (*Batch Normalization*)



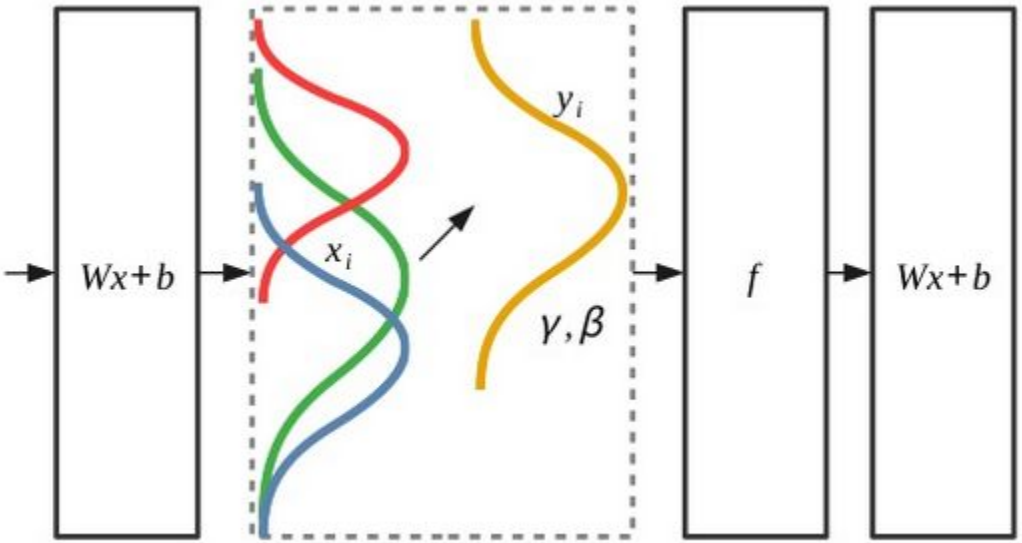



Source:
https://www.slideshare.net/ssuserd6984b/batch-normalization

- La normalisation pour batchs est une technique qui permet d'améliorer la vitesse, les performances et la stabilité pendant l'entraînement d'un réseau.
- Les poids optimaux des neurones du réseau sont calculés après que plusieurs "exemples aient été montrés au réseau" (un batch) et en normalisant les valeurs qui sont entrées dans les neurones.
- Cette normalisation se fait une fois par batch.

# Normalisation pour batch (*Batch Normalization*)



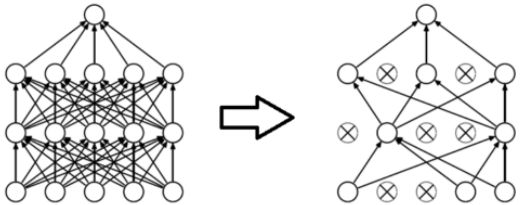



Source:
https://www.slideshare.net/ssuserd6984b/batch-normalization

- Après chaque couche, la distribution des données peut changer, et ce changement rend plus difficile pour le réseau d'extraire les caractéristiques utiles.
- Lorsque nous nous entraînons par batch, chaque batch a une distribution différente.
- En d'autres termes, la normalisation pour batch fixe les premier et deuxième moments des données (moyenne et variance) à zéro et un respectivement.

# Dropout





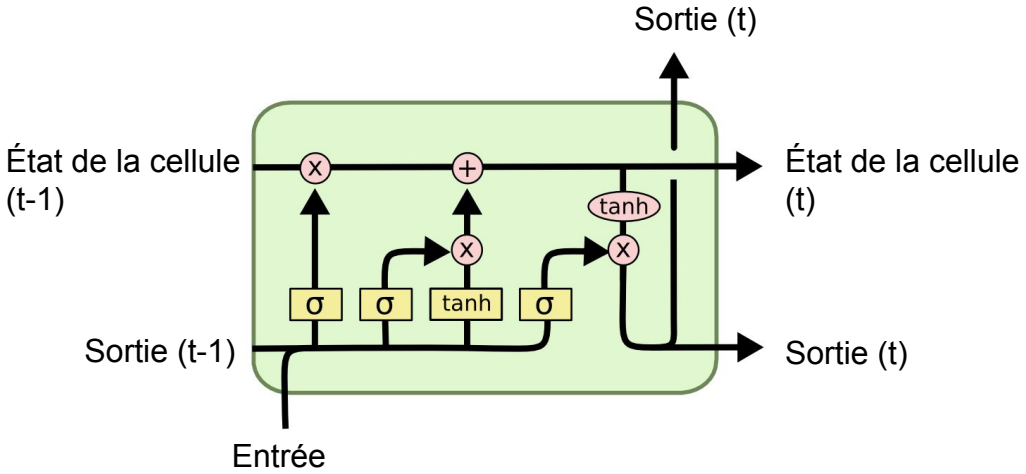
Source:
https://data-science-blog.com/blog/2019/05/20/understanding-dropout-and-implementing-it-on-mnist-dataset/

