



UNIVERSIDAD ANDRÉS BELLO
FACULTAD DE INGENIERÍA

INGENIERÍA CIVIL INFORMÁTICA

**Generación de curvas de luz artificiales mediante una
Red Neuronal Generativa Adversaria.**

André Etienne Ducheylard Lolic
Profesor guía: Billy Peralta Márquez

SANTIAGO - CHILE

Enero - 2022

Índice

1. Introducción	2
2. Marco Teórico	3
2.1. <i>Long short-Term Memory - LSTM</i>	3
2.2. <i>Generative Adversarial Network - GAN</i>	5
2.3. Curvas de luz	7
2.4. Foldeo	7
2.5. <i>IAR</i>	8
3. Hipótesis	9
4. Objetivos del trabajo	10
4.1. Objetivos Específicos	11
5. Metodología	12
5.1. Ambiente y experimento	12
5.1.1. Cómputo	12
5.1.2. Ambiente de desarrollo	12
5.1.3. Configuración del modelo de la red <i>GAN</i> y <i>LSTM</i>	12
5.1.4. Propuesta de Generación	16
5.1.5. Configuración de parámetros de entrada	17
5.2. Foldeo	19
5.3. Comparación con <i>IAR</i>	22
6. Resultados	23
6.1. Datasets	23
6.2. Pre-procesamiento de datos	27
6.3. Primer experimento con Generador de una capa	29
6.3.1. OGLE-III	29
6.3.2. GAIA	34
6.3.3. WISE	38
6.4. Segundo experimento con Generador de dos capas	43
6.4.1. OGLE-III	43
6.4.2. GAIA	48
6.4.3. WISE	51
6.5. Comparación <i>IAR</i>	56
6.5.1. Sin forzar	57
6.5.2. Con forzar	73
7. Análisis de datos	91
8. Conclusión y trabajos futuros	94

1. Introducción

El estudio de los cuerpos celestes en el espacio ha entregado enormes cantidades de conocimiento al Ser Humano. Siendo este tipo de estudio el más antiguo de las ciencias naturales, nacen diversos temas religiosos, astrológicos, mitológicos y varias otras prácticas que se aprecian en las épocas prehistóricas. En combinación, los temas abordados todavía están presentes en la actualidad y demuestran la gran capacidad de conocimiento que entregaron en su momento, y que sin duda, sigue siendo una de las áreas más importante a investigar.

Sin duda, cada cuerpo celeste que se estudia entrega distinto conocimiento y formas de entender el cosmos en general. En particular, las estrellas, son sumamente importantes para el estudio de cualquier otro astro, ya que, son la base para la creación de planetas y son un pilar fundamental del espacio. El estudio de estas, conllevó a conocer información básica de los planetas, identificar otros sistemas solares, descubrir nuevas galaxias, e incluso descubrir distintos tipos de estrella que se creían inexistentes o imposibles de formarse. Otro punto importante por destacar de la astronomía, es que, a medida que va pasando el tiempo, la tecnología va mejorando considerablemente, permitiendo así, obtener mejores resultados, construir telescopios y observatorios con mejor equipamiento. Todo esto, con el fin de poder estudiar y analizar cuerpos y estrellas más allá de lo que se permite en el momento.

Dada las dimensiones del espacio, limitaciones tecnológicas y humanas, conocer y entender todo el espacio es prácticamente imposible por el momento, por eso, cada vez que se descubre y se identifica un nuevo cuerpo celeste, el ser Humano se vuelve más cerca de poder responder preguntas triviales, tales como: ¿Que había antes del *Big Bang*? ¿Cómo se formó el Universo? ¿Podría existir vida en otros planetas?. Estas preguntas y muchas más, intentan ser respondidas a través del estudio de las estrellas y otros cuerpos celestes. En particular, para la última pregunta expuesta, en los planetas se busca analizar los gases y elementos presentes junto con aquellos con los que interactúan, buscando principalmente rastros de Carbono, ya que este elemento corresponde a uno de los elementos básicos para la formación de vida. Esto representa un simple ejemplo de las implicancias que tiene el estudio de los cuerpos celestes en general.

Particularmente, este trabajo está enfocado en la luminosidad o brillo aparente de las estrellas variables para entender y mostrar sus periodos. Para ello, se utilizó una red neuronal GAN para poder generar series de tiempo lo más realistas posibles y luego ser foldeadas para obtener su periodo. Se cree que todas o casi todas las estrellas tienen una luminosidad o brillo aparente variable, sea debido al comportamiento natural de los gases que la componen, o producto de una poca visibilidad desde la tierra, de ahí nace el nombre de brillo aparente. Esta última variación suele producirse por polvo estelar que reduce la calidad y la cantidad de luz que logra llegar a la tierra o producto de otros cuerpos que pueden eclipsar la estrella. Para los casos de las estrellas no variables, se habla de situaciones donde su brillo aparente varía en valores muy pequeños o valores que simplemente no se pueden medir actualmente, entregándole el nombre de estrellas no variables.

2. Marco Teórico

2.1. Long short-Term Memory - *LSTM*

Las redes neuronales *LSTM* son similares a las redes neuronales recurrentes (*RNN*), con la diferencia que son capaces memorizar información pasada de corto y largo plazo. Estas redes se utilizan cuando existe una secuencia en los datos de entrada. Por ejemplo, si queremos generar un párrafo de texto, no basta con solo generar letras o palabras, sino, que hay que generar palabras que tengan una conexión o significado entre ellas, y para eso, se necesita tener una memoria de largo plazo, llamada, Estado de la célula. Este componente es equivalente a la memoria del modelo, el cual contiene funciones llamadas *Gates* o compuertas que permiten ir eliminando y agregando distintas información a la memoria, para luego poder generar una serie de palabras que en conjunto tengan un significado y logren formar un párrafo.

Esta memoria mencionada anteriormente, funciona como un canal que está presente durante todo el proceso de la red pasando por cada iteración de secuencia. Además, existe otro canal llamado Estado oculto o memoria de corto plazo, que corresponde a un vector con los datos de entrada de la iteración actual más la salida del estado oculto de la iteración anterior. En la Figura 1, se puede apreciar las distintas entradas y salidas junto con los dos tipos de memoria mencionados anteriormente. La Figura 2, muestra cómo van trabajando las células en conjunto a medida que el modelo va creciendo con una mayor cantidad de células. Finalmente, la Figura 3, muestra la *LSTM* de forma compacta a un alto nivel.

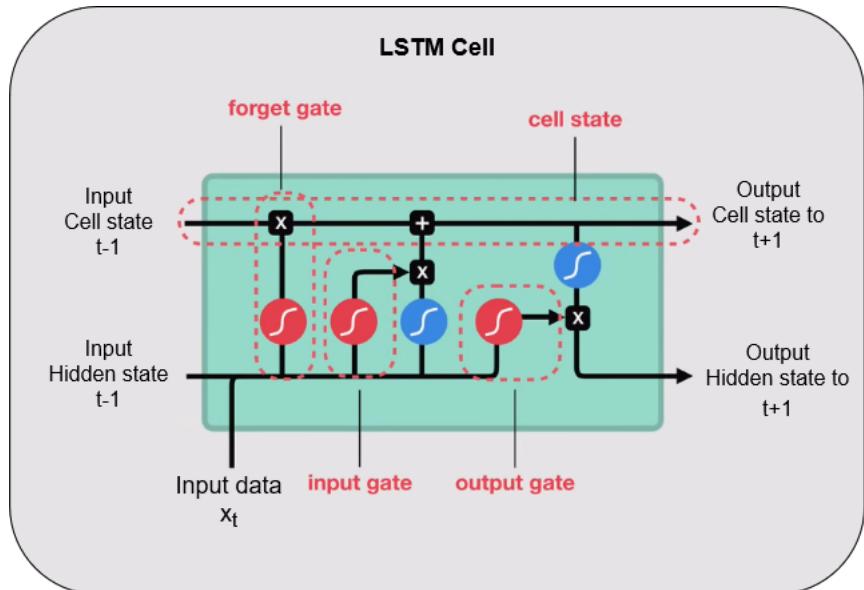


Figura 1: Célula de una *LSTM*

Cada *gate*, al tener una función distinta, tienen también fórmulas distintas. Por ejemplo, la ecuación 1, corresponde a la fórmula de la compuerta de eliminación, que funciona recibiendo un vector compuesto por el estado oculto de la célula anterior, h_{t-1} , más los datos de entrada correspondientes a la iteración actual x_t . Luego, el vector es multiplicado

por sus pesos, W_f , y sumado con su *bias*, b_f .

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$

Siguiendo con el flujo, el vector pasa por la compuerta de entrada, la cual, primero utiliza una función *sigma* para conocer qué datos serán actualizados, y luego, se usa la función *tanh* sobre el vector anterior multiplicado por sus pesos correspondientes, W_C , la cual crea un vector con los valores candidatos a actualizar la memoria. Dichas funciones corresponden a las ecuaciones 2 y 3 respectivamente.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (3)$$

Después de conocer los valores que se van a eliminar de la memoria, y cuáles se van a agregar, se prosigue a actualizar la memoria. Para ello, se realiza la operación del producto vectorial entre los datos a eliminar y la memoria anterior, $f_t \otimes C_{t-1}$, sumados con la información que se va a agregar, $i_t \otimes \tilde{C}_t$, generando así, la memoria o *Cell State* de la iteración actual, C_t , descrita por la siguiente ecuación.

