Generación de datos de consumo eléctrico con GANs

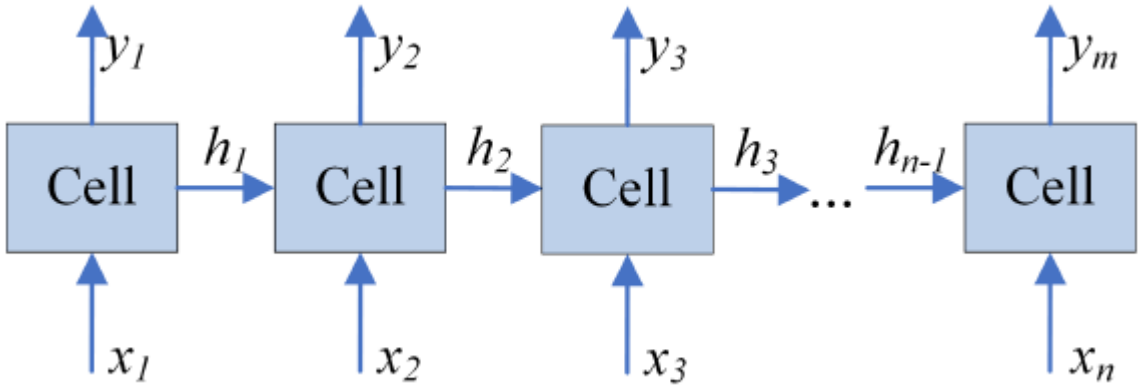
GANs para datos de consumo eléctrico

- Generating Energy Data for Machine Learning with Recurrent Generative Adversarial Networks. M. Fekri, A. Ghosh, K. Grolinger. Energies 13(130):1-23, 2020.
- R-GANs: usan ANN recurrentes (RNNs) para el generador y el discriminador.
- Requiere un preprocesamiento de los datos de entrada para ajustarlos a la arquitectura de las RNN utilizadas:
  1. Análisis de características: ARIMA y transformada de Fourier;
  2. Normalización;
  3. Generación de muestras con un método de Ventana deslizante.

Redes neuronales recurrentes

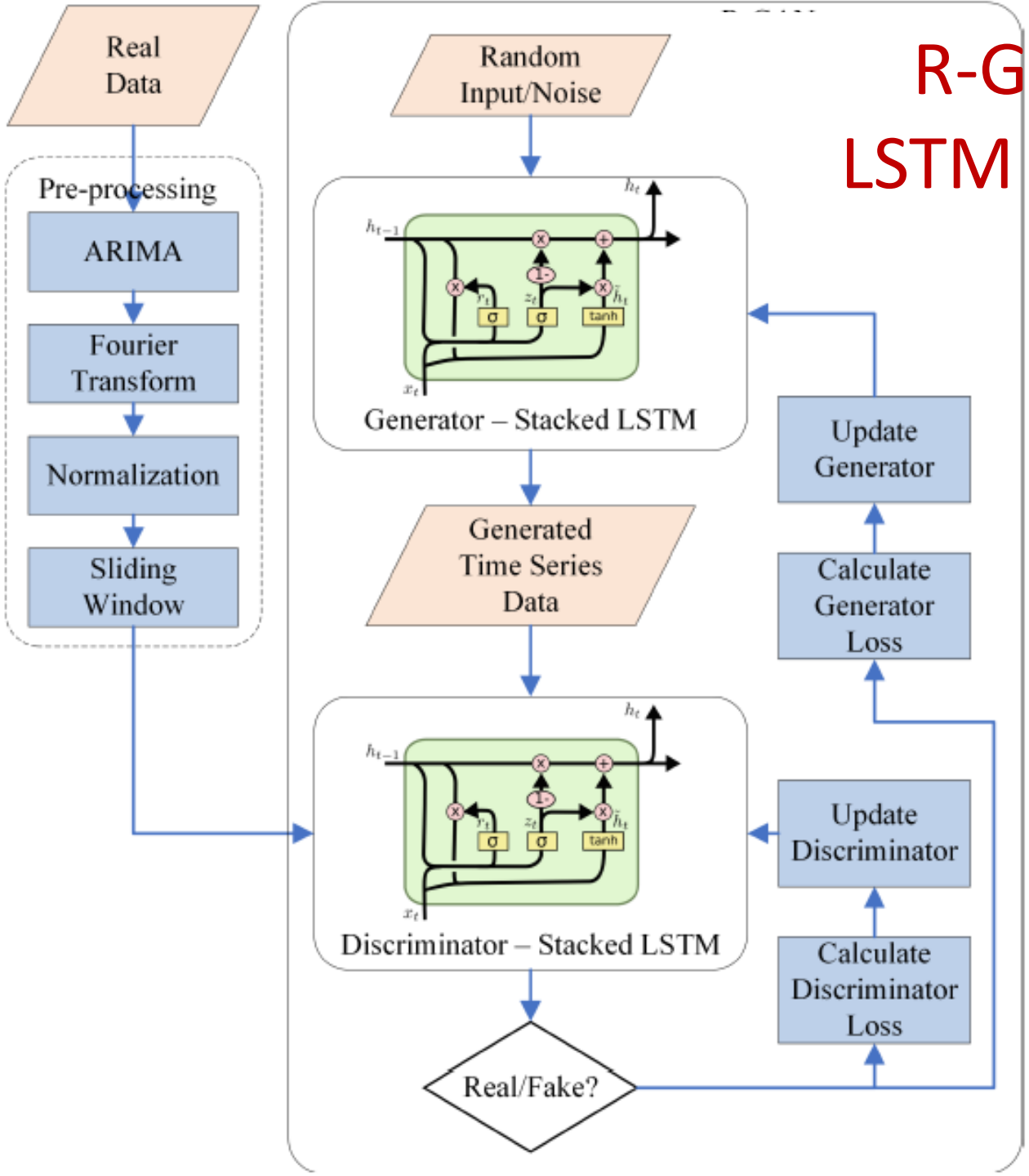
- ANN diseñadas para procesar datos secuenciales, como las series temporales.
- A diferencia de las ANN feedforward, que solo consideran la entrada actual para calcular la salida, las RNN consideran la entrada actual en conjunto con todas las entradas recibidas previamente, mediante el uso de conexiones 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 a largo-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.
- Ejemplo: RNN de una sola capa.
- 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.