- Parfois, une partie du réseau a des poids très importants et finit par dominer l'entraînement.
- Alors que l'autre partie du réseau reste faible et ne joue pas vraiment un rôle dans la formation.
- Pour résoudre ce problème, le dropout arrête de façon aléatoire certains nœuds et empêche les gradient de leur traverser.

# LSTM



Une couche **L**ong **S**hort **T**erm **M**emory fait partie des architectures dans le contexte des réseaux des neurones permettant d'utiliser les séries temporelles comme donnés d'entrée car elle peut conserver des informations sur le passé de la série temporelle en traitant le problème de disparition du gradient.




$\sigma$ - fonction sigmoïde.

tanh - fonction tangente hyperbolique.

x - opération de multiplication point par point.

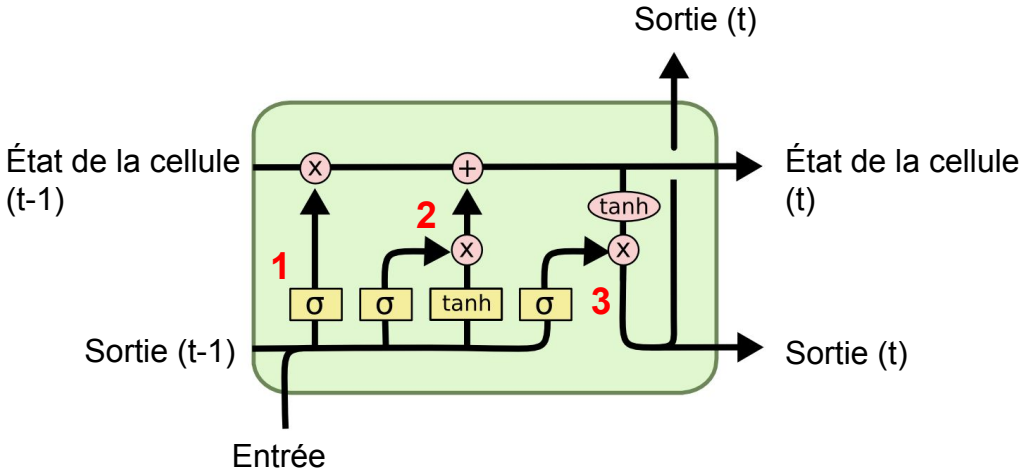
\+ - opération d'addition point par point

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# LSTM



Une couche **L**ong **S**hort **T**erm **M**emory fait partie des architectures dans le contexte des réseaux des neurones permettant d'utiliser les séries temporelles comme donnés d'entrée car elle peut conserver des informations sur le passé de la série temporelle en traitant le problème de disparition du gradient.




**1 -** Forget gate

**2 -** Input gate

**3 -** Output gate

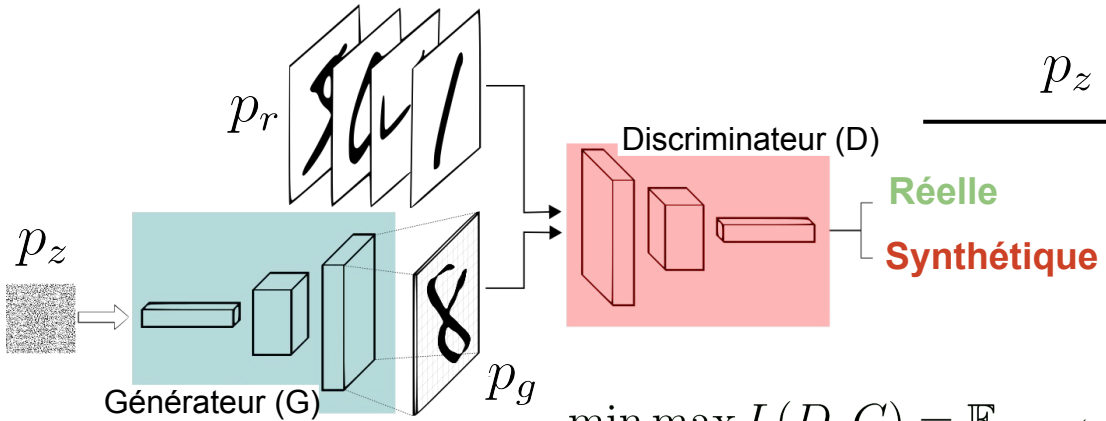
Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Generative Adversarial Network (GAN)