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \quad (4)$$

Finalmente, para decidir que *output* obtener de la célula, se debe conocer qué información extraer y luego así, obtenerla desde la memoria. Entonces, el vector de entrada, compuesto por el estado de la célula anterior más los datos de entrada, $[h_{t-1}, x_t]$, es multiplicado nuevamente por su peso W_o , para ser sumado con su *bias* para luego ser procesados con la función *sigma*, y así obtener los datos que se quieren recuperar desde la memoria, siendo denotado por o_t . Luego, queda utilizar la función *tanh* sobre la memoria para transformar los datos a valores de -1 y 1, y así, realizando una multiplicación vectorial con o_t , se logra extraer la información filtrada de la memoria, la que se utilizará para la siguiente célula como estado oculto.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \otimes C_t \quad (6)$$

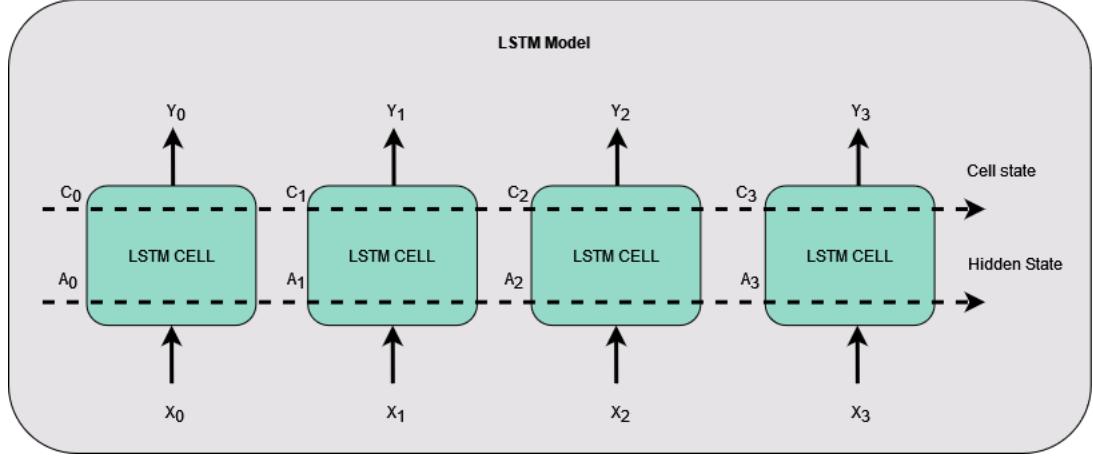


Figura 2: Modelo de una red *LSTM* extendida.

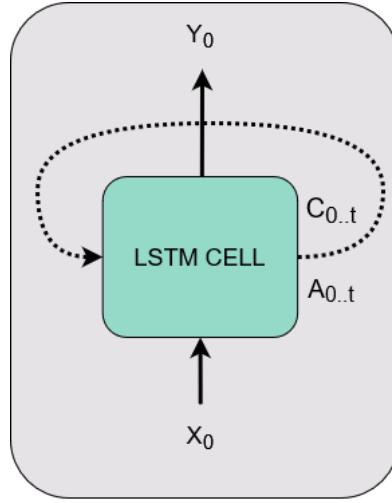


Figura 3: Modelo de una red *LSTM* compacta.

2.2. *Generative Adversarial Network - GAN*

Las redes neuronales *GAN* están construidas bajo dos pilares principales o submodelos. Uno de ellos es un modelo que actúa como Discriminador, cuya tarea dada por su nombre, es poder discriminar o discernir entre tipos de datos reales y falsos. Este modelo analiza los datos de entrada y los compara con datos reales, para poder decidir si finalmente los datos de entrada corresponden a datos ficticios o no. El segundo pilar por considerar de las redes *GAN*, es el modelo del Generador, el cuál es el modelo encargado de generar datos falsos con el objetivo de que estos datos sean lo más reales posibles, o en términos simples, tratar de generar datos que sean lo más reales posibles para engañar al Discriminador. Este tipo de redes tienen el nombre de *Generative Adversarial Networks* dada la competencia y sinergia que tienen sus pilares. Esto se debe a que, por un lado, existe un modelo que intenta generar datos lo más reales posibles, mientras que hay un segundo modelo que va clasificando los datos creados, creando así, un feedback mutuo entre los modelos. La Figura 4 muestra el funcionamiento de la red *GAN* a un alto nivel.

Uno de las principales dificultades de utilizar este modelo, es debido al entrenamiento, ya que necesita que los dos pilares se vayan entrenando mutuamente y tengan un ritmo

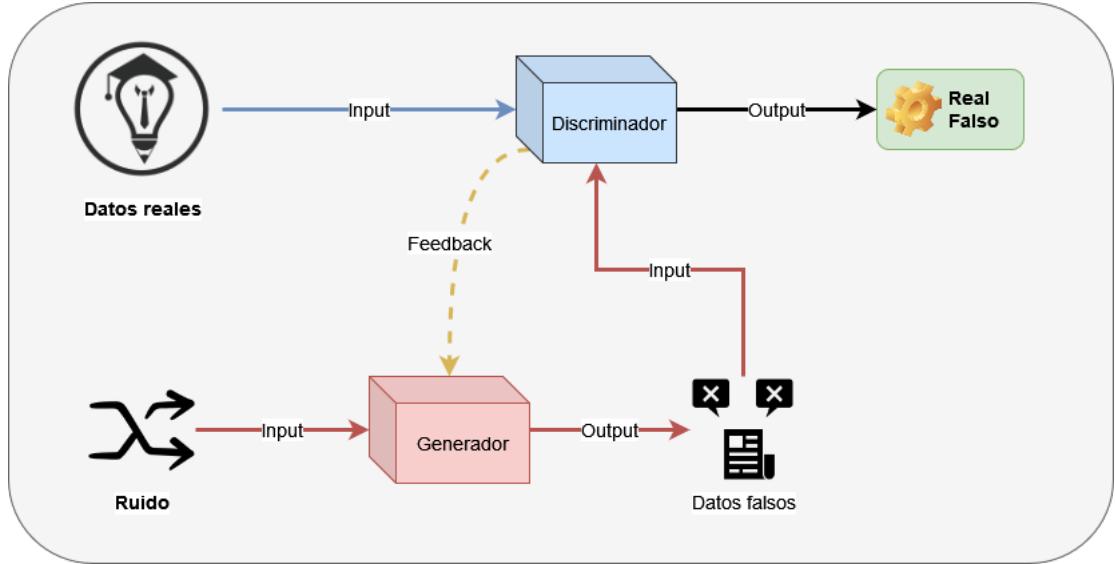


Figura 4: Funcionamiento de la red *GAN*. [1]

similar. Basta con sobre-entrenar un modelo o con que uno se aprendan de memoria los datos de entrenamiento para que este modelo no entregue resultados satisfactorios. Además, por la naturaleza de este modelo, los datos no suelen converger, por ende, se debe realizar un riguroso proceso al momento de definir los parámetros de entrenamiento siendo acompañado de muchas pruebas y errores para encontrar el punto de equilibrio de este modelo.

La ecuación 7, describe la expresión matemática del modelo *GAN*. La cual está compuesta por el modelo Discriminador, que busca maximizar el resultado final, y por el modelo del Generador, el cual lo busca minimizar. Es aquí, donde se representa la rivalidad entre los modelos, lo que permite poder dividir la fórmula en dos partes que deben ser sumadas. La primera, corresponde a la parte izquierda, la cual dice que se debe calcular el logaritmo de las predicciones del Discriminador dado un set de datos reales. Es decir, se debe aplicar el logaritmo sobre el promedio de predicciones de datos reales, en este caso, series de tiempo reales. En la segunda parte de la ecuación, omitiendo de momento el logaritmo y la resta al interior de ella, teniendo así, $D(G(z))$, se representa la predicción del Discriminador sobre series provenientes del Generador, es decir, series de tiempo falsas. Ya que el Discriminador busca maximizar el resultado de la ecuación, sería lógico que se maximizara también en ambas partes, sin embargo, cuando el Discriminador estima $G(z)$, este espera que el valor fuese el menor posible, ya que, se quiere que el Discriminador detecte la menor cantidad de series generadas como reales. Esto pone en desequilibrio la ecuación, y para equilibrarla, se introduce la resta que se dejó de lado anteriormente, la cual, invierte el resultado de estimación del Discriminador sobre los datos generados, permitiéndole ahora maximizar la diferencia entre 1 y su estimación de datos reales sobre datos provenientes del Generador. Si observamos la ecuación desde el punto de vista del Generador, se puede apreciar que en la primera parte, va a querer minimizar el resultado del Discriminador, es decir, minimizar la cantidad de series catalogadas como reales. Mientras que en la segunda parte, va a tender a minimizar el complemento de las predicciones del Discriminador sobre datos falsos, es decir, quiere que el Discriminador catalogue como series reales, la mayor cantidad de series que provienen

del Generador.

$$\min_G \max_D V(D, G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (7)$$

2.3. Curvas de luz

Las curvas de luz representan la luminosidad de un cuerpo celeste o una región del espacio en función del tiempo. Estas se miden en frecuencias y comúnmente se describen por distintas bandas, tales como, Banda B, la cual filtra la luz azul o la Banda V, que filtra la luz verde. Cada una de ellas, funciona en un cierto intervalo de frecuencias, lo que permite filtrar ciertos colores de la luz. Comúnmente, estudiar la luminosidad de los cuerpos celestes permite conocer ciertas propiedades únicas o características que con ciertos telescopios no pueden ser observados, sea por la cantidad de ruido en la región del espacio o por el tamaño y distancia del cuerpo. Dado que la luminosidad está modelada en función del tiempo, se puede detectar por ejemplo, los períodos de luminosidad, amplitudes y cambios en su brillo. Se puede ver el caso de las clases **Cepheid**, las cuales logran pulsar de una forma bien rápida, teniendo un periodo y amplitud muy definidos y estables. Dada que esta clase tiene una correlación muy fuerte entre su periodo y su luminosidad, se suelen usar como referencia para poder escalar y transformar distancias en el espacio. Dentro de esta clase, se pueden definir subclases, las cuales difieren según sus tamaños, masas, edades, gases y otro factores que permiten catalogarlas de distintas formas. Por ejemplo, las clases **Classical Cepheids** y **Type II Cepheids** pertenecen a la clase **Cepheid**. Sin entrar a profundizar cada una, existen varias formas de clasificar las estrellas y de sub-clasificarlas según sean sus características y el nivel de detalle que se quiera tener, siendo uno de ellos, una clasificación automatizada y supervisada [2]. En las siguientes figuras, 5 y 6, podemos ver las curvas de una estrella de la clase **Cepheid** y **Type II Cepheids** respectivamente.

