Generación de series temporales con GANs

Generación de series temporales

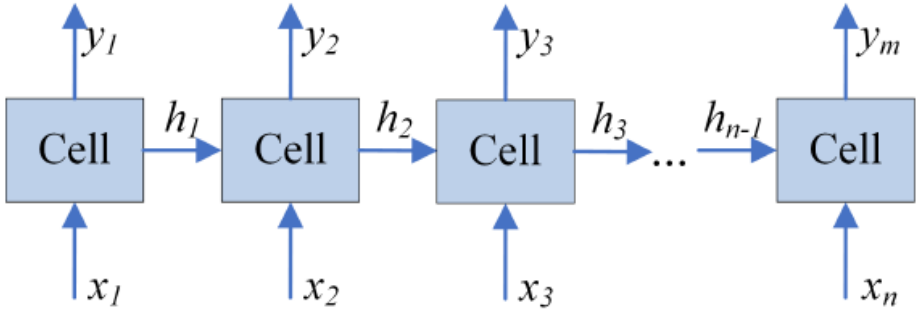
- Capacidad de GANs para aprender y generar nuevos datos sintéticos que preservan la fidelidad y las características de un conjunto de datos reales
- Sin embargo, generar imágenes o datos tabulares es bastante más simple que generar conjuntos de datos que deben preservar una dinámica temporal.
- Modelar apropiadamente los datos de series temporales significa que un modelo no solo debe capturar las distribuciones de las características de los conjuntos de datos en cada punto de tiempo, sino que también debe poder capturar la dinámica compleja de esas características a lo largo del tiempo.

Redes neuronales recurrentes

- ANN diseñadas para procesar datos secuenciales, como las series temporales.
- A diferencia de las ANN feedforward, que solo consideran la entrada actual para calcular la salida, las RNN consideran la entrada actual en conjunto con entradas recibidas previamente, mediante el uso de conexiones recurrentes que cubren pasos de tiempo adyacentes.
- Una RNN toma una secuencia de entradas $(x_1, x_2, \dots, x_n)$ para producir una secuencia de salidas $(y_1, y_2, \dots, y_m)$, con $m \leq n$.
- Se utilizan estados ocultos h que sirven como un mecanismo para recordar información a través del tiempo. La salida en el paso de tiempo t se obtiene como $y_t = f(x_t, h_{t-1})$ donde x_t es la entrada actual, h_{t-1} es el estado oculto anterior y f es una función no lineal.

Redes neuronales recurrentes

- Celdas: RNN estándar (vanilla), memoria larga de corto plazo (LSTM) u otras.
- En una RNN estándar, el algoritmo de backpropagation puede causar desaparición del gradiente para secuencias largas.
- La celda LSTM incluye un estado interno adicional (estado de celda) y puede almacenar información durante períodos de tiempo más largos.
- RNN multicapa: apila capas de celdas de memoria, que agregan niveles de abstracción y potencialmente permiten que las celdas de cada capa operen en diferentes escalas de tiempo.



TimeGAN (Yoon et al. 2019)

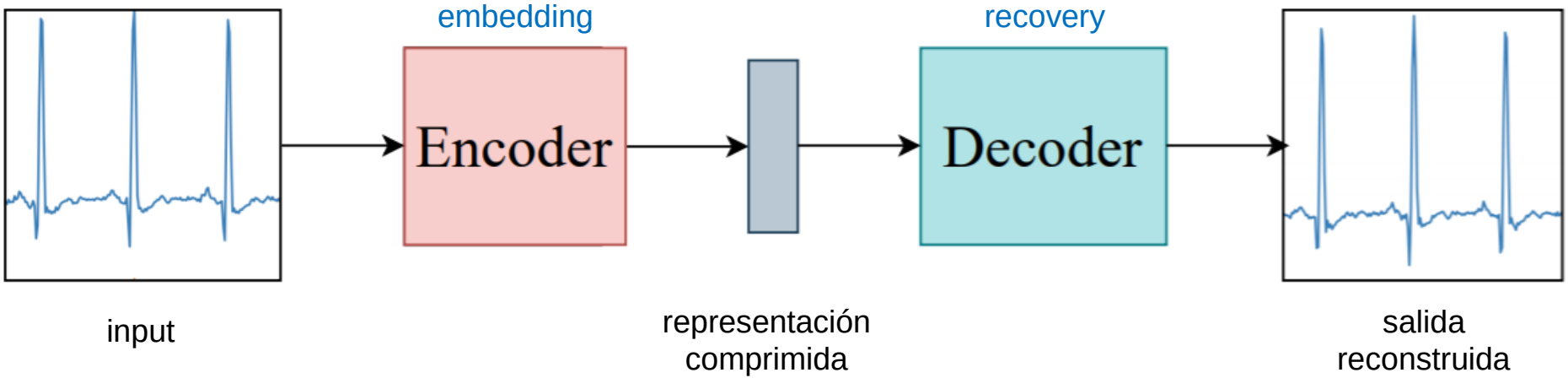
- Idea global: preservar la dinámica temporal de las series temporales.
- Métodos tradicionales de las GANs aplicados a series tienen dificultades para capturar las relaciones temporales.
- Modelos supervisados para la predicción de series temporales permiten un control más preciso de la dinámica de la ANN, pero son inherentemente deterministas.
- Propuesta de TimeGAN: combinar ambos enfoques

TimeGAN (Yoon et al. 2019)

- Propuesta: combinar ambos enfoques, aportando la flexibilidad del paradigma no supervisado con el control proporcionado por el entrenamiento supervisado.
- Se propone trabajar con un embedding space optimizado simultáneamente con objetivos supervisados y adversarios, para permitir a la ANN capturar la dinámica de los datos de entrenamiento considerados en el muestreo.

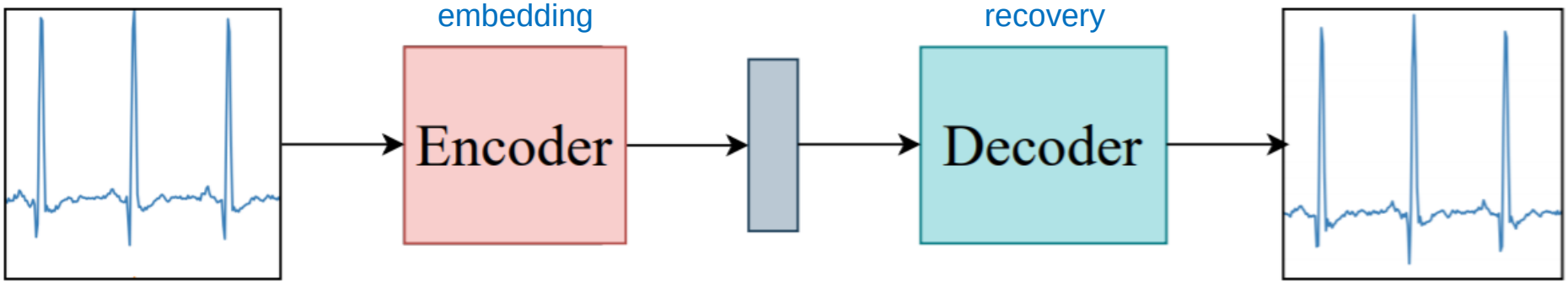
TimeGAN: propuesta algorítmica

- Idea fundamental: incorporar una ANN (embedding net) para implementar un mapeo reversible entre características del problema y representaciones del espacio latente.
- La embedding net permite reducir la dimensión del espacio de aprendizaje adversativo.