R-GAN



R-GAN: WGAN + LSTM cells + MHGAN

R-GAN: pre-procesamiento

- Se trabaja a partir de un conjunto de características de base (core features): consumo, fecha/hora de las mediciones, día de semana/fin de semana, etc.
- Las core features se enriquecen a través de 'feature engineering'.
- Modelo de media móvil integrada auto regresiva (Auto regressive integrated moving average, ARIMA): se ajusta a las series temporales para comprender mejor los datos o para pronosticar valores futuros.
  - Componente autoregresivo (AR): modela la variable de interés como regresión de sus propios valores pasados.
  - Integrada (I) refiere a la capacidad de lidiar con la no estacionariedad.
  - Componente de promedio móvil (MA) usa una combinación lineal de términos de error pasados para el modelo.

R-GAN: pre-procesamiento (ARIMA)

- Se utiliza ARIMA debido a su capacidad para capturar diferentes estructuras temporales en la característica consumo energético.
- Los valores obtenidos del modelo ARIMA ajustado se agregan como una nueva característica al conjunto de datos existente.
- La nueva característica ARIMA mejora la capacidad de la RNN para capturar dependencias temporales y modelar el tiempo.

R-GAN: pre-procesamiento (Transformada de Fourier)

- La Transformada de Fourier (FT) descompone una señal de tiempo en sus representaciones en el dominio de frecuencia (e.g., suma de funciones sinusoidales).
- La FT inversa sintetiza la señal original a partir de sus representaciones en el dominio de frecuencia.
- FT identifica las frecuencias presentes en la señal original: es adecuada para descubrir ciclos en los datos.

R-GAN: pre-procesamiento

- FT se aplica a la función de consumo energético.
- Se descompone la serie temporal en representaciones sinusoidales, se seleccionan las n frecuencias dominantes y se construye una nueva serie temporal utilizando estas n señales constituyentes. La nueva serie temporal representa una nueva característica.
- Cuando el número de componentes n es bajo, la nueva serie temporal solo captura los ciclos más dominantes, mientras que para un gran número de componentes, la serie temporal creada se acerca a la serie temporal original.
- Un valor de n crea una nueva característica, pero se utilizan varios valores con sus características correspondientes capturar dependencias temporales y mejorar la calidad de los datos generados.

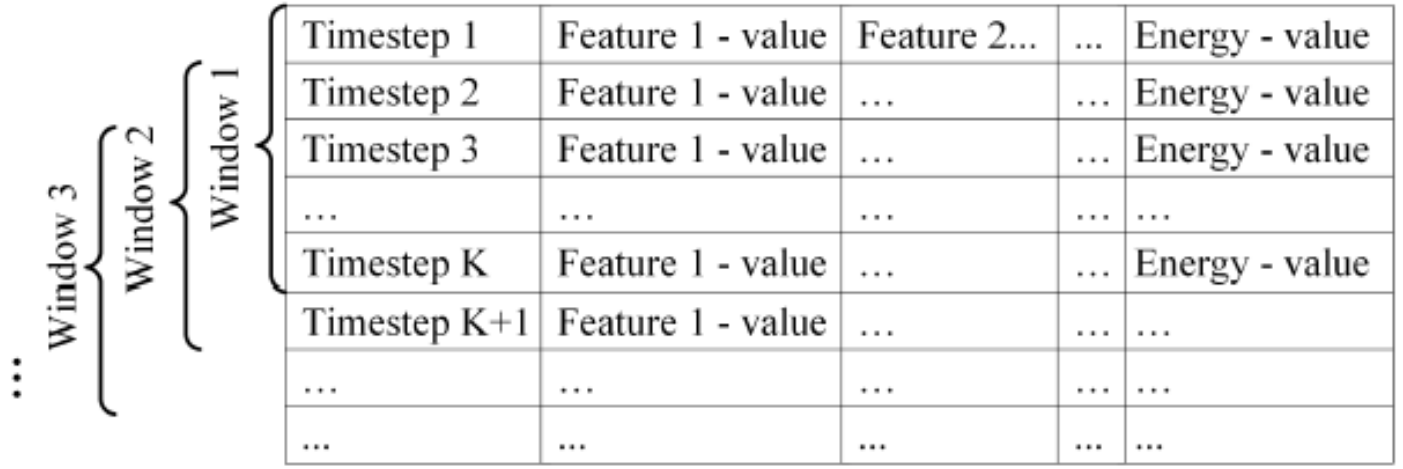
R-GAN: pre-procesamiento (normalización)

- Normalización MinMax, para llevar los valores de todas las características a una escala similar.
- Escala valores de cada característica al intervalo [0,1] mediante

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

R-GAN: pre-procesamiento (ventana deslizante)

- Convierte datos en formato matriz (las filas contienen valores de características originales y añadidas para cada paso de tiempo) a un formato apropiado para RNN.
- K filas corresponden a la primera ventana de tiempo y constituyen la primera muestra de entrenamiento.
- La ventana se desliza usando un paso S y las lecturas de los pasos S a $K + S$ forman la segunda muestra.
- Cada muestra es una matriz de dimensión $K \times F$, donde K es la longitud de la ventana y F es el número de características.