Uno de los dilemas con algunas curvas de luz, es que pueden contener mucho ruido, dificultando su análisis y la recopilación de otras características, causando que puedan ser catalogadas erróneamente en otra categoría. Por eso, se suelen aplicar y construir otros modelos y formas para volver a catalogar y verificar los cuerpos celestes. [5] Es común, que cuando se grafican las curvas de luz, no se puedan visualizar ni destacar ciertos patrones en la gráfica, sea por, tener mucho ruido u otras fuentes del luz que fueron captadas igual, polvo estelar, e incluso, los mismos campos gravitacionales de otros cuerpos pueden afectar las curvas de luz. Es por esto, que estimar datos, variables o características de las curvas suele ser un proceso difícil y complejo, que requieren de modelos matemáticos y procesamiento de máquinas para obtener resultados con mayor precisión.

2.4. Foldeo

El foldeo consiste en obtener una muestra o firma representativa del periodo de luminosidad de una clase de estrella en particular. A grandes rasgos, debido a que las series de tiempo representan datos que están ligados a un tiempo en particular, el foldeo logra captar el tiempo en común que tengan las mediciones y trata de buscar un patrón en ellas. Este patrón, corresponde al ciclo donde la magnitud de la estrella varía en cierto

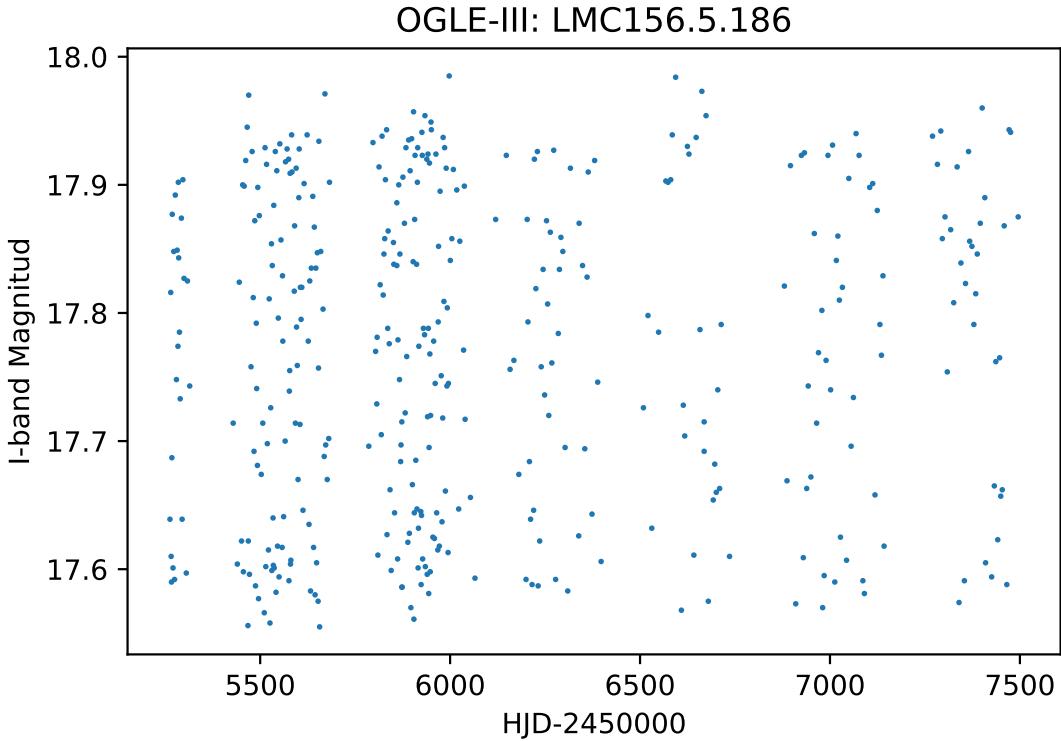


Figura 5: Fotometría de una serie de luz perteneciente a la clase **Classical Cepheid**. [3]

intervalo, y así, permite obtener la cantidad de días, meses, años, o más bien, el periodo o ciclo donde la magnitud de la estrella varía.

La figura 7 demuestra el foldeo sobre la curva de luz 5 mencionada anteriormente, perteneciente a la clase **Classical Cepheid** del dataset de **OGLE-III**. Aquí, se puede apreciar que el periodo de dicha estrella equivale 12 días aproximadamente, es decir, la magnitud máxima y mínima que va a entregar la estrella, será dentro de un intervalo de 12 días.

Este foldeo, fue realizado con una librería en particular, donde se ve claramente que no se puede describir un patrón en foldeo, produciendo la inseguridad de saber si el periodo estimado es el correcto o no. En la siguiente figura, se puede apreciar el foldeo correcto para la curva de luz mencionada anteriormente, la cual es obtenida desde el mismo repositorio de **OGLE** teniendo un periodo aproximado de 0.30681810 días. [3]

2.5. *IAR*

IAR, [6] corresponde a una librería la cual contiene distintos modelos y *scripts* para ser implementados sobre series de tiempo irregulares, de ahí viene su nombre *Irregular Autoregressive*. Esta librería se utilizó para generar series de tiempo, al igual que el modelo GAN, y así poder calcular un *score* con la LSTM Score y analizar cual tiene mejor rendimiento.

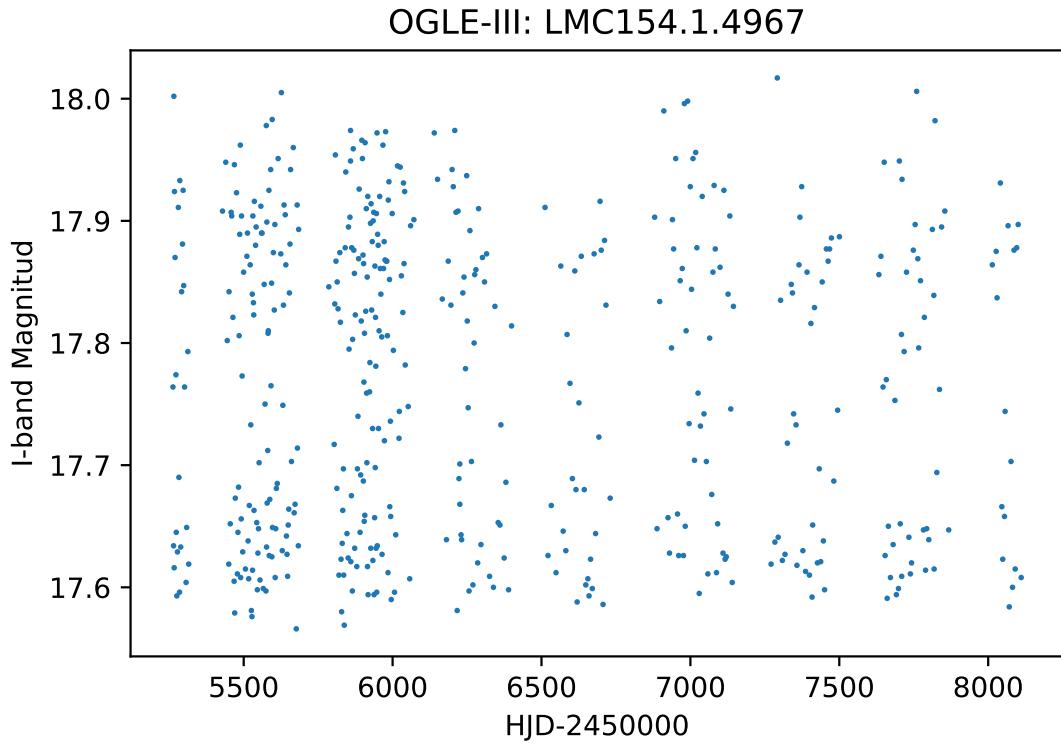


Figura 6: Fotometría de una serie de luz perteneciente a la clase **Type II Cepheid**. [4]

3. Hipótesis

Mediante una red Neuronal Adversaria Generativa, o *GAN* por sus siglas en inglés, se van a generar series de tiempo lo más reales posible a las series originales, para luego, foldear una muestra y analizar si el foldeo resulta similar al foldeo de la clase a la cual la serie de tiempo pertenece. Adicionalmente, se usará un modelo de *Long Short-Term Memory*, o *LSTM*, para calcular el *accuracy* de las series generadas versus las series reales, donde se espera que el resultado debiese tender los 0.5 o 50 %, representando así, una existencia igual en cantidad de series de tiempo generadas y reales.