- Les GAN consistent en réalité d'une composition de deux modèles: Générateur (G) et Discriminateur (D)

| Symbole | Signification | Remarques |
|---|---|---|
| $p_r$ | Distribution des données réelles | |
| $p_g$ | Distribution des données générées par G | |
| $p_z$ | Distribution des données sur le bruit d'entrée z | Généralement gaussian ou uniforme |

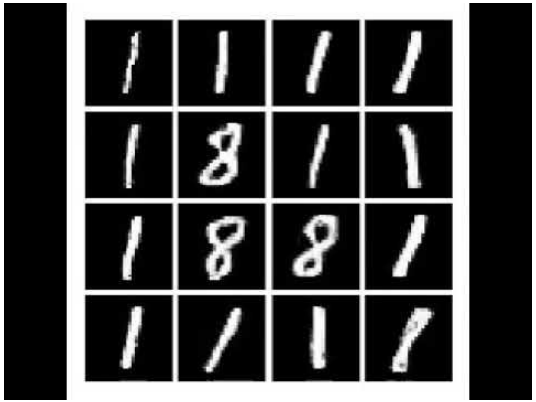



$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

# Problèmes avec des GANs



- Le GAN est vulnérable à ce qu'on appelle *mode collapse* (faible gamme de données créées) et à l'instabilité pendant l'entraînement.



Mode Collapse - Source: https://www.youtube.com/watch?v=ktxhiKhWoEE

Pour surmonter ces deux problèmes on peut se poser, dans la phase de conception d'une architecture GAN les question suivantes [5]:

1) Le modèle a-t-il une capacité (complexité, c'est-à-dire nombre des paramètres) suffisante pour saisir la distribution complexe des données d'entrée ?

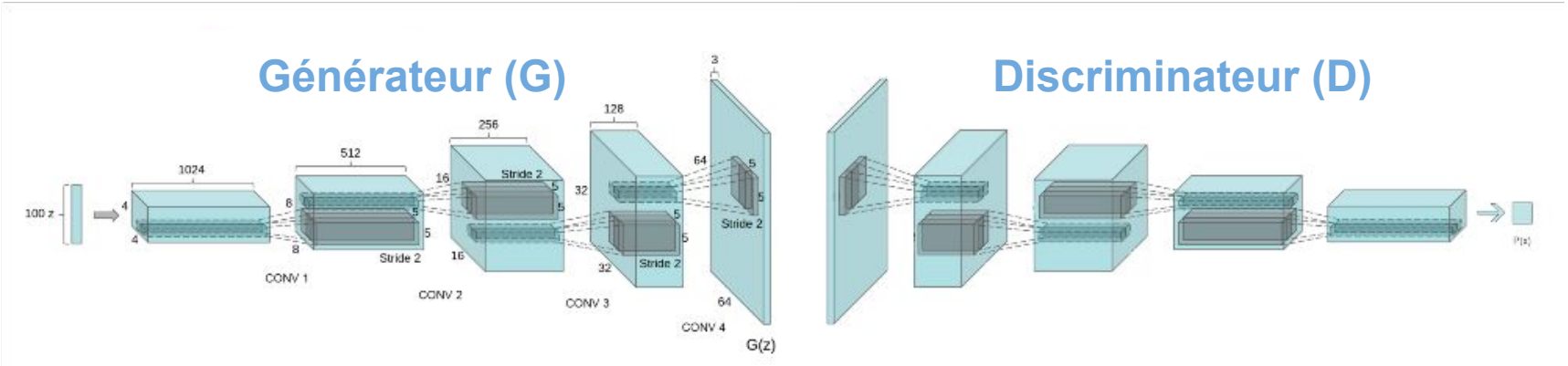
2) Si la capacité est suffisante, la minimisation de la fonction de perte donnée garantira-t-elle une optimalité des données avec des limites de généralisation considérables ?

3) En supposant qu'il existe un optimum global, avons-nous un algorithme d'optimisation qui l'atteint dans des itérations finies ?