TimeGAN: propuesta algorítmica

- Una función de recuperación (recovery) revierte el mapeo, volviendo al espacio de características.
- Se busca aprovechar que la dinámica temporal de los sistemas [incluyendo sistemas complejos] en general es impulsada por pocos factores de variación de menor dimensión.



TimeGAN: conceptos clave

- TimeGAN aprende a codificar características, generar representaciones e iterar a lo largo del tiempo.
- La embedding ANN provee el espacio latente, la red adversativa opera dentro del espacio latente, y las dinámicas inherentes de los datos (tanto reales como sintéticos) se sincronizan a través de una pérdida supervisada.

TimeGAN: conceptos clave

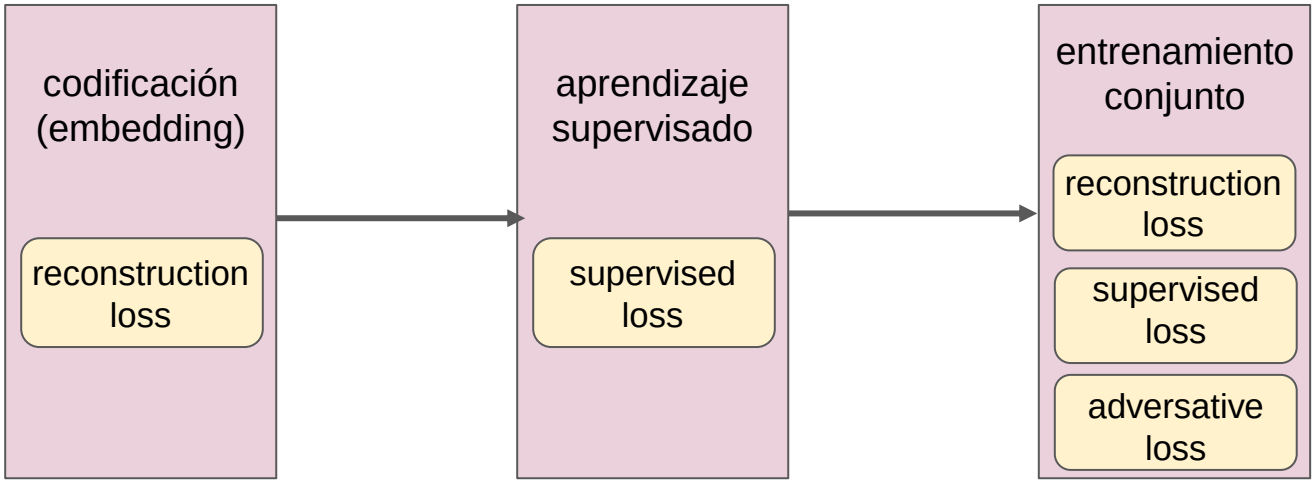
- La función de pérdida del aprendizaje supervisado se minimiza mediante el entrenamiento conjunto de la embedding net y del generador, de modo que el espacio latente está específicamente transformado para facilitar al generador el aprendizaje de las relaciones temporales.
- Discriminador y generador se entrenan (y por tanto tienen entradas y salidas) usando representaciones del espacio latente.
- Las funciones de mapeo (embedding y recovery) se implementan con ANN recurrentes (embedding) y feed-forward ANN (recovery).
- El modelo se ha generalizado para trabajar con datos mixtos, para generar datos estáticos y de series temporales simultáneamente.

TimeGAN: conceptos clave

- Diferencia con GAN: el generador considera una función de pérdida supervisada
- El generador recibe datos en el espacio latente y genera datos (en el siguiente paso) en el espacio latente.
  - Asegura que el generador aprenda las dependencias paso a paso y pueda generar datos sintéticos con transiciones similares (paso a paso).
  - El modelo se orienta a capturar la distribución (condicional) de los datos en el tiempo, utilizando los datos originales como supervisión.
- El generador es muy débil al comienzo del aprendizaje, por este motivo se lo apoya con el supervisor.
- Se diferencia de las GANs tradicionales, que trabajan con una función de pérdida adversaria no supervisada en datos reales y sintéticos.

TimeGAN: entrenamiento

- TimeGAN tiene tres fases de entrenamiento: auto-encoder (embedding), entrenamiento supervisado del generador usando datos reales y entrenamiento conjunto de todos los componentes.



TimeGAN: funciones de pérdida

- TimeGAN se basa en tres funciones de pérdida diferentes
  - Pérdida de reconstrucción: pérdida del auto-encoder (embedding y recovery), que asegura que los datos se codifiquen y decodifiquen de manera eficiente desde y hacia el espacio latente. Evalúa qué tan bien se realiza el proceso de reconstrucción de los datos codificados comparados con los datos originales.
  - Pérdida supervisada para entrenar el generador en modo de bucle cerrado y aprender la dinámica (temporal) de los datos y generar secuencias realistas en el siguiente paso de tiempo. La pérdida supervisada también se aplica al embedding para preservar la dinámica temporal de las características.
  - Pérdida adversativa no supervisada para el generador y el discriminador, que fuerza al generador a crear secuencias de datos realistas.

TimeGAN: funciones de pérdida

- Datos secuenciales
  - Muestra de los datos $\{x_0, ..., x_T\}$, de largo T y dimensión del espacio de características d (cada x_i es un vector $x_{i1}, x_{i2}, \ldots, x_{id}$).
  - x_i depende de $x_{i-1}, x_{i-2}, \ldots, x_0$ para $i = 0 \ldots T$.
- Función embedding (E) transforma la secuencia x (dimensión $T \times d$), en una representación latente e (dimensión $T \times l$)
- Función recovery (R) mapea representaciones e al espacio de características a una secuencia $\hat{x}$
- Función de pérdida para embedding y recovery (pérdida de reconstrucción)

$$L_{ER} = \frac{1}{n} \sum_{t=1}^{n} (x_t - \hat{x}_t)^2$$

TimeGAN: funciones de pérdida

- Generación de secuencias $\hat{e}$ en el espacio latente e
  - G toma ruido aleatorio z (dimensión $\mathbf{T}\times\mathbf{d}$), con distribución uniforme en [-1,1].
- G es débil al comienzo del entrenamiento, se le apoya con el supervisor (S).
- S mapea e y $\hat{e}$ a h y $\hat{h}$, también con dimensión ($\mathbf{T}\times\mathbf{l}$), preservando transiciones escalonadas similares a e.