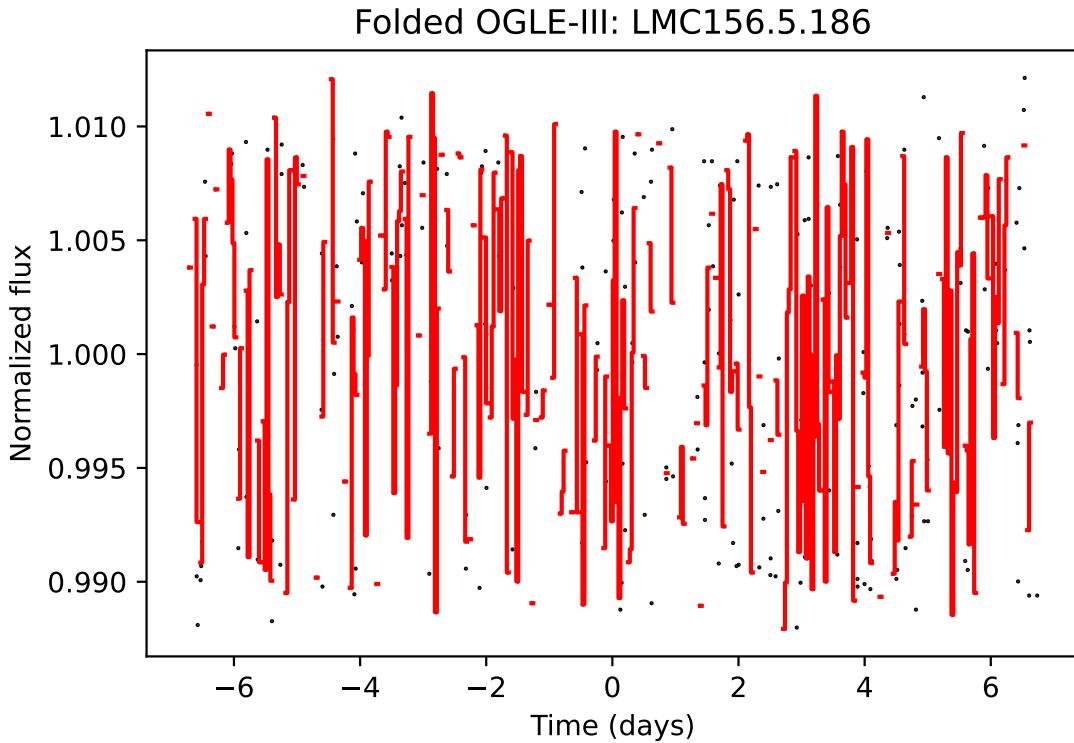


Figura 7: Foldeo estimado sobre una curva de luz de la clase ***Classical Cepheid*** del dataset de ***OGLE-III***.

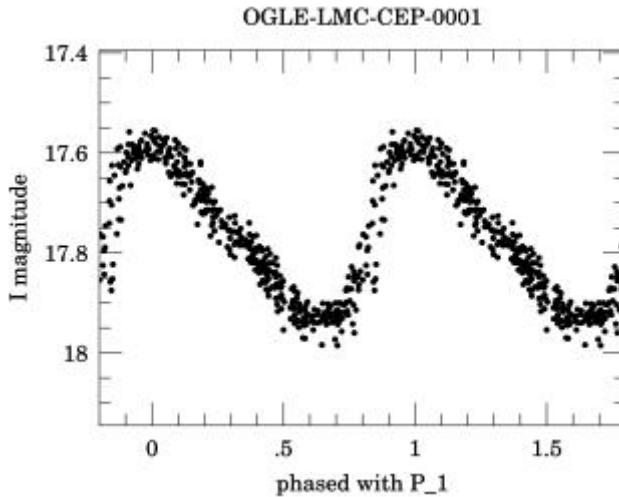


Figura 8: Foldeo sobre una curva de luz de la clase ***Classical Cepheid*** del dataset de ***OGLE-III*** ajustado con un periodo de 0,30681810 días.

4. Objetivos del trabajo

Analizar y generar distintas fotometrías de estrellas variables provenientes de tres datasets utilizando redes neuronales *LSTM* y de generación adversaria (*GAN*) con el fin de ser foldeadas y comparar si son similares a los foldeos reales.

4.1. Objetivos Específicos

Construir un modelo Generativo Adversarial para la generación de series de tiempo.

Construir un modelo para calcular el *accuracy* de las series reales versus generadas por la *GAN*.

Utilizar librería de *Lightkurve* para estimar los periodos de las fotometrías y foldearlas.

Implementar un modelo auto regresivo de series irregulares (*iAR*), para generar series de datos.

Analizar y comparar los resultados de *accuracy* entre el modelo *GAN* e *iAR* mediante una métrica.

5. Metodología

5.1. Ambiente y experimento

5.1.1. Cómputo

El trabajo se desarrolló en una misma máquina local desde un inicio, ya que, gracias a la tarjeta de video la máquina presente ya tenía los recursos de cómputo necesarios para poder realizar las actividades correspondientes. Esta máquina consta de las siguientes características donde no se le realizó ningún overclocking durante el trabajo:

- Procesador Intel i7-10700k.
- Tarjeta de video NVIDIA GTX 1060 de 6GB.
- 2 x 16GB RAM 3200 MHz.

5.1.2. Ambiente de desarrollo

Originalmente, se inició programando en el software de *Visual Studio Code*, pero, dada la naturaleza del trabajo y la necesidad de poder tener una flexibilidad al momento de analizar distintos resultados, se optó por cambiarse a programar con *Anaconda* y *Google Collab*. A menos de la mitad del progreso, dada ciertas actualizaciones de librerías, Anaconda dejó de funcionar correctamente debido a conflictos de versiones. Por lo tanto, la gran parte del desarrollo terminó siendo programado en *Google Collab* mediante un entorno local. Este entorno local de *Jupyter* fue ejecutado desde una *CMD* con el siguiente código:

Código 1: Código para ejecutar el entorno Local.

```
1 jupyter notebook —NotebookApp.allow_origin='https://colab.research.google.com' —port=8888 —NotebookApp.port_retries=0 —NotebookApp.token=abcd
```

5.1.3. Configuración del modelo de la red *GAN* y *LSTM*

Anteriormente, se mencionó que se realizaron experimentos por cada dataset. Ahora, acotando de mayor forma los experimentos, cada clase de cada dataset se ejecutó dos veces. Esto se debe a que se usaron dos formatos del modelo Generador, uno donde el generador presenta solamente una capa oculta, y otro, donde presenta dos capas ocultas. La imagen número 9 muestra los experimentos realizados que a continuación serán detallados. El modelo Discriminador de la red *GAN*, fue construido con una capa oculta de 128 nodos con una función de activación 'tanh' y una segunda capa correspondiente a la de salida con una activación *sigmoid* para obtener valores entre 0 y 1. Inicialmente, el modelo Generador fue construido de la misma forma que el modelo Discriminador, teniendo las misma configuración de la capa oculta y de salida. Mientras que la segunda iteración del trabajo, consistió en agregar una segunda capa oculta de 64 nodos con la misma función de activación. Ambos modelos fueron entrenados con un ratio de aprendizaje de 0,001, donde finalmente fueron combinados en un solo modelo que es el cual

llamamos *GAN*, denotado en el código número 2. Los resultados de los experimentos y de las distintas iteraciones serán mostradas en secciones posteriores a esta.

Dentro del mismo código, se puede apreciar el modelo de *LSTM*, que sigue una estructura bien similar a la red *GAN*, teniendo la misma función de activación. Sin embargo, esta difiere en la cantidad de capas ocultas y nodos de la capa de salida, teniendo dos y una respectivamente. Este modelo analiza la serie de tiempo completa, es decir, utiliza tanto el tiempo como la magnitud para entregar un *score* definiendo la similitud entre series reales y falsas. Esta red de *LSTM* es llamada *LSTM_Score*, la que se encarga de evaluar las series de tiempo generadas, utilizando un 50 % de series de tiempo reales y el otro 50 % con series de tiempo generadas por la *GAN* para su entrenamiento. Entonces, luego de que los modelos sean entrenados, se le pasan series de tiempo generadas (sabiendo que son falsas) junto a unas series reales como *input*. Esto produce que el modelo *LSTM_Score*, asumiendo que está bien entrenado, teóricamente entregue un *accuracy* con tendencia a 0,5. Podemos suponer que se logra generar una serie de tiempo que aparenta ser completamente real, pero, que sabemos que es falsa, y concatenarla con una cantidad igual de series reales y utilizada como input al modelo. Este debiese entregar valores entre 0,5 y 1 teóricamente, ya que de base, la mitad de las series entregadas eran reales, por ende, podemos decir que el *accuracy* debiese estar definido en un valor mínimo de 0,5. Pero, el valor del final de *accuracy*, dependerá de cómo son clasificadas el resto de series generadas, ya que, si la mitad de ellas son clasificadas como series falsas, el modelo entregaría un *accuracy* final de 0,75. En cambio, si la totalidad de series generadas son catalogadas como series reales, lógicamente el *accuracy* final debiese ser 1. En la práctica, se vio que no siempre los valores estarán en un intervalo de 0,5 y 1, debido a que el modelo está lejos de ser perfecto, y las series generadas no son tan perfectas a las reales como se esperaría. En la sección de resultados, se verá el *accuracy* de las variables *x_test* e *y_test* que se tiene por cada clase, donde se puede observar que algunas de las instancias tienen un resultado cercano al ideal, y casos donde el resultado es completamente lejano del esperado.

A continuación, en la figura 10 se puede apreciar un ejemplo del *score* y *loss* del modelo *LSTM* sobre el dataset de OGLE-III.