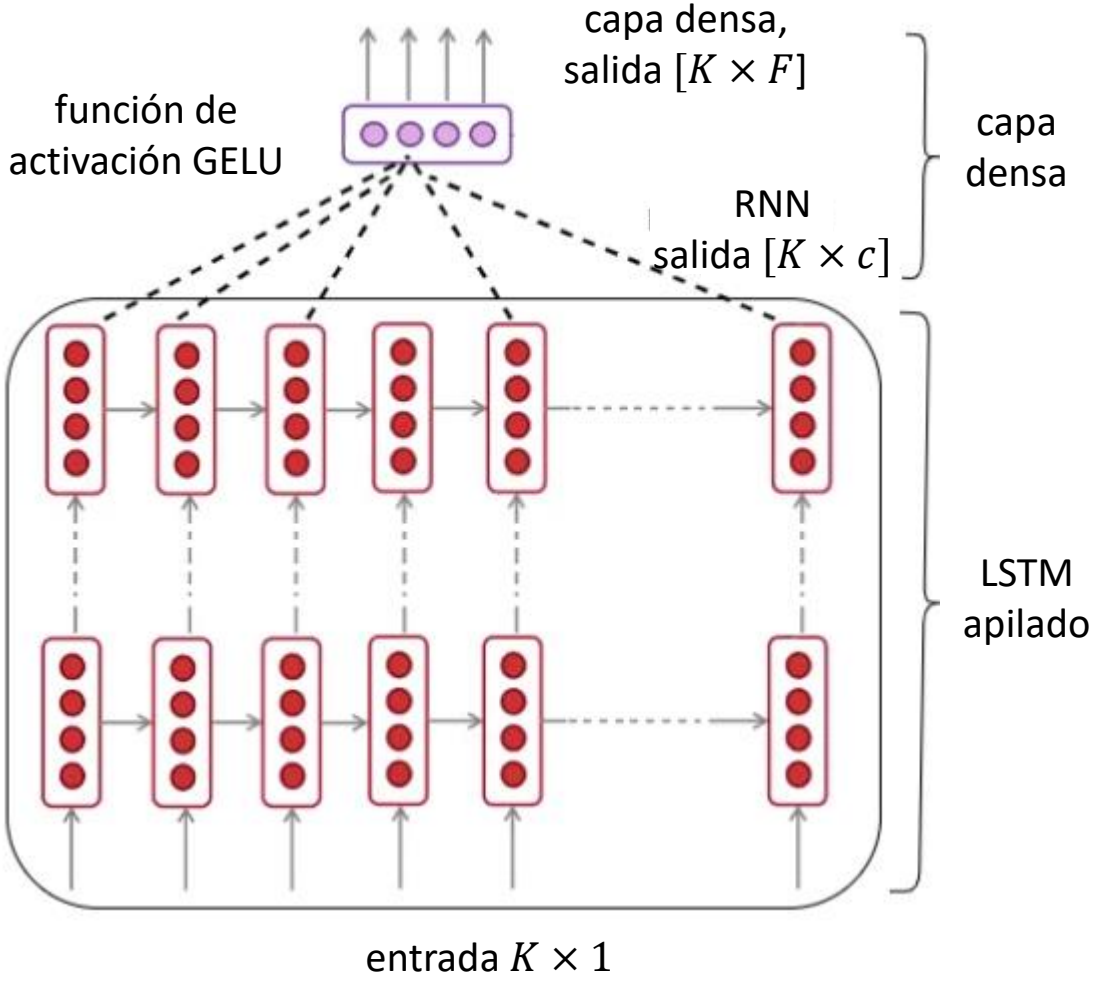


R-GAN: arquitectura

- En una GAN estándar se usa CNN para el generador y el discriminador, R-GAN sustituye CNN con una arquitectura de celdas LSTM apiladas y una sola capa densa.
- Las celdas LSTM apiladas permiten almacenar información para secuencias más largas que las celdas RNN estándar. Además, las capas ocultas apiladas permiten capturar patrones en diferentes escalas de tiempo.