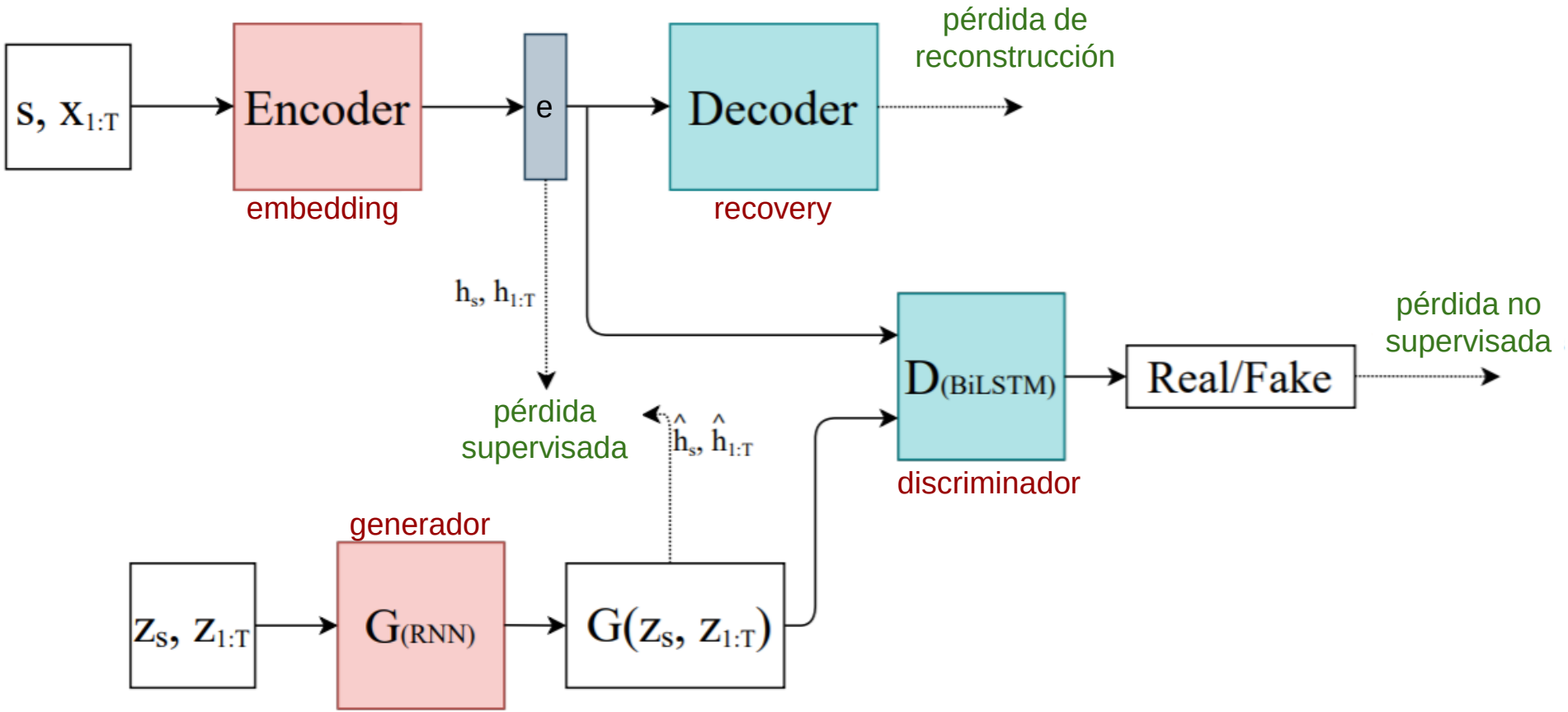
- La función de pérdida supervisada es $$L_S = \frac{1}{n-1}\sum_{t=2}^{n}(e_t - h_t)^2$$

TimeGAN: funciones de pérdida

- Luego de entrenar E, R y S, se entrenan G y D.
- G genera una secuencia $\hat{e}$, que S mapea a una secuencia $\hat{h}$.
- D recibe una secuencias real e, secuencias sintéticas $\hat{h}$ y $\hat{e}$ (generada sin ayuda de S) y mapea e a y (dimensión $T \times l$). También mapea $\hat{e}$ y $\hat{h}$ a $\hat{y}$ (dimensión $T \times l$), determinando la probabilidad de que cada elemento de la secuencia original sea real o sintético.
- La función de pérdida no supervisada es la habitual en las GAN

$$\frac{1}{n}\sum_{t=1}^{n}\log y_t + \frac{1}{n}\sum_{t=1}^{n}\log(1-\hat{y}_t)$$

TimeGAN: entrenamiento

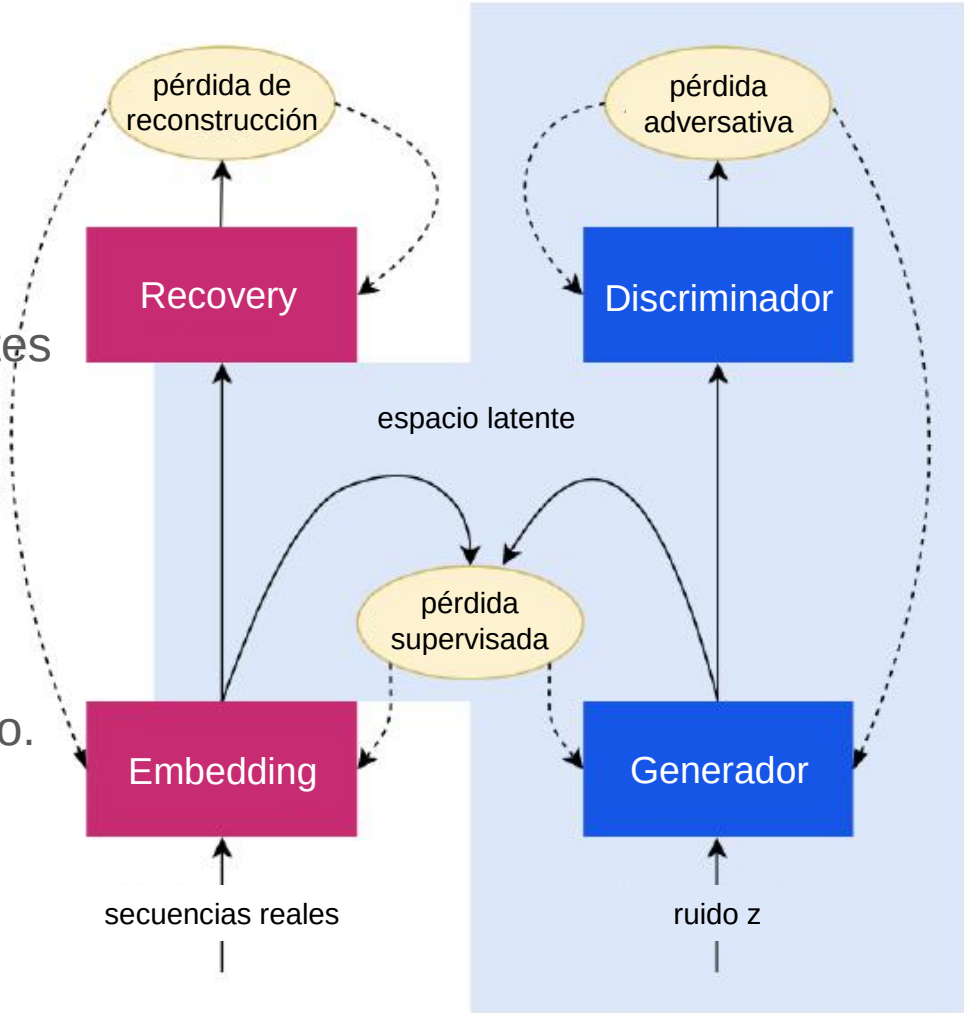


TimeGAN: entrenamiento

- Entrenamiento de autoencoder/embedding: se realiza sobre los datos reales, para asegurar una reconstrucción óptima de los datos en el proceso de codificación/decodificación.
- Entrenamiento del supervisor: se realiza sobre los datos reales, para capturar el comportamiento de la información histórica (serie temporal).
- En el entrenamiento conjunto, TimeGAN aprende a codificar, iterar y generar la serie temporal.
    - El entrenamiento es adversativo y es la fase que consume más tiempo.
- En la implementación original de TimeGAN, todas las fases se entrenan sobre la misma cantidad de iteraciones (se usa una ponderación uniforme).