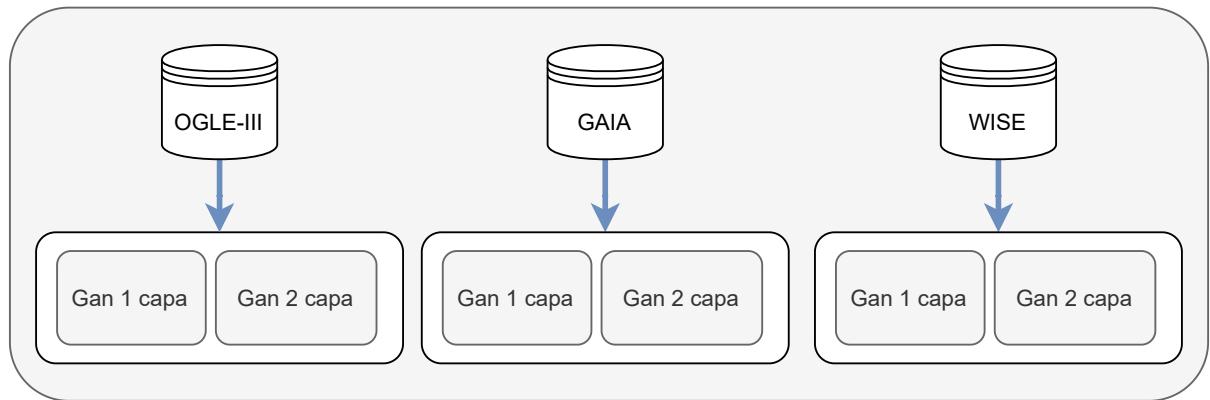
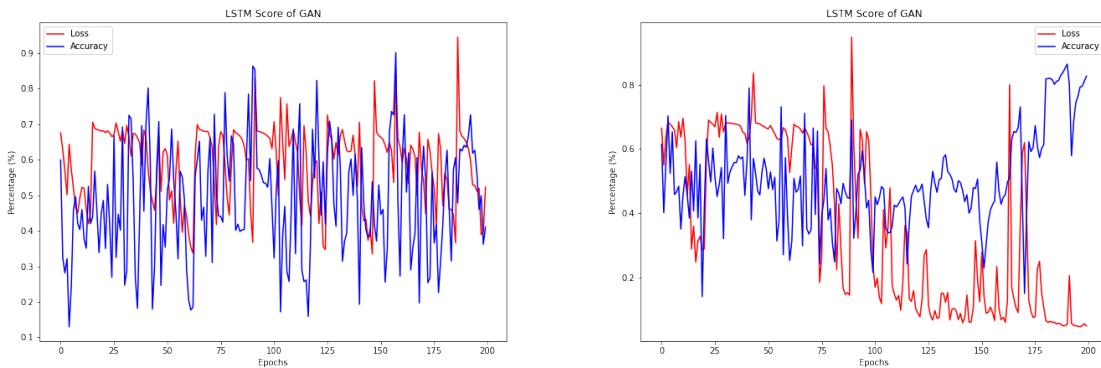


Figura 9: Orden de experimentos, donde por cada dataset disponible se ejecutaron dos experimentos, siendo el primero con un Generador de una capa, y el segundo con uno de dos capas.



(a) OGLE-19_11_2021-12_47_21-std con *accuracy* de 51.34 %
(b) OGLE-18_11_2021-15_58_39-cep con un *accuracy* de 83.25 %

Figura 10: Ejemplos del *accuracy* y *loss* del modelo LSTM Score de los experimentos donde el Generador tiene una capa. La primera gráfica representa un resultado ideal mientras que la segunda no entrega resultados satisfactorios.

```

1 #MODELOS GAN
2 def define_discriminator(n_input, cantidad_mediciones):
3     opt = tf.keras.optimizers.Adam(learning_rate=0.001)
4     model = Sequential(name='Discriminador')
5     model.add(LSTM(128, activation='tanh', batch_size=n_input,
6         input_shape=(cantidad_mediciones, 1), return_sequences=True))
6     model.add(Dense(1, activation='sigmoid'))
7     model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['
8         accuracy'])
9     model.summary()
10    return model
11
11 def define_generator(n_input, cantidad_mediciones):
12     opt = tf.keras.optimizers.Adam(learning_rate=0.001)
13     model = Sequential(name='Generador')
14     model.add(LSTM(128, activation='tanh', batch_size=n_input,
15         input_shape=(cantidad_mediciones, 2), return_sequences=True))
15     model.add(LSTM(64, activation='tanh', return_sequences=True))
16     model.add(Dense(1, activation='sigmoid'))
17     model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['
18         accuracy'])
19     model.summary()
20    return model
21
21 def define_gan(g_model, d_model):
22     opt = tf.keras.optimizers.Adam(learning_rate=0.001)
23     #No entrenamos los pesos del discriminador
24     d_model.trainable = False
25     #Por bug al guardar los modelos the keras, debemos sobreescibir
26     #que si queremos entrenar los pesos del generador.
27     g_model.trainable = True
28     model = Sequential(name='GAN')
29     model.add(g_model)
30     model.add(d_model)
31     model.compile(loss='binary_crossentropy', optimizer=opt)
```

```

31     model.summary()
32     return model
33
34 #MODELO LSTM PARA EL SCORE DE LA GAN
35 def define_gan_score(cantidad_mediciones, len_vec_lstm_score):
36     lstm_score_model = Sequential(name='LSTM_Score')
37     lstm_score_model.add(LSTM(128, activation='tanh', input_shape=(
38         cantidad_mediciones, len_vec_lstm_score), return_sequences=True))
39     lstm_score_model.add(Dense(2, activation='sigmoid'))
40     lstm_score_model.compile(loss='binary_crossentropy', optimizer='
41         adam', metrics=['accuracy'])
41     lstm_score_model.summary()
41     return lstm_score_model

```

Código 2: Parámetros del modelo de la red neuronal *GAN* con un generador de dos capas ocultas y la LSTM Score para evaluar los datos generados luego del entrenamiento de la red *GAN*.

5.1.4. Propuesta de Generación

En general, el experimento consistió en entrenar el modelo de la red *GAN* con las series de tiempo reales provenientes de los datasets de **OGLE-III**, **GAIA Y WISE**. [7] El método de selección de las series, fue eligiendo, un número determinado de series según su clase, en orden, a partir de la primera serie del dataset. De modo de ejemplo, si una clase requiere tener 1000 series de tiempo, se seleccionaron las primeras 1000 series provenientes del dataset. Dado a que existe más de un dataset disponible, se realizaron tres grandes experimentos para separar los resultados de cada uno. Sin embargo, el procedimiento fue equitativo para todos los dataset y sin modificar sus procesos. El modelo Discriminador, recibe como *input* los valores de la magnitud normalizadas, mientras que, el Generador, se le deben realizar diversas operaciones de manera previa. Estas operaciones son las siguientes; Primero, se calcula la diferencia de los tiempos de las mediciones. En segundo lugar, se normalizan dichas diferencias. Para luego, sumar los resultados y concatenarlos con datos generados aleatoriamente -o ruido-. Finalmente, son enviados como entrada o *input* al modelo Generador para seguir con el proceso de entrenamiento. Mientras se entrena el modelo Generador, cada cierta cantidad de iteraciones o *Epochs*, se generan varias series de tiempo para poder foldear una muestra y ser almacenada para un posterior análisis. Este análisis consiste en comparar visualmente la gráfica obtenida del foldeo con el foldeo real de la clase que se esta analizando, el cual permite conocer el rendimiento que tiene el modelo *GAN* durante su entrenamiento. El flujo de este proceso, está denotada por la figura número 11.

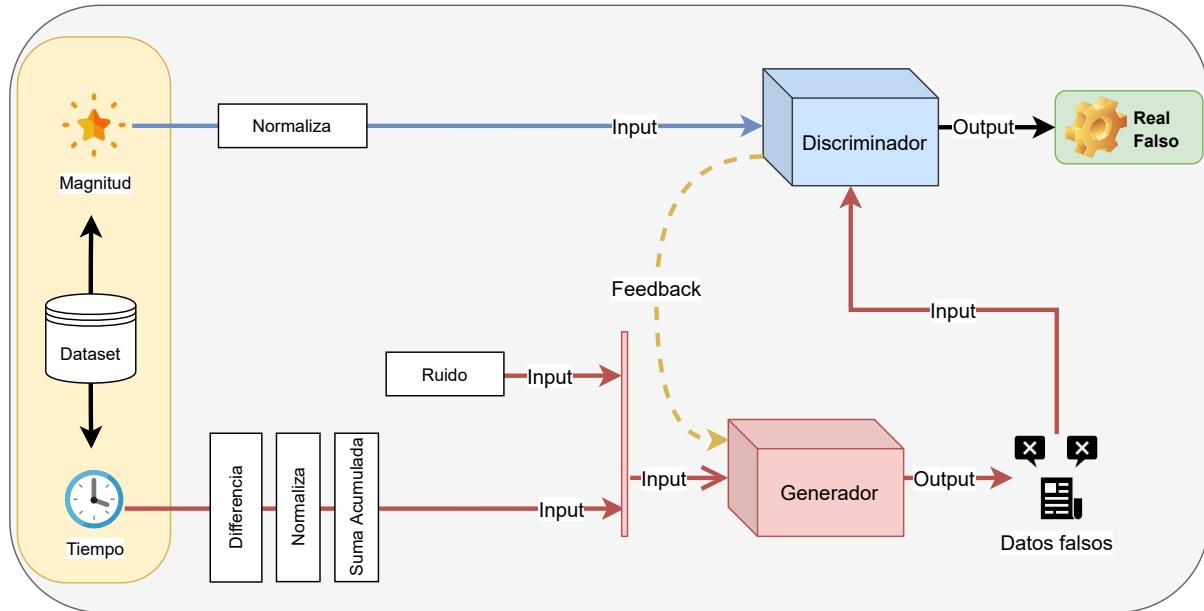


Figura 11: Funcionamiento general del experimento.