# DCGAN



- Le GANs avec des couches convolutionnels (DCGAN) sont une famille de ConvNets qui imposent certaines contraintes architecturales pour stabiliser la formation des GAN.
- Dans les DCGAN, le générateur est composé d'une série d'opérations de convolution transposées.
- Ces opérations prennent un vecteur de bruit aléatoire z et le transforment en augmentant progressivement ses dimensions spatiales tout en diminuant la profondeur du volume de ses caractéristiques.

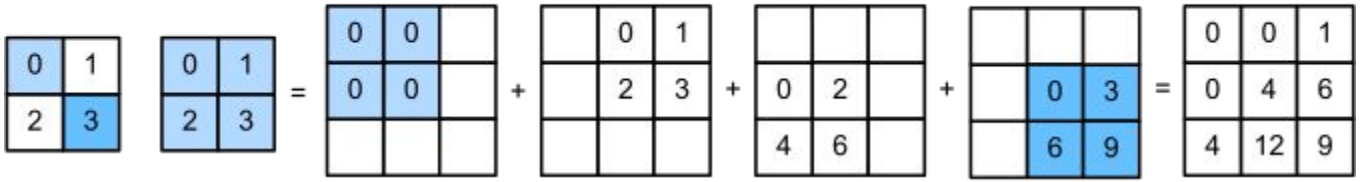



DCGAN - Source: https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html

# DCGAN - Convolutions transposées



- Le GANs avec des couches convolutionnels (DCGAN) sont une famille de ConvNets qui imposent certaines contraintes architecturales pour stabiliser la formation des GAN.
- Dans les DCGAN, le générateur est composé d'une série d'opérations de convolution transposées.
- Ces opérations prennent un vecteur de bruit aléatoire z et le transforment en augmentant progressivement ses dimensions spatiales tout en diminuant la profondeur du volume de ses caractéristiques.




Convolution transposée - Source:
http://d2l.ai/chapter_computer-vision/transposed-conv.html

# Normalisation spectrale



- La normalisation spectrale [2] s'appuie sur le concepte de continuité de Lipschitz.
- Dans le cas d'un discriminateur d'une GAN on peut le voir comme une application: $D : I \to \mathbb{R}$, où s'agit $I$ de l'espace des images, par exemple: $\mathbb{R}^{32\times32}$

Si notre discriminateur est K-Lipschitz continu, alors
pour tous les x et y en I, nous avons (|.| la norme L2) :

$$||D(x) - D(y)|| \leq K||x - y||$$

- Demander une continuité de Lipschitz peut limiter les gradients dans notre discriminateur.
- Faire une normalisation spectrale dans notre réseau ça veut dire simplement pour chaque poids *W* computer sa mise à jour comme: $W/\sigma(W)$

$$\sigma(W) = u_{t+1}^T W v_{t+1}$$

$$u_{t+1} = W v_t$$

$$v_{t+1} = W^T u_{t+1}.$$

En voyant W comme une application linéaire $W : \mathbb{R}^n \to \mathbb{R}^m$ et où $\sigma(W)$ est la racine carrée de la plus grande valeur de l'application.

Dans la pratique on initialise $v \in \mathbb{R}^n$ et $u \in \mathbb{R}^m$ aléatoirement et on réalise les mises à jour à chaque itération en suivant les équations à gauche.

# GENERATION DE DONNEES AVEC DES GANs

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# Génération de données avec des GANs