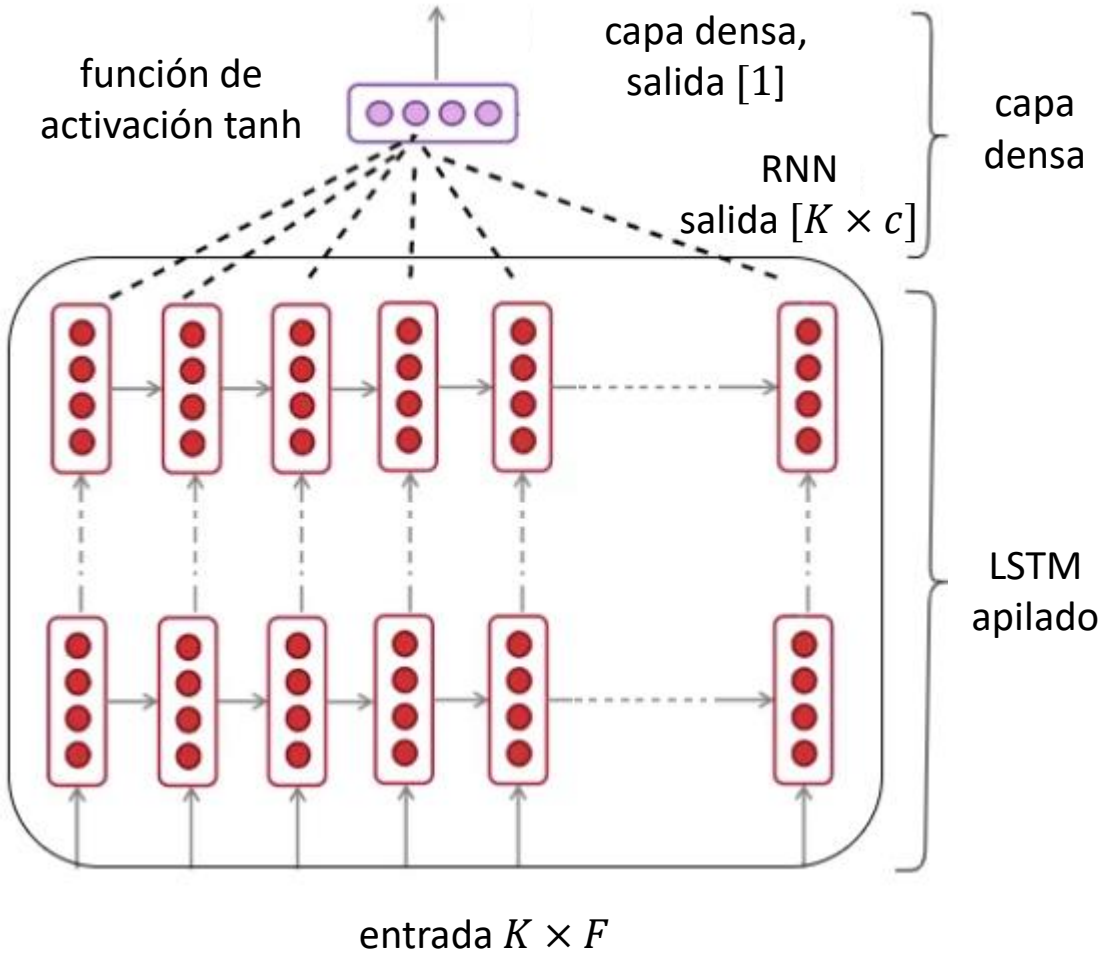
R-GAN: generador

- La entrada tiene como dimensión la longitud de la ventana deslizante $[K \times 1]$.
- La salida de la RNN tiene dimensión longitud de la ventana × dimensión del estado de la celda LSTM $[K \times c]$.
- Los datos generados tienen la misma dimensión que los datos reales preprocesados: longitud de ventana × número de características $[K \times F]$.
- Se utiliza Gaussian error linear unit (GELU) como función de activación (mejora sobre ReLU y otras).



R-GAN: discriminador

- La entrada tiene como dimensión la de los datos preprocesados/salida del generador $[K \times F]$.
- La salida de la RNN tiene dimensión longitud de la ventana × dimensión del estado de la celda LSTM $[K \times c]$.
- La salida de la capa densa es binaria (true/false).
- Se utiliza $\mathbf{tanh}$ como función de activación.



R-GAN: funcionamiento

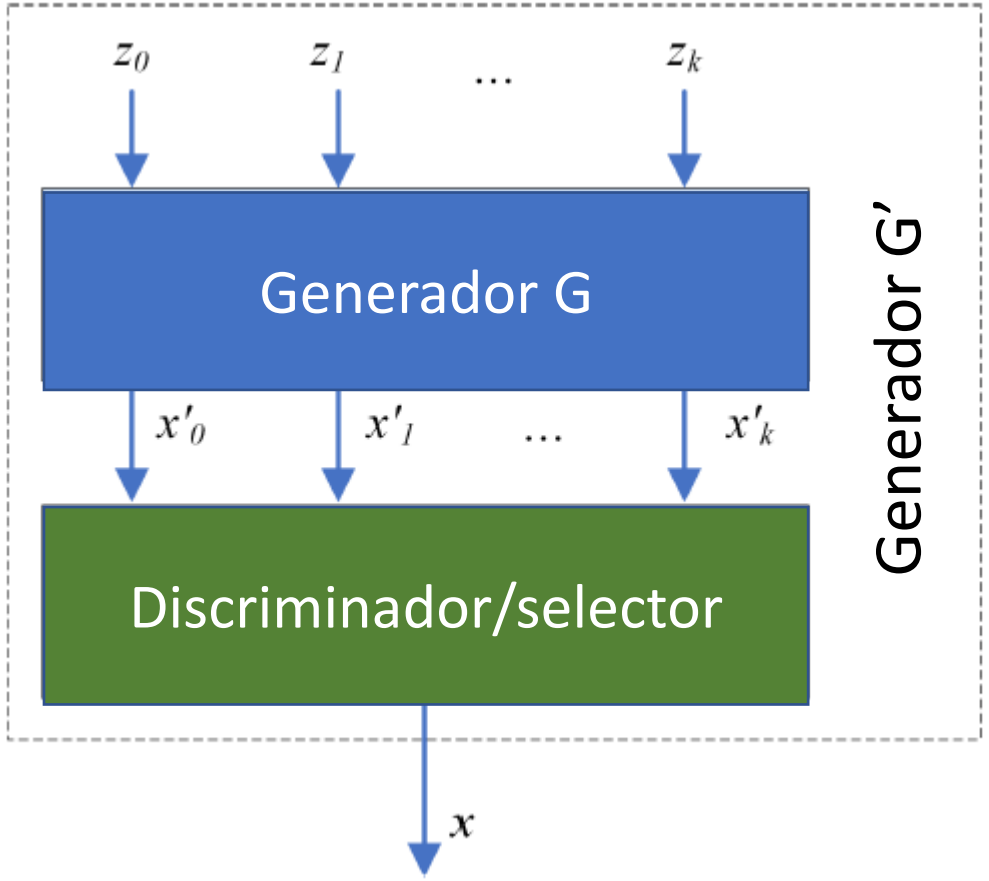
- Los datos generados junto con los datos preprocesados se pasan al discriminador que aprende a diferenciar entre muestras reales y falsas.
- Para cada minibatch (que consta de varias muestras reales y generadas), se calcula la pérdida del discriminador y se actualizan sus pesos
- R-GAN usa WGAN, las actualizaciones se realizan de manera ligeramente diferente que en una GAN estándar: en lugar de actualizar el discriminador y luego actualizar el generador, WGAN entrena al discriminador relativamente bien antes de cada actualización del generador.
- Existen varios ciclos de actualización del discriminador antes de cada actualización del generador.