Luego de que la red *GAN* es entrenada y se generan magnitudes, se concatenan con datos de tiempo bootstrapeados. El bootstrapping corresponde a un proceso de seleccionar una muestra aleatoria de datos, en este caso, seleccionar una cierta cantidad de series de tiempo. Para este trabajo, se eligió un número entre cero y la cantidad máxima de series generadas por la *GAN*, para luego extraer de la totalidad de series reales, la serie en dicha posición del numero aleatorio. Esta selección es iterada según la cantidad de series

de tiempo que se generaron. Por ejemplo, si el modelo *GAN* fue entrenado para generar 2000 series de tiempo, se elige al azar un número entre 0 y 2000, la cual corresponderá a la serie real que se debe extraer. Este proceso de selección es el que se ejecuta 2000 veces para extraer dicha cantidad de series. Ahora, de las series extraídas, se les elimina las magnitudes, ya que, la idea es concatenar las magnitudes generadas por la *GAN* con los tiempos de estas muestras. Para efectos de la *LSTM Score*, este tiempo también es procesado de la misma manera que el tiempo utilizado al condicionar la red *GAN*, es decir, pasa por los procesos de diferencia, normalización y suma acumulada.

Teniendo ya una lista con las series reales utilizadas, y las magnitudes concatenadas con sus tiempos bootstrapeados como series generadas, se concatenan también con dos vectores los cuales contienen valores de unos y ceros, representando las series reales con valores de uno, y las falsas con 0. Estos son utilizados en la función de *train_test_split* de la librería de **scikitlearn** [8], la cual divide los datos en variables de entrenamiento y de evaluación. Para dicho proceso, se configuró la función con un *test_size* de 0,2, y un *random_state* igual a 5. Esto con el fin de eliminar variables que dificulten la replicación de los experimentos. Finalmente, este modelo entrega un *accuracy* sobre los datos entregados como entrada, el cual representa la similitud entre series reales y falsas.

5.1.5. Configuración de parámetros de entrada

El código desarrollado logra entrenar una red *GAN* a partir de una clase en particular, para así, intentar generar series de tiempo lo más reales a la clase utilizada como input.

Antes de poder ejecutar el código se debe configurar ciertos parámetros de entrada, tales como, las rutas de archivos, el nombre de la clase con la que se va a trabajar, la cantidad de series de tiempo y cantidad de mediciones que tenga la clase. Esto está definido y documentado a continuación.

```

1 #PARAMETROS DE LOS MODELOS COMPARTIDOS
2 training = True #Si queremos entrenar los modelos (True) o cargarlos
                 desde una carpeta (False)
3 cantidad_series_de_tiempo = 100 #Cantidad de series de tiempo con las
                                 que vamos a trabajar del dataset.
4 cantidad_mediciones = 30 #Cantidad de mediciones que con la que vamos a
                           entrenar el modelo.
5 path_datos = ".\DATOSWISE\LCS" #Carpeta donde estan todas las series
                                 de tiempo
6 path_index = ".\DATOSWISE\WISE_dataset.dat" #Archivo donde se enlistan
                                                informacion de las series de tiempo, es como el index. ID CLASE PATH
                                                P_1 errorT1 N.
7 extension_series_de_tiempo = '.dat'
8 path_models_root = ".\modelos" #Carpeta raiz donde guardaremos los
                                 modelos
9 path_models = ".\modelos\\03_11_2021-00_24_52-EC\modelos\\" #Carpeta de
                                 donde cargamos los modelos. SOLO SI {training} = False
10 columnas_index = ['ID', 'Class', 'Path', 'N'] #Agregar columnas que
                                                 sean necesarias.
11 nombre_clase = "RRd" #Nombre de la clase a cargar de las series de
                        tiempo
12
13
14 #PARAMETROS DEL MODELO GAN
15 epochs = 1000 #Cantidad de epochs para el entranamiento de la GAN

```

```

16 batch_size = 10 #Tamano del batch con la cual se entrena los datos,
17     una mitad son datos reales y la otra falsos.
18 sample_interval = 5 #Intervalo para analizar o evaluar el training de
19     los modelos mientras se estan entrenando.
20 len_vec = 1 #Cantidad de atributos de las mediciones (este caso solo
21     genero la magnitud)
22 n_input = None #Input de dimension que recibe el modelo (Dejo en 'None'
23     para que sea generico)
24
25 #PARAMETROS EXCLUSIVOS LSTM_SCORE
26 epochs_score = 200 #Cantidad de epochs para el entranamiento de la LSTM
27     para el score de la GAN(aprox 400)
28 len_vec_lstm_score = 2 #Analizamos dos elementos el tiempo y la
29     magnitud.
30 n_samples_evaluate = int(0.5 * cantidad_series_de_tiempo) #Cantidad de
31     muestras generadas que se usaran para medir el score de la GAN
32     mediante la LSTM.

```

Código 3: Código con parámetros de configuración. Se puede ver que está dividido en 3 partes, la primera es para configurar parámetros que se comparten a lo largo de los modelos y otras funciones del código. La segunda parte se utiliza para configurar el entrenamiento y configuración básica de la red *GAN*. Y finalmente, la tercera parte trabaja con el modelo de *LSTM* que se utiliza para evaluar la generación de datos de la *GAN* luego de que haya terminado de entrenar.

Existen ciertos parámetros de los modelos que se mantuvieron constantes a lo largos del entrenamiento e iteraciones posteriores. Esta lista es la siguiente:

- *training*: Esta variable booleana representa la opción de entrenar los modelos *GAN* y *LSTM* o cargarlos un modelo ya entrenado.
- *epochs*: Representa la cantidad de ciclos o iteraciones con la que se va a entrenar el modelo de la *GAN*.
- *batch_size*: El tamaño con que los modelos extraen una muestra de datos para poder entrenar en cada iteración.
- *sample_interval*: Intervalo para imprimir el estado del entrenamiento de la *GAN*, junto con el foldeo correspondiente al epoch en el cual se está entrenando.
- *len_vec*: Cantidad de atributos que la red *GAN* va a generar.
- *n_input*: Las dimensiones de los datos de entrada utilizados a los modelos.
- *epochs_score*: Cantidad de *epochs*/iteraciones con la que se va a entrenar el modelo de la *LSTM* que calcula el *score* de la red *GAN*.
- *len_vec_lstm_score*: Largo de vectores que se utiliza en la red *LSTM* para calcular el *score* de la *GAN*. En este caso, se evalúa la combinación del tiempo y de la magnitud de las series reales y generadas.

Además, se debe destacar que al momento de entrenar el modelo de *LSTM_Score* se definió la variable ***random_state*** igual a 5 y el valor del *test size* a 0,2. Esto es para

poder reducir la cantidad de variables aleatorias que influyen los resultados, permitiendo entrenar el modelo en base a la misma semilla y con una cantidad porcentual fija de series de evaluación.

5.2. Foldeo

Cada clase de estrella analizada tiene un periodo de luminosidad único. Esto representa el ciclo donde la estrella tendrá momentos de mayor y menor luminosidad que otros. Para conocer dicho periodo, comúnmente se utilizan funciones matemáticas para poder detectar en que secciones del tiempo la data se repite, formando así, un ciclo. Estos ciclos, y la cantidad que existan, van a determinar la clase a la cual pertenece la serie analizada. Por ejemplo, la figura número 12 corresponde a un ejemplo de una estrella foldeada del dataset de GAIA perteneciente a la clase **CEP**.

En los experimentos, para realizar el foldeo y estimar los periodos y ciclos de las series de tiempo, se utilizó la librería de *Lightkurve*[9], la cual, permite obtener las gráficas representadas en la figura número 12.

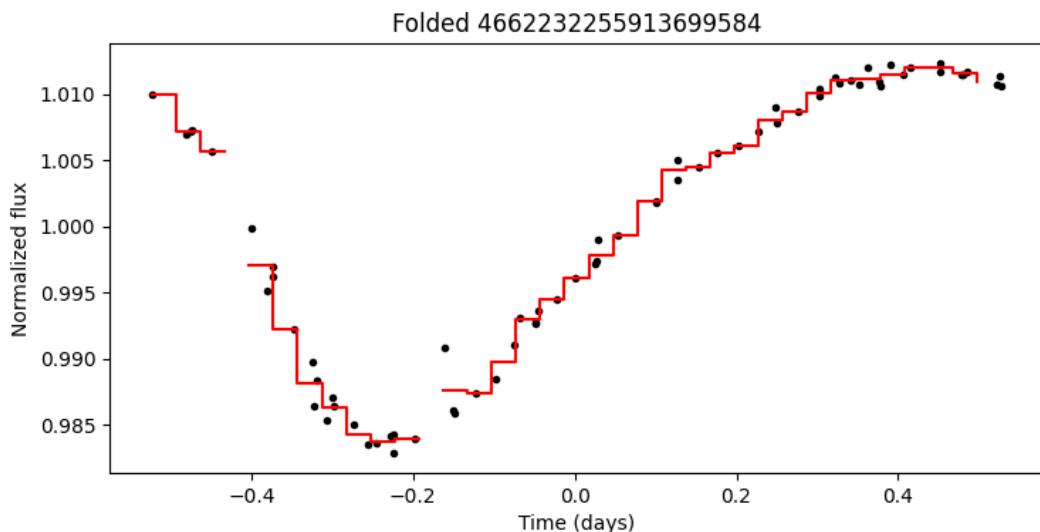


Figura 12: Ejemplo de foldeo sobre una estrella del dataset GAIA perteneciente a la clase CEP.