## Modélisation des données

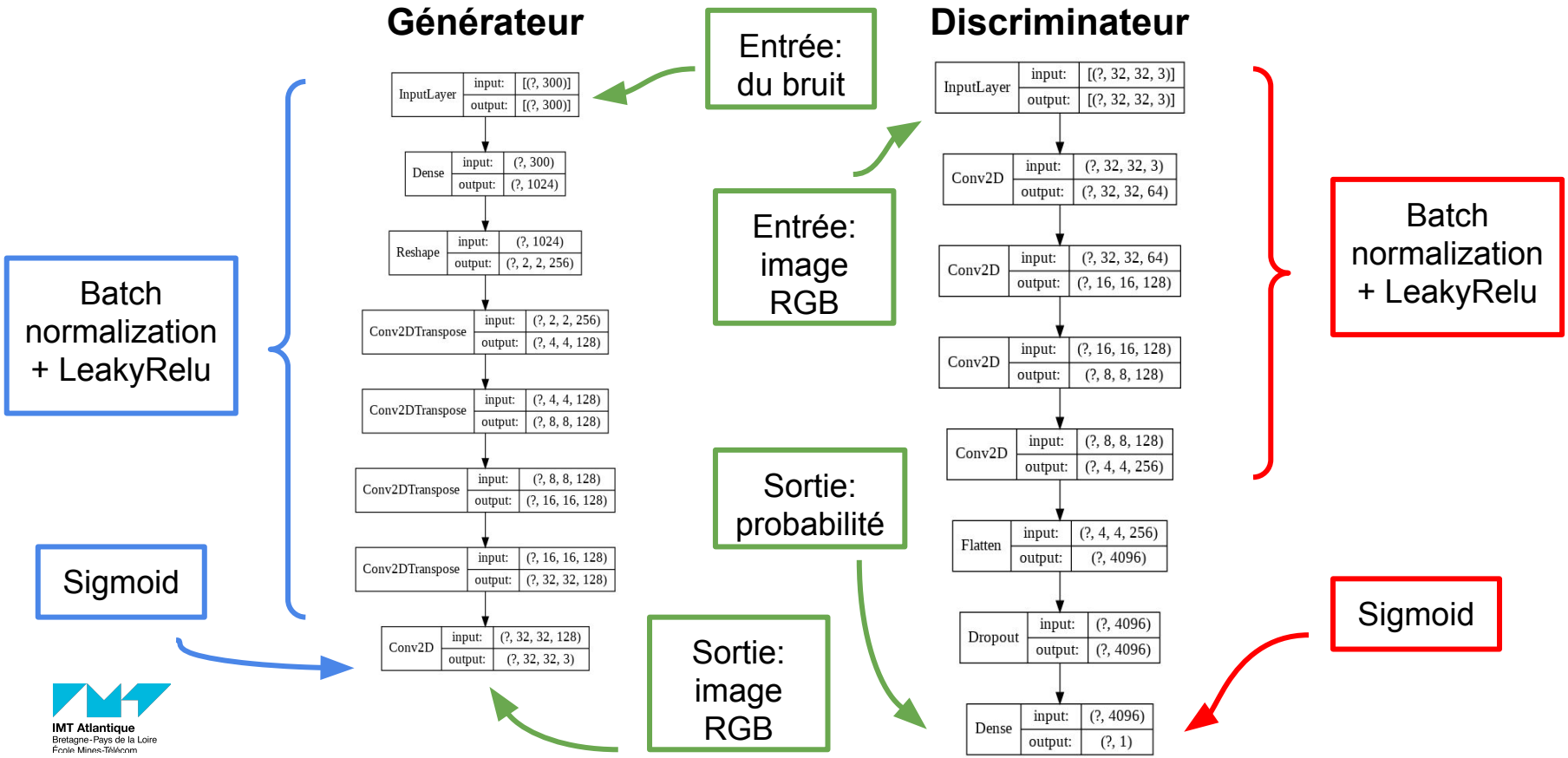
- Dataset composé de 1000 exemples de longeur 1000.
- On utilise dt=0.1 pour le pas d'intégration
- T=100 temps de Lorenz
- On transforme chaque série temporelle dans une image RGB, où chaque canal représente chacune des trois coordonnées.

Dataset composé de 1000 images RGB de taille 32x32

# Génération de données avec des GANs



Architecture proposée



# Génération de données avec des GANs



Techniques utilisées pour arriver à la convergence

- **Label smoothing:** on remplace les labels “1” et “0” (données réelles et fausses, respectivement) pour des valeurs aléatoires dans la gamme 0.9-1 et 0-0.1 pendant l’entraînement du régresseur, pour lui empêcher d’être trop confident. Cela lui empêche d’utiliser peu de caractéristiques pour faire la classification.

- **Ajout du bruit:** pour difficulter l’entraînement du régresseur, on ajoute du bruit gaussien aux données réelles et générées. La variance du bruit diminue avec les epochs.

- **Two-Times-Scale-Update-Rule:** on utilise deux learning rates différentes pour le discriminateur et le générateur. Typiquement, celui du discriminateur est 4 fois plus grand que celui du générateur. Voir [1]