R-GAN: generación de datos

- Luego del entrenamiento, la R-GAN está lista para generar datos.
- En una GAN tradicional solo se usa el generador [si su entrenamiento fuera perfecto, la distribución generada sería la misma que la real].
- Como no siempre el entrenamiento de una GAN converge a la distribución de datos reales, utilizar directamente un generador imperfecto puede producir muestras de baja calidad.
- Se aplica el enfoque Metropolis-Hastings GAN, que utiliza el generador y el discriminador en el proceso de generación de muestras.

Generación de datos con Metropolis-Hastings GAN

- El discriminador, junto con el generador entrenado G, forma un nuevo generador G'.
- El generador G toma como entrada muestras aleatorias $\{z_0, z_1, \ldots, z_k\}$ y produce muestras de series de tiempo $\{x_0', x_1', \ldots, x_k'\}$.
- Algunas muestras generadas están más cerca de la distribución de datos reales que otras.
- El discriminador funciona como selector para elegir la mejor muestra x del conjunto $\{x_0', x_1', \ldots, x_k'\}$.
- El resultado final es la muestra x de la serie temporal generada.



R-GAN: evaluación

- Se evalúa la calidad de modelos predictivos entrenados con los datos sintéticos generados.
- Entrenamiento con datos sintéticos, prueba en datos reales (Train on Synthetic, Test on Real, TSTR): se entrena un modelo predictivo con datos sintéticos y se prueba con datos reales.
- Se usa una RNN para predicción, diferente de las RNN utilizadas para el generador y discriminador.
- TSTR evalúa la capacidad de los datos sintéticos como datos de entrenamiento del modelo predictivo. Si R-GAN sufre mode collapse, TSTR se degrada porque los datos generados no capturan la diversidad de los datos reales y el modelo predictivo no es capaz de modelarla.

R-GAN: evaluación

- Entrenamiento con datos reales, prueba con datos sintéticos (TRTS): proceso inverso de TSTR, con los roles de datos sintéticos y reales invertidos.
- TRTS evalúa la capacidad de GAN para generar datos de aspecto realista.
- TRTS no se ve afectado por mode collapse, ya que una diversidad limitada de los datos sintéticos no afecta la precisión de la predicción.
- Como el objetivo es generar datos para entrenar modelos de aprendizaje, TSTR es una métrica más significativa que TRTS.

R-GAN: evaluación

- Entrenar con datos reales, probar con datos reales (Train on Real, Test on Real, TRTR): Evaluación estándar del modelo, entrenado y probado en los datos reales (con conjuntos separados de entrenamiento y prueba).
- TRTR no evalúa la calidad de los datos sintéticos en sí, pero proporciona una línea de base para comparar la precisión de un modelo entrenado con datos reales y sintéticos.
- Precisión baja en TRTR y TSTR indica que el modelo predictivo no es capaz de capturar variaciones en los datos. No implica baja calidad de los datos sintéticos.
- El objetivo de la R-GAN para generar datos de energía es lograr un valor de TSTR comparable al valor TRTR [independientemente de sus valores absolutos]: implica que el modelo entrenado con datos sintéticos tiene capacidades similares a las del modelo entrenado con datos reales.

R-GAN: evaluación

- Entrenar con datos sintéticos, probar con datos sintéticos (Train on Synthetic, Test on Synthetic, TSTS): similar a TRTR, TSTS evalúa la capacidad del modelo predictivo para capturar variaciones en los datos [en este caso se evalúa la precisión con datos sintéticos].
- Discrepancias grandes entre TRTR y TSTS indican que el modelo es mucho mejor con datos reales que con sintéticos (o viceversa). Esto significa que los datos sintéticos no muestrean apropiadamente la distribución de los datos reales.

R-GAN: resultados (Fekri et al. 2020)

- Evaluación sobre el conjunto de datos de energía UCI (Universidad de California, Irvine)
- UCI consta de lecturas de consumo de energía para diferentes aparatos e incluye atributos adicionales (temperatura y humedad).
- El intervalo de lectura es de 10 min y el número total de muestras es 19736.
- Se crearon las características día de la semana y día del mes a partir de la fecha de lectura, lo que resultó en un total de 28 características.