El foldeo está descrito en el siguiente código número 4.

```

1 #PROCESAMOS LOS DATOS GENERADOS PARA FOLDEARLOS
2
3 #Desnormalizamos la magnitud generada
4 generated_mag_sin_norm_2dim = scaler_mag_normal.inverse_transform(
    generated_mag_2dim.T).T
5 #Cambiamos la dimension
6 generated_mag_sin_norm_3dim = generated_mag_sin_norm_2dim.reshape((
    cantidad_series_de_tiempo, cantidad_mediciones, 1))
7 #Juntamos el tiempo bootstrapeado con la magnitud generada
8 time_series_generada_3dim = np.vstack((samplesBoot_tiempo_sin_norm_3dim
    .T, generated_mag_sin_norm_3dim.T)).T
9
10 #Nombre de columnas de la instancia del archivo de OGLE

```

```

11 nombre_columnas = ['time', 'magnitud', 'error']
12 #Rellenamos la columna 'error' de la libreria para foldear (no podemos
13     #foldear sin esta columna).
14 test_0 = np.zeros((cantidad_series_de_tiempo, cantidad_mediciones, 1))
15 #Concatenamos la columna de 0's
16 time_series_generada_3dim = np.vstack((time_series_generada_3dim.T,
17                                         test_0.T)).T
18 #Pasamos la serie de tiempo al df y utilizamos solo 1 serie de tiempo
19     #para ver el foldeo
20 df_generado = pd.DataFrame(time_series_generada_3dim[20,:,:], columns=
21                             nombre_columnas)
22 if 'WISE' not in ruta_instancia:
23     #Sumamos un escalar al tiempo para un foldeo correcto
24     df_generado['time'] = df_generado['time'] + 2450000
25 #print(df_generado)
26
27 df_generado_dtype_float = df_generado.astype(float) #Creamos una copia
28     #del {df_generado} como flotante.
29 #print(df_generado_dtype_float)
30
31 if training:
32     df_generado_dtype_float.to_csv(path_or_buf=ruta_instancia + '\\\\' +
33                                     'SerieTiempo-Generada-Post-Training.csv', sep=' ', index=False)
34
35 #Convertimos el df a un LightCurve
36 lc_generado = lk.LightCurve(time=df_generado_dtype_float['time'], flux=
37                             df_generado_dtype_float['magnitud'], flux_err=
38                             df_generado_dtype_float['error'])
39 #print(lc_generado)
40
41 #Calculamos el periodograma usando el metodo de'lombscargle'
42 pg_generado = lc_generado.to_periodogram(method='lombscargle',
43                                            oversample_factor=10)
44
45 #PLOTEAMOS EL PERIODOGRAFO
46 periodograma = pg_generado.plot()
47 periodograma.set_title("Periodograma generado")
48 if training:
49     periodograma.figure.savefig(ruta_instancia + '\\\\' + 'Periodograma-
50                                 -Generado-Post-Training.png')
51
52 #INFO
53 pg_generado.show_properties()
54 print("Frequency at max power:", pg_generado.frequency_at_max_power)
55 print("Periodo:", pg_generado.period_at_max_power)
56 lista_resultados.append(pg_generado.period_at_max_power)
57
58 #FOLDEO
59 #Todavia no puedo foldear con la fase normalizada en la misma funcion
60     #ya que elimina la unidad de 'JD' y no he logrado solucionarlo. Usar
61     #dias como unidad tampoco funciona.
62 lc_generado_foldeado = lc_generado.fold(pg_generado.period_at_max_power
63                                         , normalize_phase=False)
64 #print(lc_generado_foldeado)

```

```

54 lc_scatter = lc_generado_foldeado.scatter(color='red')
55 lc_scatter.set_title("Serie de tiempo generada")
56 if training:
57     lc_scatter.figure.savefig(ruta_instancia + '\\\\' + 'SerieTiempo-
      Generada-Scatter-Post-Training.png')
58
59 #Normalizamos las magnitudes
60 lc_generado_normalizado = lc_generado.normalize()
61 #print(lc_generado_normalizado)
62
63 #Intento de samplear y normalizar el foldeo
64 median_generada = st.median(lc_generado_foldeado['flux'])
65 mean_generada = st.mean(lc_generado_foldeado['flux'].value)
66 stddev_generada = st.stdev(lc_generado_foldeado['flux'].value)
67
68 lc_generado_foldeado['flux_norm'] = lc_generado_foldeado['flux'] /
      mean_generada
69
70 lc_generado_foldeado_binned = lc_generado_foldeado.bin(0.03)
71 #print(lc_generado_foldeado_binned)
72
73 plt.clf()
74 plt.plot(lc_generado_foldeado.time.jd, lc_generado_foldeado['flux_norm'],
      'k.', markersize=1)
75 plt.plot(lc_generado_foldeado_binned.time_bin_start.jd,
      lc_generado_foldeado_binned['flux_norm'], 'r-', drawstyle='steps-mid')
76 plt.xlabel('Time (days)')
77 plt.ylabel('Normalized flux')
78 plt.title("Foldeo - Generado")
79 if training:
80     plt.savefig(ruta_instancia + '\\\\' + 'Foldeo-Generado-Post-Training.
      png')
81 plt.show()
82 plt.pause(0.0001)

```

Código 4: Código utilizado para el Foldeo de estrellas.

Particularmente, concatenamos las magnitudes generadas por la *GAN* y el tiempo boottapeado para generar series de tiempo falsas. Luego, para poder utilizar la librería de foldeo, se necesitan tres columnas, que corresponden al tiempo, magnitud y error de magnitud respectivamente, por esto, a la serie falsa se le concatena una columna de 0's para poder representar su error. Debido a la cantidad de series generadas, sería tedioso foldear y analizar todas las series de tiempo, por ello, se elige una al azar para realizar las operaciones pertinentes (En este caso se trabajó siempre con la serie de tiempo en la posición 20). A continuación, la serie de tiempo es convertida a un objeto de *LightCurve* [10] para poder utilizar sus funciones y métodos específicos a la librería. Esto permite generar un periodograma, que corresponde a la transformación de *Fourier* para poder detectar su periodo estimado, momento pic o de mayor luminosidad de la estrella y conocer si ocurre de manera cíclica o no. Siguiendo con el proceso, se utiliza la función *fold* con el periodo obtenido desde el periodograma para realizar un foldeo aproximado sobre la serie de tiempo. El resultado de dicha operación se puede ver en la figura 12 usada anteriormente como ejemplo.

5.3. Comparación con *IAR*

La librería de *iAR* se utiliza para generar modelos autoregresivos con períodos irregulares, como es el caso de las estrellas en estudio. *iAR* se implementó para poder comparar el *score* de la red *GAN*. Dado que este modelo no permite ser entrenado con varias series de tiempo a la vez, se tuvo que iterar una a una las series de tiempo reales y generar a su vez cada serie de tiempo falsa. Este proceso se realizó hasta tener la cantidad de series de tiempo deseada, que tiene que ser igual al valor de series de tiempo que se utilizaron para entrenar la red *GAN*. El entrenamiento y generación de datos por *iAR* fue ejecutado siguiendo las instancias generadas por la *GAN*, donde por cada clase existente, se entrenó el *iAR* y se generaron sus series de tiempo correspondientes.

En el proceso, se encontraron series de tiempo reales que tenían valores repetidos, como por ejemplo, dos mediciones con el mismo tiempo o varias magnitudes con los mismo valores. Esto genera un error en el transcurso del entrenamiento causando que las series no puedan ser generadas correctamente, por lo que se optó por saltar estas series y continuar con la siguiente. Dada los resultados de las primeras iteraciones de *iAR*, se observó que no siempre lograba generar series de forma correcta, sino que, en base a una comparación visual y promedios de las magnitudes entre las series reales y generadas, se notó casi que la mitad de las series eran generadas con una media de magnitud cercana a 0, mientras que las series reales tenían un promedio de magnitudes cercanas a 15. Para intentar comparar los experimentos de la *GAN* con resultados ideales, se decidió ejecutar nuevamente el *iAR*, con la diferencia de que esta segunda iteración, se le forzó a generar series que tuvieran un promedio de magnitud mayor a 1. Esto produjo, que todas las series generadas por *iAR*, fueran visualmente bien similares a las series reales y sus magnitudes fueran cercanas al de las series reales.

Por último, para analizar los resultados de ambos modelos, se definió una métrica a partir del *accuracy* obtenido del modelo *LSTM Score*, llamada ***AccuracyMetric***, el cual, describe, la similitud entre los datos generados y los reales. Dicha fórmula es utilizada para mostrar los resultados en la sección Comparación *IAR*, y esta denotada a continuación:

$$\text{AccuracyMetric} = |\text{accuracy}(x) - 0,5| \quad (8)$$

El valor de la métrica será igual a la diferencia entre el *accuracy* obtenido por el modelo de *LSTM_Score* de una clase X, y 0,5. Siendo el mejor resultado 0, y el peor 0,5, donde 0 representa una completa similitud entre series, mientras que el 0,5, representa una completa diferencia entre ellas.