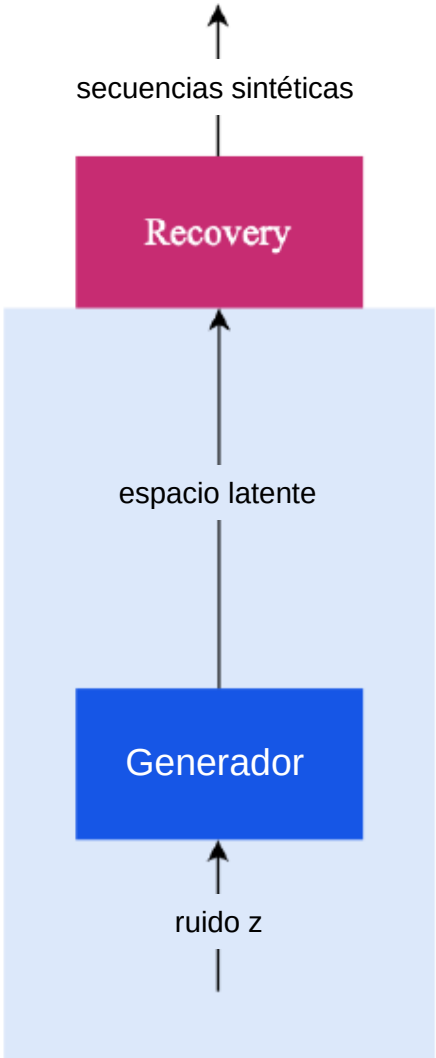
TimeGAN: entrenamiento

- TimeGAN tiene una alta demanda computacional al agregar componentes a la arquitectura y entrenar/optimizar tres funciones de pérdida.
- Flechas completas: flujo de datos.
- Flechas punteadas: funciones de pérdida utilizadas en el entrenamiento.



TimeGAN: generación de datos sintéticos

- Generación de datos sintéticos luego de la etapa de entrenamiento.
- El generador produce secuencias de ruido aleatorio en el espacio vectorial latente, que luego se traducen al espacio de características por medio de la función de recovery.

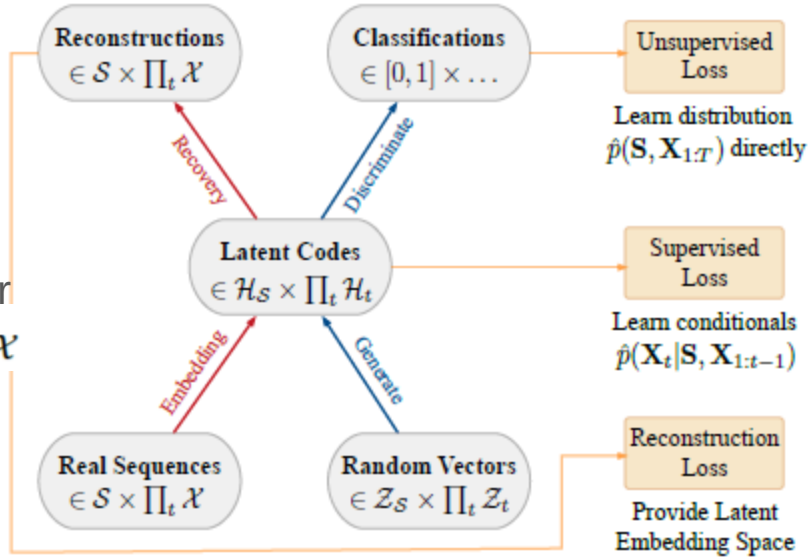


TimeGAN: implementación de Yoon et al. (2019)

- Se propuso TimeGAN para datos mixtos, para generar datos estáticos y de series temporales simultáneamente.
- Embedding y recovery deben ser autoregresivas y de orden causal (la salida de cada paso solamente depende de la información precedente).
- Embedding net: ANN recurrente, para capturar la dinámica de la serie temporal.
    - Ejemplo: embedding con convoluciones temporales y recovery con decodificador con atención, para modelar el contexto.
- En la propuesta de Yoon et al. se implementa la embedding net con una ANN recurrente y el recovery con una ANN feedforward en cada paso.
- Construida para manejar secuencias cortas (24 pasos de tiempo).

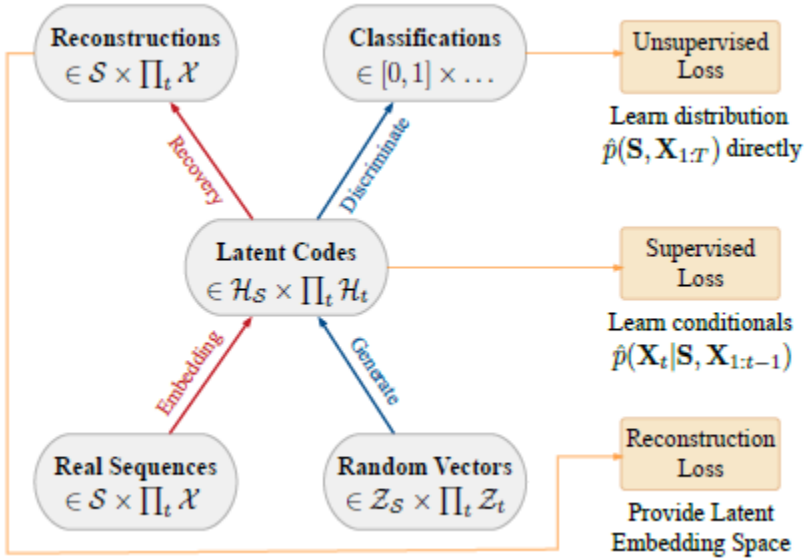
TimeGAN: implementación de Yoon et al. (2019)

- Generador: en lugar de producir datos sintéticos directamente en el espacio de características, el generador tiene su salida en el espacio de mapeo.
- Función de generación $g : \mathcal{Z}_S \times \prod_t \mathcal{Z}_X \to \mathcal{H}_S \times \prod_t \mathcal{H}_X$
  - Toma una tupla de vectores aleatorios estáticos y temporales.
  - $\mathcal{Z}_S$: espacio vectorial con la distribución de datos estáticos
  - $\mathcal{Z}_X$: espacio vectorial con la distribución de datos temporales
  - Se toman muestras de $\mathcal{Z}_S$ y $\mathcal{Z}_X$ para generar resultados en los espacios latentes $\mathcal{H}_S$ y $\mathcal{H}_X$
- Se implementa con una ANN recurrente.
- El generador debe ser autoregresivo.