R-GAN: evaluación

- Se aplicó ARIMA para crear una característica adicional (modelo ajustado ARIMA para el conjunto de datos UCI).
- Se aplicó FT con cuatro números diferentes de componentes (transformaciones con 1, 10, 100 y 1000 componentes).
  - Utilizando un componente se obtiene como resultado valores casi constantes.
  - Utilizando 10 componentes se capturan solo tendencias a gran escala.
  - Con 100 y 1000 componentes se capturan más cambios a menor escala y la representación se acerca más a los datos originales.
- Las cuatro transformaciones con 1, 10, 100 y 1000 componentes forman cuatro características adicionales.

R-GAN: evaluación

- Se generaron todas las características adicionales (un total de 33 características) y se continuó con la normalización.
- Para preparar los datos para la RNN, se aplicó la técnica de ventana deslizante con una longitud de la ventana $K = 60$ (60 pasos de tiempo forman una muestra) y paso $S = 30$ (la ventana se desliza por 30 pasos de tiempo para generar la siguiente muestra).
- Los valores de tamaño de ventana y paso se determinaron a partir de experimentos iniciales.

R-GAN: evaluación

- La R-GAN se implementó en Python con Tensorflow y el entrenamiento se realizó sobre GPU.
- La arquitectura LSTM apilad del discriminador y el generador utilizó:
  - Número de capas $L = 2$
  - Tamaño de la dimensión del estado de la celda $c = 128$
  - Tasa de aprendizaje = 2×10^{-6}
  - Tamaño del batch = 100
  - Optimizador = Adam

R-GAN: evaluación

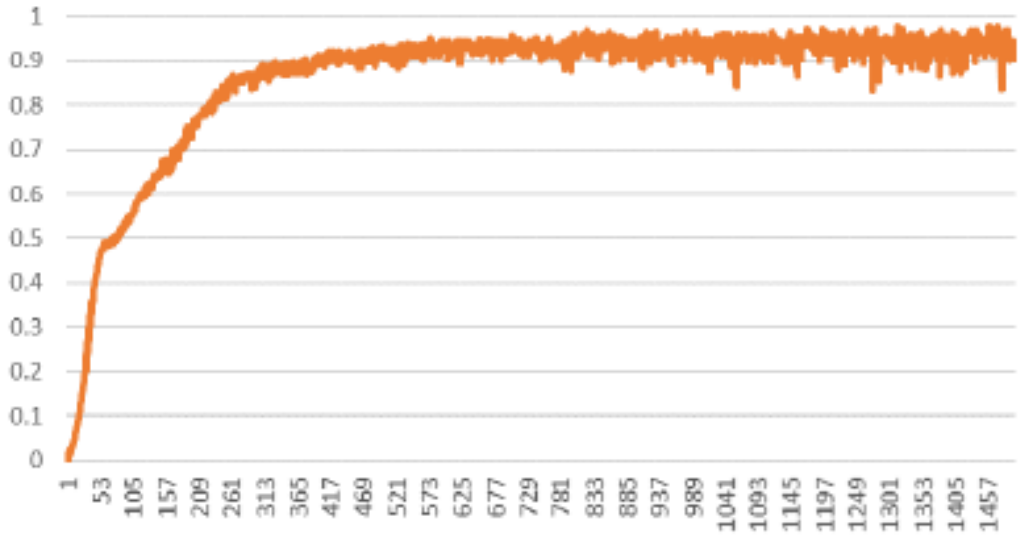
- La entrada al generador consistió en muestras de tamaño 60 (para coincidir con la longitud de la ventana) extraídas de una distribución gaussiana.
- La salida del generador fue de tamaño 60×33 (número de características × longitud de la ventana).
- La entrada del discriminador tuvo la misma dimensión que la salida del generador y los datos reales preprocesados.
- El rendimiento se degrada para tamaños de batch más grandes que los de uso común (30–500).

R-GAN: evaluación

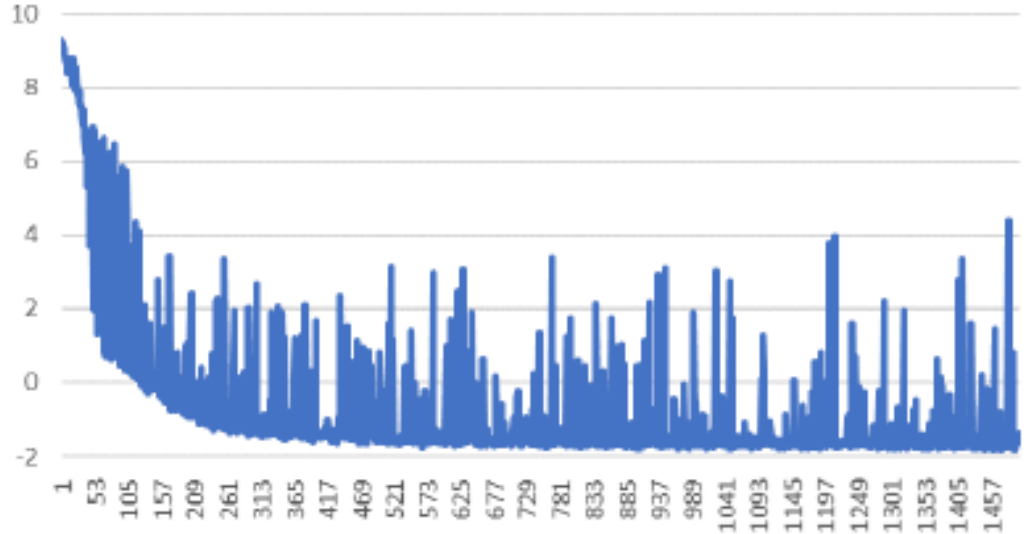
- Para cada batch, 100 muestras sintéticas generadas y 100 muestras reales seleccionadas al azar se pasaron al discriminador para su clasificación.
- El aumento de la dimensión del estado de la celda normalmente conduce a una mayor precisión de LSTM, pero también aumenta el tiempo de entrenamiento; por lo tanto, se seleccionó un tamaño moderado de 128.
- La tasa de aprendizaje es el parámetro más relevante para redes LSTM y a menudo hay un amplio rango (hasta dos órdenes de magnitud) de buenos valores para la tasa de aprendizaje.