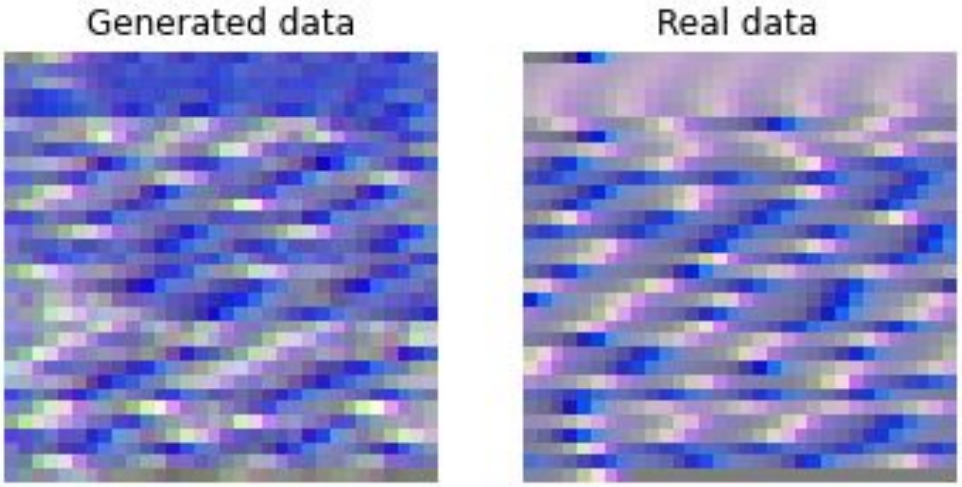
- **Inversion aléatoire des labels:** toujours dans le but d’entraver le travail du discriminateur, on inverse aléatoirement quelques labels de fausses à vraies (et vice-versa).

- **On actualise les poids du discriminateur avec moins fréquence que celles du générateur**.

# Génération de données avec des GANs



## Résultats - images générées

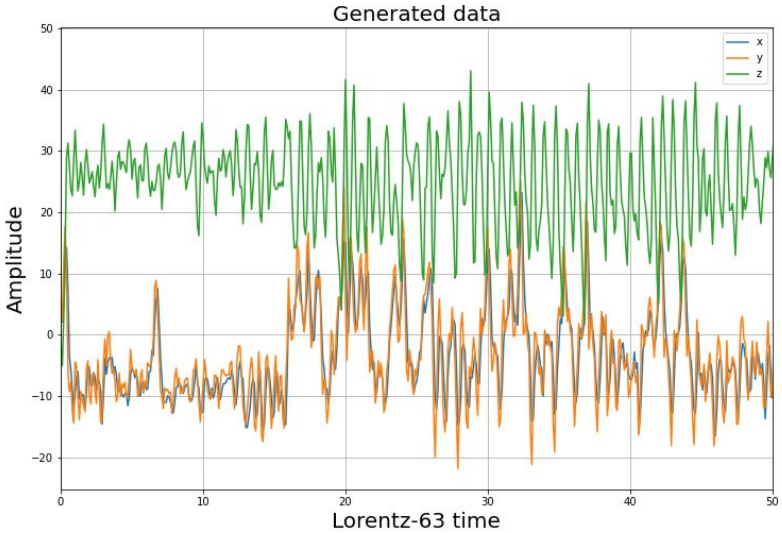


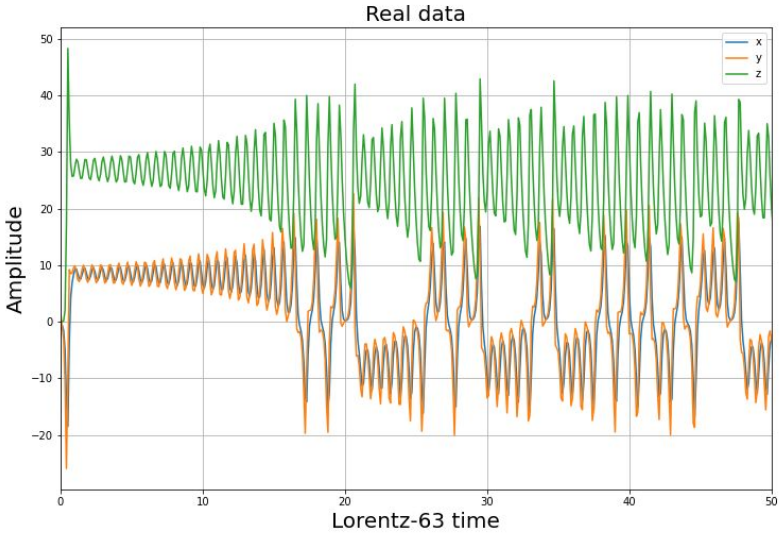

À la fin de l'entraînement,le réseau est capable de générer des images qui ressemblent à celles d'entrée.