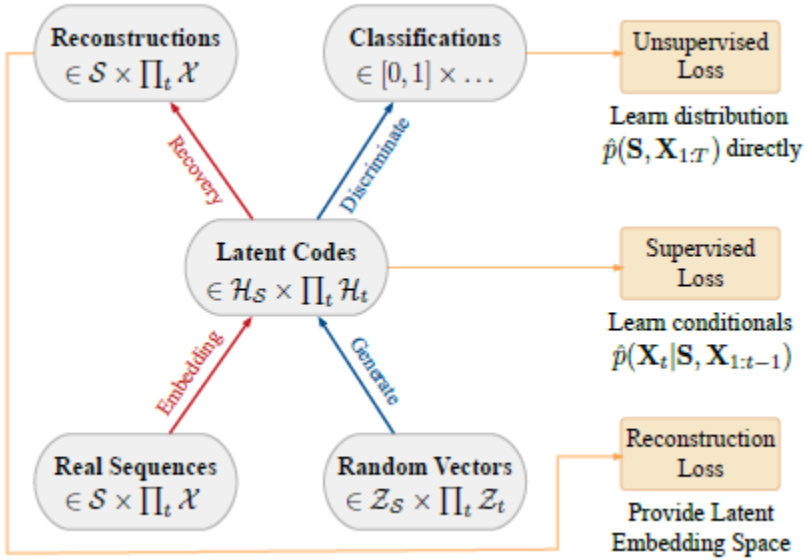
TimeGAN: implementación de Yoon et al. (2019)

- Embedding $e : \mathcal{S} \times \prod_t \mathcal{X} \to \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_\mathcal{X}$
- Toma como entrada características estáticas y temporales y determina sus codificaciones en el espacio latente.
- $e_\mathcal{S} : \mathcal{S} \to \mathcal{H}_\mathcal{S}$ embedding ANN para características estáticas
- $e_\mathcal{X} : \mathcal{H}_\mathcal{S} \times \mathcal{H}_\mathcal{X} \times \mathcal{X} \to \mathcal{H}_\mathcal{X}$ embedding ANN (recurrente) para características temporales.



TimeGAN: implementación de Yoon et al. (2019)

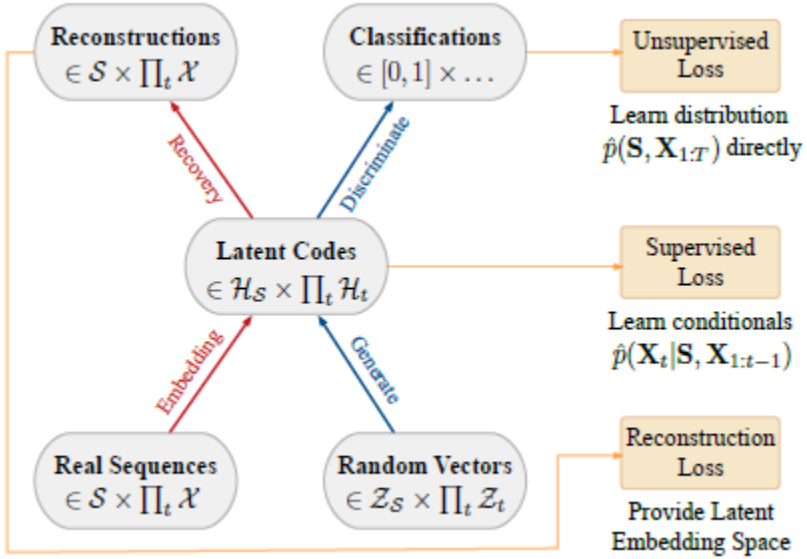
- Recovery $r : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \rightarrow \mathcal{S} \times \prod_t \mathcal{X}$
- Toma como entrada una representación en el espacio latente y retorna el vector de características estáticas y temporales.
- Se implementa como una ANN feedforward en cada paso
- $r_{\mathcal{S}} : \mathcal{H}_{\mathcal{S}} \rightarrow \mathcal{S}$ y $r_{\mathcal{X}} : \mathcal{H}_{\mathcal{X}} \rightarrow \mathcal{X}$ son las funciones de reconstrucción para embeddings estáticos y temporales.



TimeGAN: implementación de Yoon et al. (2019)

- Discriminador: toma su entrada del espacio de mapeo.
- Función de discriminación $d : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \rightarrow [0,1] \times \prod_t [0,1]$
- Toma una tupla de vectores (mapeos) estáticos y temporales y retorna clasificaciones para ambos componentes.

- Se implementa con una ANN recurrente bidireccional con una capa de salida feedforward



TimeGAN: funciones de pérdida de Yoon et al. (2019)

- Embedding/recovery: pérdida de reconstrucción

$$\mathcal{L}_{\mathrm{R}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T}\sim p}\left[\|\mathbf{s} - \tilde{\mathbf{s}}\|_2 + \sum_t \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|_2\right]$$

  - Reconstrucciones precisas de $\boldsymbol{s}$ y $\boldsymbol{x}_{1:T}$
- Generador y discriminador: función de pérdida no supervisada
  - El generador recibe embeddings sintéticos y sus propias salidas previas $\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{1:t-1}$ para generar el siguiente vector sintético $\hat{\mathbf{h}}_t$
  - Pérdida adversativa no supervisada para maximizar (discriminador) y minimizar (generador) la probabilidad de producir clasificaciones correctas $\hat{y}_{\mathcal{S}}, \hat{y}_{1:T}$ de los datos de entrenamiento $\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{1:T}$ y de los sintéticos $\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{1:t-1}$

$$\mathcal{L}_{\mathrm{U}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T}\sim p}\left[\log y_{\mathcal{S}} + \sum_t \log y_t\right] + \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T}\sim \hat{p}}\left[\log(1-\hat{y}_{\mathcal{S}}) + \sum_t \log(1-\hat{y}_t)\right]$$

TimeGAN: funciones de pérdida de Yoon et al. (2019)

- Supervisor: pérdida supervisada
  - La retroalimentación del resultado binario del discriminador no es suficiente para capturar las distribuciones condicionales escalonadas en los datos.
  - Ciclo cerrado: el generador recibe secuencias de embeddings de datos reales $\mathbf{h}_{1:t-1}$ (calculadas por la ANN de embedding) para generar el siguiente vector latente
  - Los gradientes capturan las discrepancias/diferencias entre las distribuciones $p(\mathbf{H}_t|\mathbf{H}_{\mathcal{S}}, \mathbf{H}_{1:t-1})$ y $\hat{p}(\mathbf{H}_t|\mathbf{H}_{\mathcal{S}}, \mathbf{H}_{1:t-1})$

$$\mathcal{L}_{\mathrm{S}} = \mathbb{E}_{\mathbf{s},\mathbf{x}_{1:T}\sim p}\left[\sum_t \|\mathbf{h}_t - g_{\mathcal{X}}(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{z}_t)\|_2\right]$$

  - $g_{\mathcal{X}}(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{z}_t)$ aproxima a $\mathbb{E}_{\mathbf{z}_t\sim\mathcal{N}}[\hat{p}(\mathbf{H}_t|\mathbf{H}_{\mathcal{S}}, \mathbf{H}_{1:t-1}, \mathbf{z}_t)]$ para cada muestra $\mathbf{z}_t$

TimeGAN: resumen, ventajas y desventajas

- En cada paso del entrenamiento se evalúa la diferencia entre el vector latente (embedding) del siguiente paso real y el vector latente sintético del siguiente paso (del generador, condicionado por la secuencia histórica real de latentes).
  - L_U presiona para generar secuencias realistas (evaluadas por D)
  - L_S presiona a generar transiciones similares (evaluadas por valores reales).
- Ventajas
  - El modelo es robusto (menos sensible a los cambios de hiperparámetros).
  - El proceso de entrenamiento es más estable, en comparación con otras arquitecturas.
- Desventajas
  - El modelo incluye varias ANN, la configuración y el entrenamiento es más complejo

TimeGAN: implementación de ejemplo/módulos

- data_loading.py: transforma datos en series de tiempo preprocesados y crea datos de función seno
- visualization_metrics.py: análisis PCA y t-SNE de datos originales y generados
- discriminative_metrics.py: usa RNN para clasificar datos originales y generados
- predictive_metrics.py: usa RNN para predecir un paso adelante (última característica)
- timegan.py: usa datos originales de series temporales como conjunto de entrenamiento para generar datos sintéticos de series temporales
- main_timegan.py: reporta score discriminativo y predictivo para el conjunto de datos y para el análisis t-SNE y PCA
- utils.py: funciones auxiliares para métricas y timeGAN.