R-GAN: resultados

Pérdida del generador



Pérdida del discriminador



Features	MAPE (%)				MAE			
	TRTS	TRTR	TSTR	TSTS	TRTS	TRTR	TSTR	TSTS
Core features	13.60%	17.98%	18.67%	18.80%	54.26	63.82	62.74	90.90
Core and ARIMA features	8.65%	11.43%	11.37%	8.92%	48.14	62.67	54.00	80.00
Core and FT features	9.07%	15.84%	17.79%	15.10%	48.99	63.12	61.74	90.67
Core, ARIMA and FT features	5.28%	10.81%	10.12%	6.80%	46.41	62.27	52.54	78.35

R-GAN: resultados

- Utilidad de los datos sintéticos para entrenar modelos: TSTR es métrica crucial.
- Incluir características ARIMA y FT mejora la precisión en términos de MAPE y MAE (reduce MAPE de 18.67% a 10.12% y MAE de 62.74 a 52.54).
- Los modelos predictivos tienen errores entrenados con datos reales, se debe comparar la precisión del modelo entrenado con datos sintéticos y reales:
- Valores cercanos de TSTR y TRTR para todos los modelos, independientemente de las características (MAPE TRTR: 10,81%; MAPE TRTS 10,12%).
- Se demuestra la usabilidad de los datos sintéticos para el entrenamiento de modelos de aprendizaje.

Features	Mean absolute percentage error (MAPE)				mean absolute error (MAE)			
	TRTS	TRTR	TSTR	TSTS	TRTS	TRTR	TSTR	TSTS
Core features	13.60%	17.98%	18.67%	18.80%	54.26	63.82	62.74	90.90
Core and ARIMA features	8.65%	11.43%	11.37%	8.92%	48.14	62.67	54.00	80.00
Core and FT features	9.07%	15.84%	17.79%	15.10%	48.99	63.12	61.74	90.67
Core, ARIMA and FT features	5.28%	10.81%	10.12%	6.80%	46.41	62.27	52.54	78.35