# Génération de données avec des GANs



## Résultats - graphiques temporelles



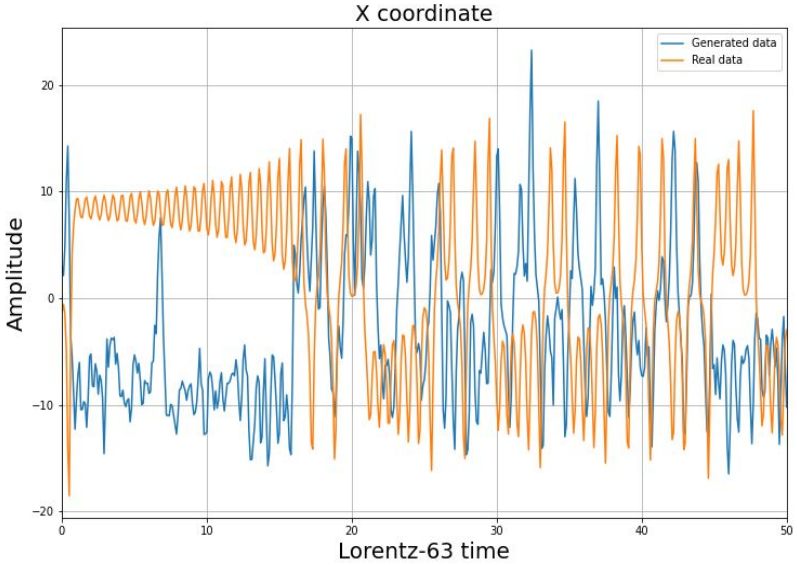



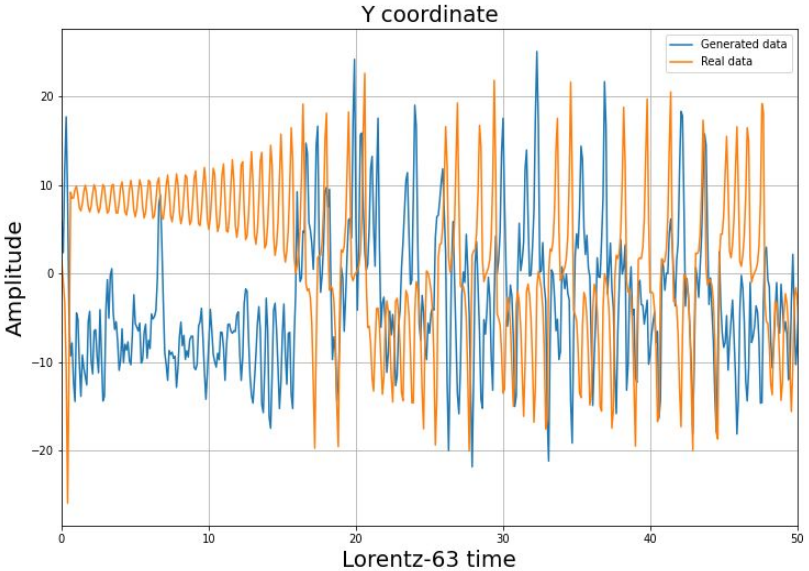

- Les variables X et Y se ressemblent entre elles
- La variable Z a une valeur moyenne plus élevée
- Oscillations avec des grosses sautes