TimeGAN: implementación de ejemplo

- Entradas
  - data_name: sine, stock, or energy; seq_len: largo de la serie
  - module: gru, lstm, or lstmLN; hidden_dim: dimensión de capas ocultas
  - num_layers: número de capas; iterations: número de iteraciones de entrenamiento
  - batch_size: número de muestras en cada batch
  - metric_iterations: número de iteraciones para el cálculo de las métricas
- Ejemplo
  - `python3.8 main.py --device cuda --exp test --is_train true --seed 43 --feat_pred_no 1 --max_seq_len 100 --train_rate 0.5 --batch_size 128 --hidden_dim 20 --num_layers 3 --dis_thresh 0.15 --optimizer adam --learning_rate 1e-3`
- Salidas
  - ori_data: datos originales; generated_data: datos sintéticos generados
  - metric_results: puntajes discriminativos y predictivos
  - visualization: análisis PCA y tSNE

TimeGAN: resultados del entrenamiento (stock data)

Start Embedding Network Training

```
step: 0/10000, e_loss: 0.2374
step: 1000/10000, e_loss: 0.119
step: 2000/10000, e_loss: 0.0865
step: 3000/10000, e_loss: 0.0786
step: 4000/10000, e_loss: 0.0713
step: 5000/10000, e_loss: 0.0662
step: 6000/10000, e_loss: 0.0625
step: 7000/10000, e_loss: 0.0589
step: 8000/10000, e_loss: 0.0565
step: 9000/10000, e_loss: 0.0529
```

Finish Embedding Network Training

Start Training with Supervised Loss Only

```
step: 0/10000, s_loss: 0.2294
step: 1000/10000, s_loss: 0.0352
step: 2000/10000, s_loss: 0.03
step: 3000/10000, s_loss: 0.0295
step: 4000/10000, s_loss: 0.0289
step: 5000/10000, s_loss: 0.028
step: 6000/10000, s_loss: 0.0288
step: 7000/10000, s_loss: 0.0282
step: 8000/10000, s_loss: 0.0279
step: 9000/10000, s_loss: 0.028
```

Finish Training with Supervised Loss Only

Start Joint Training

```
step: 0/10000, d_loss: 1.98, g_loss_u: 0.8229, g_loss_s: 0.0381, g_loss_v: 0.3665, e_loss_t0: 0.1048
step: 1000/10000, d_loss: 1.7816, g_loss_u: 1.106, g_loss_s: 0.034, g_loss_v: 0.0538, e_loss_t0: 0.0476
step: 2000/10000, d_loss: 1.8588, g_loss_u: 1.141, g_loss_s: 0.0331, g_loss_v: 0.0321, e_loss_t0: 0.0441
step: 3000/10000, d_loss: 1.7955, g_loss_u: 1.4317, g_loss_s: 0.0343, g_loss_v: 0.0362, e_loss_t0: 0.0425
step: 4000/10000, d_loss: 1.5946, g_loss_u: 1.4365, g_loss_s: 0.0336, g_loss_v: 0.0334, e_loss_t0: 0.0415
step: 5000/10000, d_loss: 1.5406, g_loss_u: 1.4571, g_loss_s: 0.0338, g_loss_v: 0.0368, e_loss_t0: 0.0402
```

TimeGAN: resultados (stock data)

(1200 épocas para el supervisor y el entrenamiento conjunto)

Model Runtime: 39.62085920174916 mins

```
Running feature prediction using original data...
Epoch: 19, Loss: 0.0476:
100%|████████████████████████████████████████████████████████████████████████████| 20/20
[00:02<00:00,  8.85it/s]
Epoch: 19, Loss: 0.0463:
100%|████████████████████████████████████████████████████████████████████████████| 20/20
[00:02<00:00,  8.95it/s]
Running feature prediction using generated data...
Epoch: 19, Loss: 0.1046:
100%|████████████████████████████████████████████████████████████████████████████| 20/20
[00:02<00:00,  8.93it/s]
Epoch: 19, Loss: 0.7370:
100%|████████████████████████████████████████████████████████████████████████████| 20/20
[00:02<00:00,  8.92it/s]
```

Feature prediction results:
(1) Ori: [0.1353 0.2076]
(2) New: [0.1372 0.2534]

TimeGAN: resultados (stock data)

(1200 épocas para el supervisor y el entrenamiento conjunto)

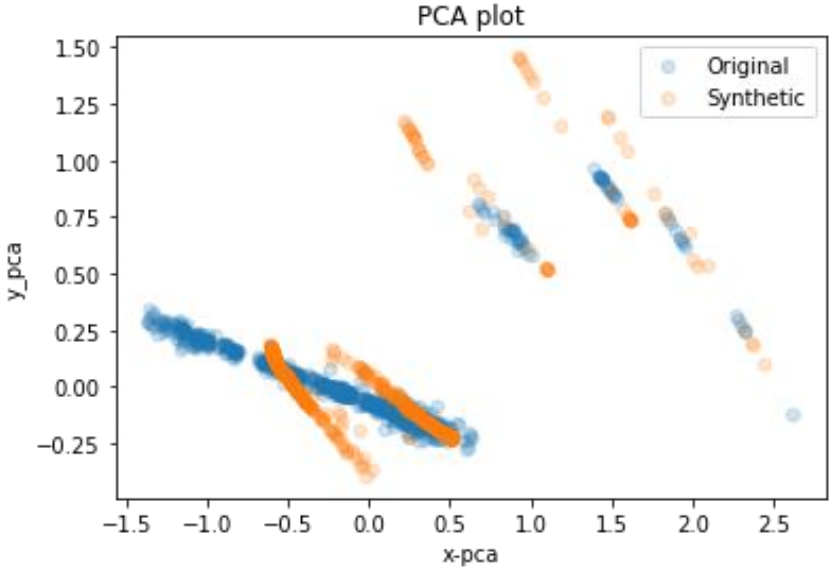
```
Running one step ahead prediction using original data...
Epoch: 19, Loss: 0.5352:
100%|████████████████████████████████████████████████████████████████████████████████████████████████████████████|
20/20 [00:02<00:00,  8.82it/s]
Running one step ahead prediction using generated data...
Epoch: 19, Loss: 1.7848:
100%|████████████████████████████████████████████████████████████████████████████████████████████████████████████|
20/20 [00:02<00:00,  8.84it/s]
```