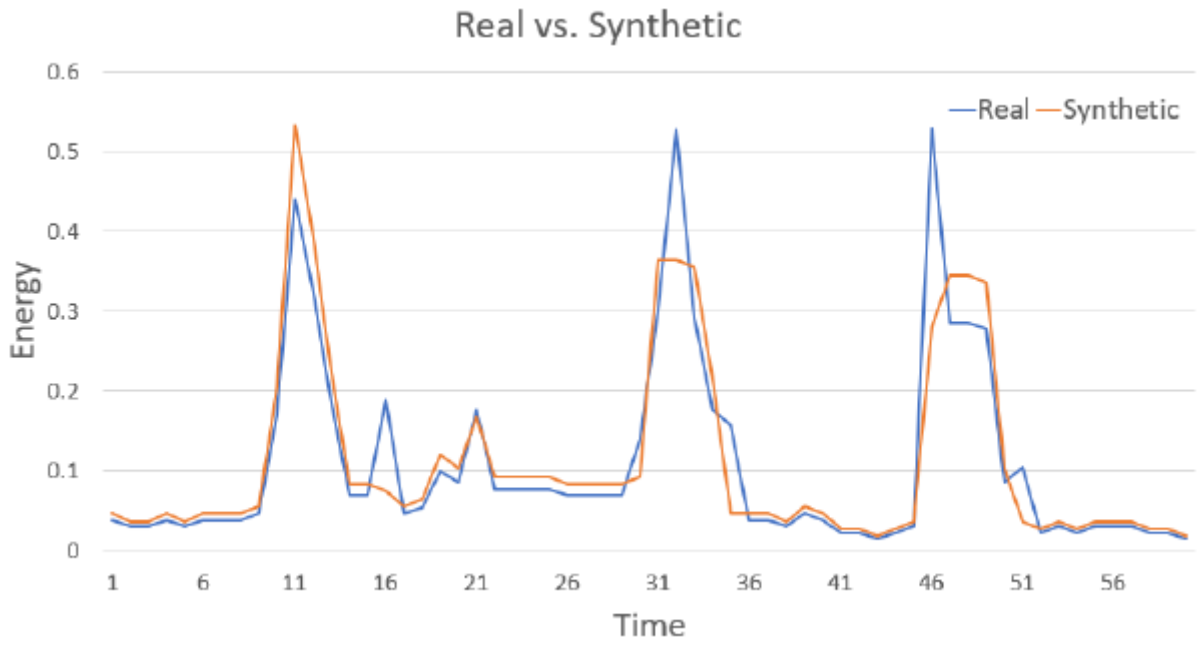
R-GAN: resultados

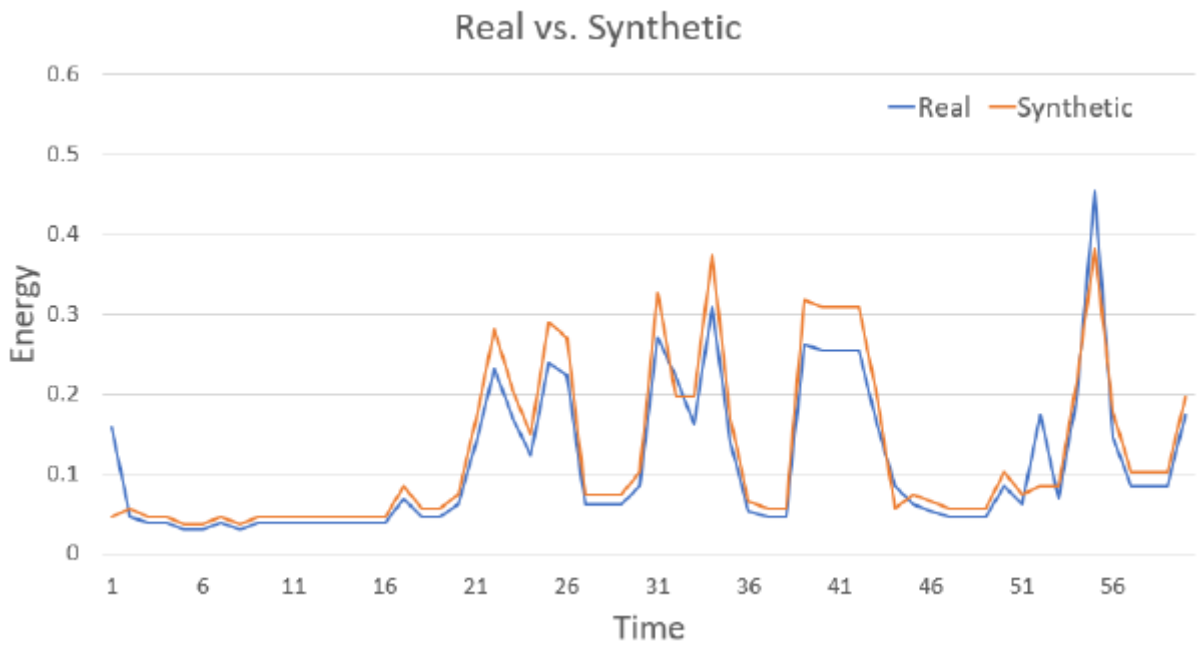
- La precisión de TRTS es mayor que la de TSTR.
- La buena precisión de TRTS muestra que el predictor generaliza a partir de datos reales y que las muestras generadas son similares a los datos reales.
- Sin embargo, los errores TSTR más altos que los errores TRTS indican que el modelo entrenado con los datos generados no captura los datos reales tan bien como el modelo entrenado sobre los datos reales.
- Una posible razón es que, a pesar de utilizar técnicas para reducir el impacto del mode collapse, la variedad de las muestras generadas no es tan alta como la variedad de datos reales.

Features	MAPE (%)				MAE			
	TRTS	TRTR	TSTR	TSTS	TRTS	TRTR	TSTR	TSTS
Core features	13.60%	17.98%	18.67%	18.80%	54.26	63.82	62.74	90.90
Core and ARIMA features	8.65%	11.43%	11.37%	8.92%	48.14	62.67	54.00	80.00
Core and FT features	9.07%	15.84%	17.79%	15.10%	48.99	63.12	61.74	90.67
Core, ARIMA and FT features	5.28%	10.81%	10.12%	6.80%	46.41	62.27	52.54	78.35

R-GAN: resultados

- Ejemplos de dos muestras generadas y datos reales más similares.
- Los patrones generados son similares, pero no iguales a las muestras reales: los datos son realistas, sin ser una mera repetición de los patrones de entrenamiento.





R-GAN: análisis estadísticos de resultados

- Evaluar si existe diferencia estadísticamente significativa entre las distribuciones de datos generados y reales.
- Tests no paramétricos: Kruskal-Wallis y Mann-Whitney (no asumen normalidad).
- Hipótesis nula: las distribuciones de los datos reales y sintéticos son iguales.
- p-value mayor que 0.05, no permite descartar la hipótesis nula: en todos los casos hay poca evidencia de que la distribución de los datos generados sea estadísticamente diferente a la distribución de los datos reales.

Model (Real vs. Synthetic)	Kruskal-Wallis H Test		Mann-Whitney U Test	
	H Value	*p* Value	U Value	*p* Value
Core features	416.50	0.312	0.247	0.619
Core and ARIMA features	428.00	0.375	0.107	0.744
Core and FT features	390.50	0.191	0.775	0.375
Core, ARIMA, and FT features	380.50	0.180	0.885	0.355

R-GAN: análisis estadísticos de resultados

- La idea intuitiva del funcionamiento de R-GAN es similar a la de técnica de sobremuestreo de minorías sintéticas (Synthetic Minority Over-sampling TEchnique, SMOTE) para tratar el problema del desequilibrio de clases.
- SMOTE toma cada muestra de clase minoritaria y crea muestras sintéticas en líneas que conectan cualquiera o todos los k vecinos minoritarios.
- R-GAN está propuesto para regresión y SMOTE en general se usa para clasificación, pero ambos crean nuevas muestras utilizando conocimiento sobre las muestras existentes. SMOTE lo hace creando nuevas muestras entre las reales (muestreo) mientras que R-GAN aprende de los datos reales para generar muestras similares.
- R-GAN es potencialmente útil para problemas de desequilibrio de clases.
- Existen propuestas combinando ambos enfoques.