# Génération de données avec des GANs



## Résultats - graphiques temporelles

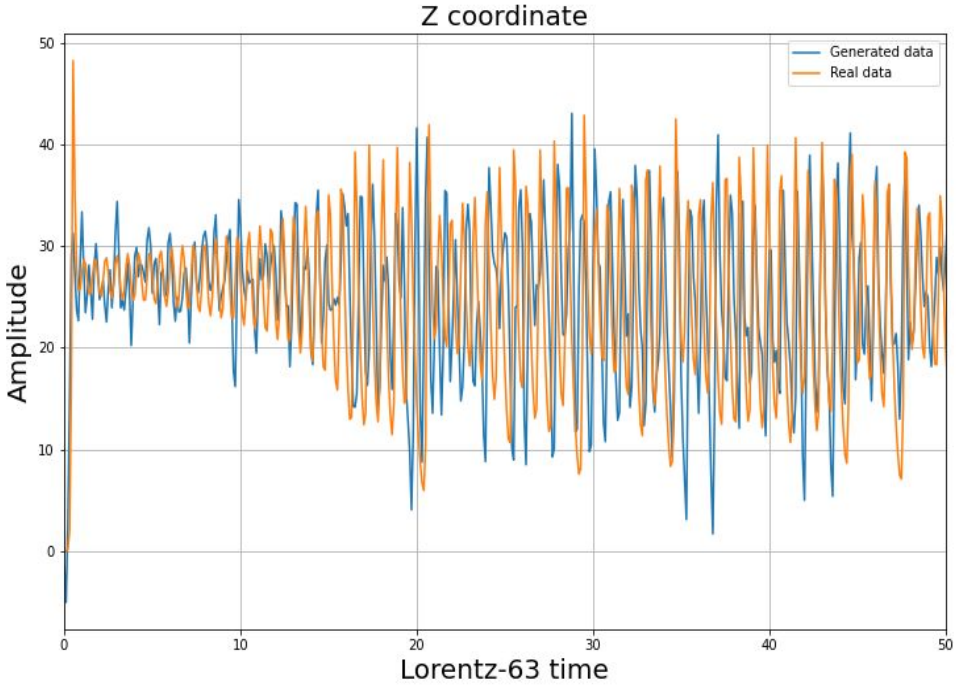


# Génération de données avec des GANs



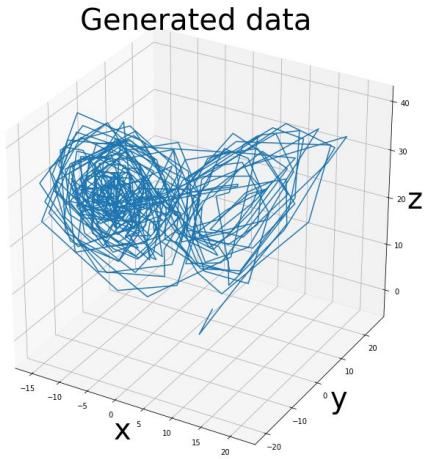
## Résultats - graphiques temporelles

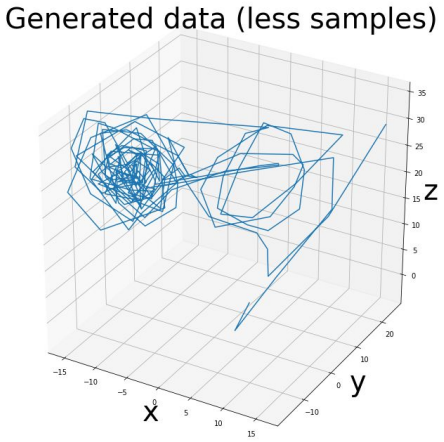


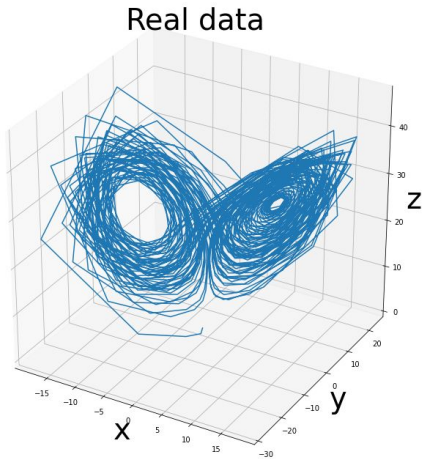
# Génération de données avec des GANs



## Résultats - graphique 3D




On voit que la série générée a, approximativement, la forme du papillon de la série d'entraînement.

# Génération de données avec des GANs



## Évaluation quantitative des résultats

On mesure le degré de chaos dans les séries générées en calculant les exposant de Lyapunov avec l'algorithme d'Eckmann pour chacune des coordonnées

| | Données générées | Données réelles |
|---|---|---|
| X | 1.35 | 2.86 |
| Y | 1.25 | 3.05 |
| Z | 0.86 | 3.64 |

Le coefficient de Lyapunov est positif dans tous les cas, ce qui indique que la série est chaotique!

# Conclusions



- Pour des dt suffisamment petits, les différentes variantes de régression linéaire produisent des bon résultats.
- Pourtant, pour dt=0.1 les régresseurs linéaires ont mal à estimer correctement les paramètres du modèle.
- La génération des données avec des GANs est une alternative intéressante. Résultats visuellement acceptables et qui conservent le chaos.

**TRY ME ON GITHUB!**

# Bibliographie



**[1] GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.** Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter

**[2] Spectral Normalization for Generative Adversarial Networks.** Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida. Disponible dans: https://arxiv.org/pdf/1802.05957.pdf

**[3] Spectral Normalization Explained.** Ode to gradients: Christian Cosgrove. Disponible dans: https://christiancosgrove.com/blog/2018/01/04/spectral-normalization-explained.html

**[4] Understanding LSTM Networks.** Christopher Olah. Disponible dans: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

**[5] Generative Adversarial Networks (GANs): The Progress So Far Image Generation.** Padala Manisha, Sujit Gujar. Disponible dans: https://arxiv.org/pdf/1804.00140.pdf