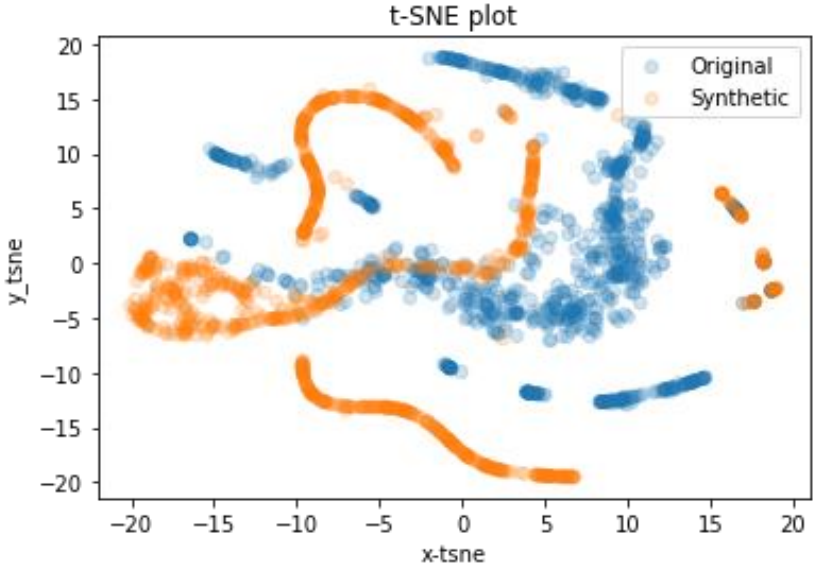
One step ahead prediction results:
(1) Ori: 0.349
(2) New: 0.3502

```
[t-SNE] Computing 121 nearest neighbors. Indexed 2000 samples in 0.000s...
[t-SNE] Computed neighbors for 2000 samples in 0.110s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2000
[t-SNE] Computed conditional probabilities for sample 2000 / 2000
[t-SNE] Mean sigma: 0.006041
[t-SNE] KL divergence after 250 iterations with early exaggeration: 50.7230
[t-SNE] KL divergence after 300 iterations: 0.558803
```

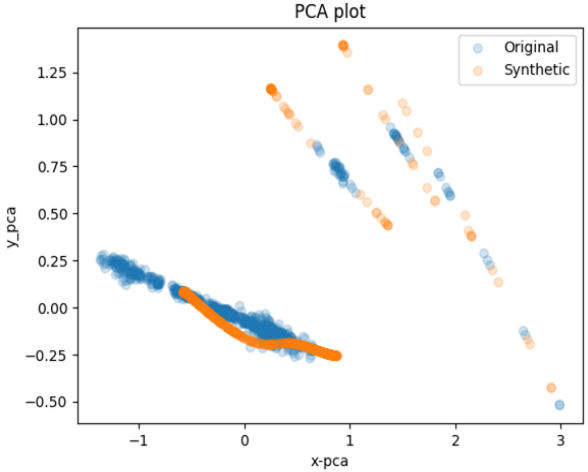
TimeGAN: resultados (stock data)

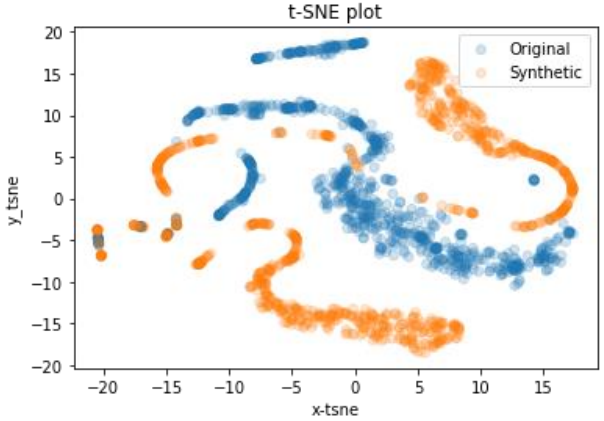
- Principal Component Analysis (PCA) describe un conjunto de datos en términos de nuevas variables (componentes) no correlacionadas, ordenadas por varianza.
- t-Distributed Stochastic Neighbor Embedding (tSNE), mapea datos de alta dimensión a una variedad de menor dimensión, creando un embedding que intenta mantener la estructura local de los datos.

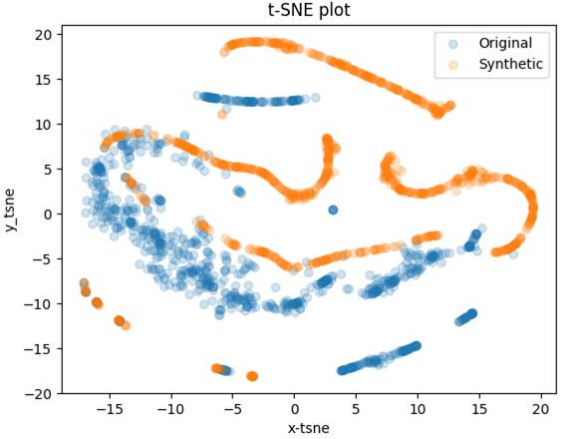


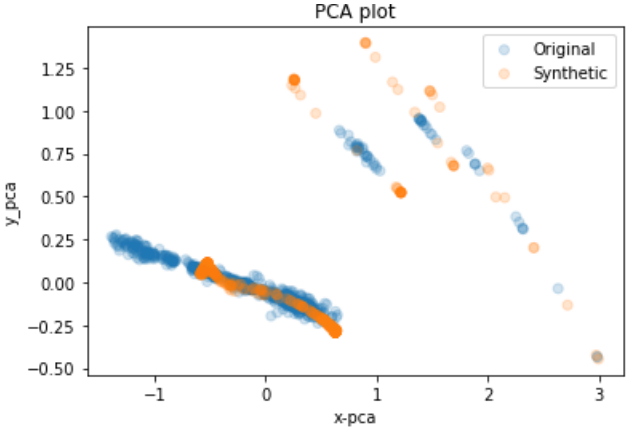


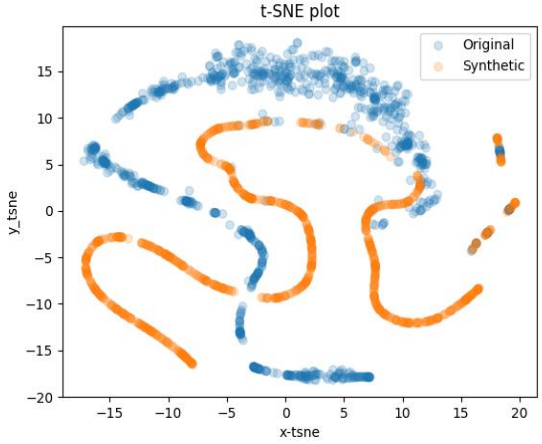
TimeGAN: resultados (stock data)

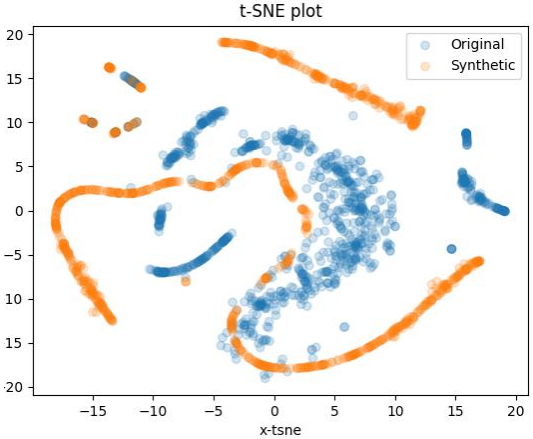












TimeGAN: resultados (stock data)

- Implementación en Tensorflow por Yoon et al. (2019).

```
$ python3 main_timegan.py --data_name stock --seq_len 24 --module gru --hidden_dim
24 --num_layer 3 --iteration 10000 --batch_size 128 --metric_iteration 10

with 11346 MB memory) -> physical GPU (device: 0, name: Tesla P100-PCIE-12GB, pci bus id:
0000:86:00.0, compute capability: 6.0)
Start Embedding Network Training
2021-09-20 04:33:19.888158: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library libcublas.so.10.0
step: 0/5000, e_loss: 0.3581
step: 1000/5000, e_loss: 0.021
step: 2000/5000, e_loss: 0.0138
step: 3000/5000, e_loss: 0.0086
step: 4000/5000, e_loss: 0.0095
Finish Embedding Network Training

Start Training with Supervised Loss Only
step: 0/5000, s_loss: 0.2216
step: 1000/5000, s_loss: 0.0221
step: 2000/5000, s_loss: 0.019
step: 3000/5000, s_loss: 0.0168
step: 4000/5000, s_loss: 0.0171
Finish Training with Supervised Loss Only
```

TimeGAN: resultados (stock data)

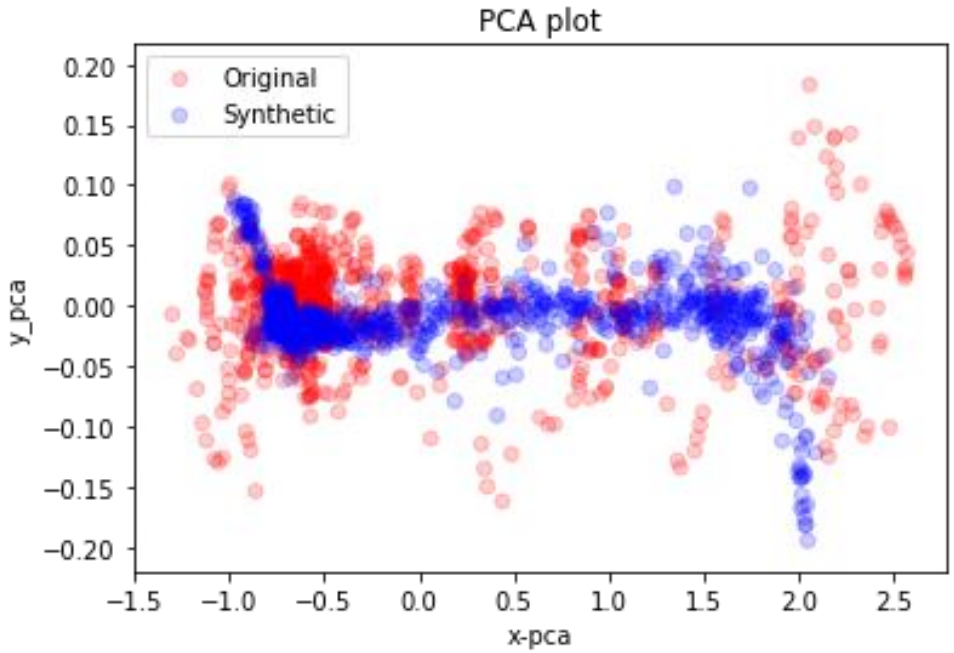
- Implementación en Tensorflow por Yoon et al. (2019).

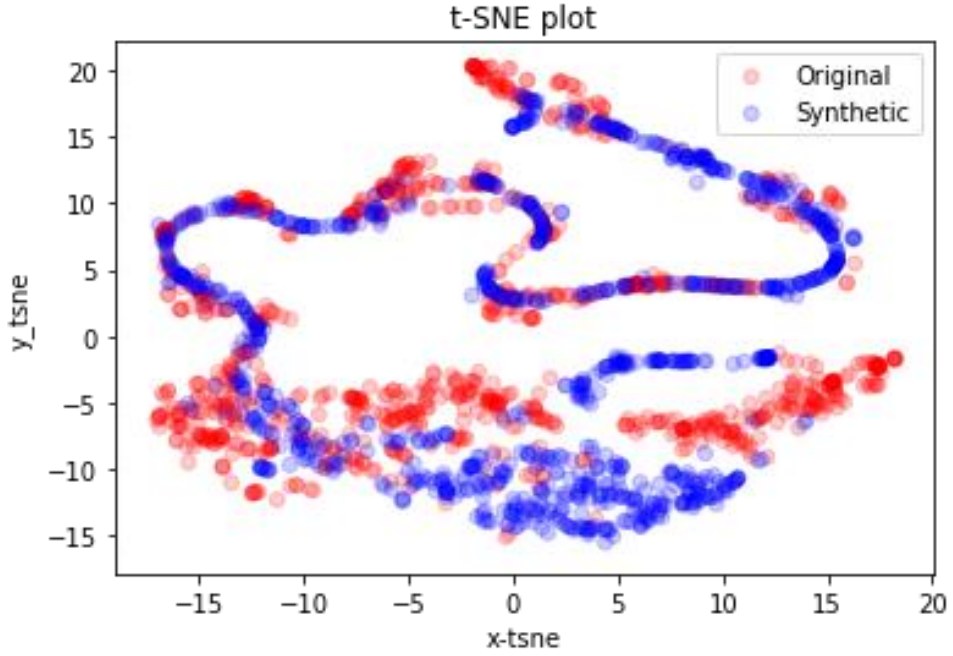
```
Start Joint Training
step: 0/5000, d_loss: 2.1191, g_loss_u: 0.6552, g_loss_s: 0.0273, g_loss_v: 0.332, e_loss_t0: 0.0711
step: 1000/5000, d_loss: 1.8125, g_loss_u: 1.5717, g_loss_s: 0.0227, g_loss_v: 0.047, e_loss_t0: 0.0032
step: 2000/5000, d_loss: 1.6669, g_loss_u: 1.3674, g_loss_s: 0.0226, g_loss_v: 0.0212, e_loss_t0: 0.0038
step: 3000/5000, d_loss: 1.8194, g_loss_u: 1.1651, g_loss_s: 0.023, g_loss_v: 0.0279, e_loss_t0: 0.0028
step: 4000/5000, d_loss: 1.7834, g_loss_u: 0.992, g_loss_s: 0.0206, g_loss_v: 0.0376, e_loss_t0: 0.0028
Finish Joint Training

Finish Synthetic Data Generation
[t-SNE] Computing 121 nearest neighbors … Indexed 2000 samples in 0.000s...
[t-SNE] Computed neighbors for 2000 samples in 0.097s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2000
[t-SNE] Computed conditional probabilities for sample 2000 / 2000
[t-SNE] Mean sigma: 0.019280
[t-SNE] KL divergence after 250 iterations with early exaggeration: 51.817013
[t-SNE] KL divergence after 300 iterations: 0.678407
{'discriminative': 0.15757162346521145, 'predictive': 0.04041279854748552}
```

TimeGAN: resultados (stock data)

- Implementación en Tensorflow por Yoon et al. (2019).





Normalized relative discriminative score: 0.1759, predictive score (MAE): 0.0412

Otras propuestas y extensiones

- Diseño para manejar secuencias temporales más largas (entre 100/1000 pasos de tiempo) utilizando Temporal Convolutional Networks (TCN) (Viklund 2021).
- VAEACGAN (Yin et al. 2021): Variational Auto-Encoder Conditional Generative Adversarial Network
  - Encoder ANN, para codificar secuencias en vectores del espacio latente, almacenando la información de la secuencia.
  - Generador de datos para predicción.
  - Discriminador para clasificación y feedback al generador, como en una GAN tradicional.
- Survey sobre GANs para series temporales (Brophy et al, 2021): https://arxiv.org/pdf/2107.11098.pdf

TimeGAN: Implementaciones de ejemplo

- Implementación en pytorch
  https://colab.research.google.com/drive/19tZ6nfv4um7PCAFW91mySAaivucM0wNf

  https://github.com/nesmachnow/TimeGAN

- Implementación original en Tensorflow
  https://github.com/jsyoon0823/TimeGAN