6. Resultados

A continuación, se van a mostrar los resultados obtenidos de los experimentos en sus respectivas tablas, separados por cada dataset y por experimento. La columna **Nombre**, corresponde al nombre de la instancia, siendo la fecha y hora en la cual fueron creados, concatenada con la clase a la cual pertenece. Los nombres fueron acortados para una mejor visualización de las tablas, mostrando solamente el segundo en el que fue creada la instancia junto al nombre de la clase. El **Foldeo Real** corresponde al foldeo de una serie de tiempo de la clase analizada, mientras que el **Foldeo Generado**, corresponde a un foldeo de una serie generada por la *GAN* luego de ser entrenada. Para los casos donde la serie de tiempo no logró ser generada correctamente, se optó por elegir el mejor foldeo de los datos que fueron generados al momento de realizar el entrenamiento de la red *GAN*. Por esto, al observar la primera instancia del dataset de OGLE-III del experimento de una capa, se puede apreciar que el título del gráfico del foldeo generado es *Foldeo Generado - 980 - 2*. El primer número, 980, corresponde al epoch del entrenamiento de la red *GAN*, de la cual se generaron y foldearon los datos. El segundo número, 2, corresponde al número de serie de tiempo que se eligió para foldear, ya que, para todos los experimentos, se evaluaron las series generadas 0, 2, 4 y 6 del total de las series, donde se buscó visualmente la más similar al foldeo real.

También, existen algunos casos donde la gráfica aparenta tener dos firmas, dos dibujos o dos patrones. Esto se debe a que existen ciertas clases que pueden llegar a tener más de un periodo, lo que causa que en la gráfica se puedan ver más de un ciclo o formas en que los datos se agrupen. Un ejemplo del escenario mencionado es la instancia **52-ESD**, del dataset de OGLE-III del experimento con una capa, donde se aprecian dos líneas de agrupaciones de datos de color negro, y su foldeo de color rojo, el cual intenta promediar o calcular un solo periodo en base a las dos agrupaciones de datos que tiene. Adicionalmente, esta misma instancia no presentó tener un foldeo luego de haber sido entrenada, ya que, los resultados que generó la *GAN* no fueron lo suficientemente reales”, haciendo que las operaciones del foldeo no lograran obtener una firma o un periodo bien definido. Es por esto, que se utilizó un foldeo del entrenamiento para mostrar que el foldeo de las series generadas no se asemeja al real.

6.1. Datasets

En este trabajo, se utilizaron los datasets de **OGLE-III**, **GAIA**, y **WISE** [7], donde la cantidad de series de tiempo y cantidad de clases fue constante durante todo el desarrollo y están definidas en la tabla número 1 que viene a continuación.

Tabla 1: Cantidad total de series de tiempo y cantidad de clases por datasets.

Dataset	Cantidad de series de tiempo	Cantidad de clases
OGLE-III	393.103	11
WISE	123.239	15
GAIA	363.142	9

Por otra parte, para poder estandarizar los experimentos y resultados se optó por filtrar por clase una cantidad de series de tiempo que entregaran resultados demostrativos. Esto está descrito en la tabla 2.

Tabla 2: Cantidad de series de tiempo según la clase y dataset.

Dataset	Clase	Cantidad de series de tiempo
OGLE	ED	21503
	ESD	9475
	EC	6862
	cep	7836
	dsct	2822
	RRab	25943
	RRc	7990
	std	34815
	OSARG	234932
	SRV	34835
GAIA	Mira	6090
	RRAB	162244
	MIRA_SR	150215
	RRC	32331
	RRD	834
	DSCT_SXPHE	8870
	CEP	6488
	T2CEP	1740
	ARRD	106
	ACEP	314
WISE	NonVar	32795
	OSARG	53890
	SRV	8605
	RRc	3831
	T2CEP	482
	RRab	16412
	DSCT_SXPHE	1098
	NC	2237
	CEP	1884
	ELL	243
	Mira	1396
	RRd	241

	ACEP	53
	C	68
	ARRD	4

Un desafío de trabajar con tantas clases de distintos dataset, fue el de poder encontrar una combinación de series de tiempo y cantidad de mediciones que fueran capaces de lograr que la red neuronal *GAN* pudiera aprender y obtener resultados viables para su debido análisis. Para buscar esta combinación, se intentó mantener siempre una cantidad elevada de mediciones dentro de las series de tiempo intentando tener entre 2000 y 5000 series de tiempo -aunque no en todas las clases fue posible-, independiente del origen del dataset o clase. Esta cantidad de mediciones, se iba disminuyendo en función de la cantidad de series de tiempo en total, ya que, existen casos donde al elegir una cantidad de mediciones muy elevada, disminuía considerablemente la cantidad de series en particular, afectando así los datos disponibles para entrenar la *GAN*. Finalmente, el objetivo de esta etapa es poder variar la cantidad de mediciones de cada clase para obtener una cantidad ideal de series de tiempo mencionada anteriormente. La tabla 3 muestra la combinación obtenida.

Tabla 3: Combinación de series de tiempo y cantidad de mediciones según la clase y dataset seleccionadas previo al entrenamiento con la red *GAN*.

Dataset	Clase	# S.T	# Mediciones
OGLE	ED	4249	700
	ESD	4470	500
	EC	5551	700
	cep	4236	700
	dsct	2753	400
	RRab	7245	800
	RRc	5062	400
	std	7990	300
	OSARG	5579	1700
	SRV	5361	1300
GAIA	Mira	4320	500
	RRAB	1434	70
	MIRA_SR	1012	70
	RRC	1556	50
	RRD	294	30
	DSCT_SXPHE	2097	30
	CEP	2007	30

	T2CEP	621	30
	ARRD	92	20
	ACEP	74	30
	NonVar	8831	100
	OSARG	2929	140
	SRV	3118	100
	RRc	1824	30
WISE	T2CEP	266	30
	RRab	1627	60
	DSCT_SXPHE	246	40
	NC	1525	60
	CEP	841	60
	ELL	221	40
	Mira	1049	60
	RRd	120	30
	ACEP	33	30
	C	63	30
	ARRD	2	30

Teniendo ya la cantidad de series de tiempo por clase, se definió una cantidad arbitraria de series de tiempo a utilizar. Aproximadamente la mitad de las clases trabajadas (considerando los tres datasets utilizados) tuvieron cerca de dos mil series de tiempo para el entrenamiento. Los valores específicos están denotadas en la tabla 4. Mirando la tabla, se puede apreciar que las clases **ARRD** y **ACEP** del dataset **GAIA** junto con las clases **ACEP**, **C** Y **ARRD** del dataset **WISE** tienen 0 series de tiempo utilizadas. Esto es debido a que estas clases presentaban una baja cantidad de series de tiempo, por ende, fueron descartadas ya que la red *GAN* no lograba entrenar ni generar datos a partir de dichas clases

Tabla 4: Combinación de series de tiempo y cantidad de mediciones según la clase y dataset seleccionadas para el entrenamiento de la red neuronal *GAN*.

Dataset	Clase	# S.T	# Mediciones
	ED	2000	700
	ESD	2000	500
	EC	2000	700
	cep	2000	700
	dsct	2000	400
OGLE	RRab	2000	800

	RRc	2000	400
	std	2000	300
	OSARG	2000	1700
	SRV	2000	1300
	Mira	2000	500
GAIA	RRAB	1000	70
	MIRA_SR	1000	70
	RRC	1000	50
	RRD	200	30
	DSCT_SXPHE	2000	30
	CEP	2000	30
	T2CEP	600	30
	ARRD	0	20
	ACEP	0	30
	NonVar	2000	100
WISE	OSARG	2000	140
	SRV	2000	100
	RRc	1000	30
	T2CEP	200	30
	RRab	1600	60
	DSCT_SXPHE	200	40
	NC	1000	60
	CEP	800	60
	ELL	200	40
	Mira	1000	60
	RRd	100	30
	ACEP	0	30
	C	0	30
	ARRD	0	30

6.2. Pre-procesamiento de datos

Debido al formato de algunas fotometrías de los dataset, se tuvo que modificar los datos para hacerlos compatibles con el código. Principalmente, en el dataset de **OGLE**-

III, se tuvo que eliminar ciertas identaciones o espacios a las variables cuando el tiempo era menor a mil. Además, existían algunas series de tiempo que de forma aleatoria, o sin saber la razón alguna, venían también con identaciones al comienzo de los datos. Estas dificultades fueron arregladas con el siguiente código número 5.

```

1 import numpy as np
2 import pandas as pd
3 from os import listdir, path, mkdir
4 import os.path
5 from os.path import isfile, join
6
7 path = ".\\testing\\original" #Carpeta donde estan todas las series de
     tiempo que queremos modificar o procesar.
8 nuevo_path = ".\\testing\\new\\\" #Termina con doble '\\' para que no
     elimine la doble comilla al final.
9
10 #Lista de nombre de TODOS los archivos (Si hay mas archivos como un ..
     txt tambien los va a guardar). HAY QUE ASEGURARSE QUE SOLO HAYA
     FOTOMETRIAS AQUI.
11 lista_archivos = [f for f in listdir(path) if isfile(join(path, f))]
12
13 #Recorremos cada archivo detectado.
14 for nombre in lista_archivos:
15     #Lista con las lineas procesadas de la serie de tiempo.
16     lineas_procesadas = []
17
18     ruta = path + '\\' + nombre
19     #Leemos la serie de tiempo
20     with open(ruta) as f:
21         #Obtenemos todas las lineas de la serie de tiempo.
22         lineas = f.readlines()
23         #Iteramos por cada linea.
24         for linea in lineas:
25             new_line = linea.replace(' ', ' ')
26             new_line2 = new_line.replace(' ', ' ')
27             if new_line2.startswith(' '):
28                 lineas_procesadas.append(new_line2[1:-1])
29             else:
30                 lineas_procesadas.append(new_line2[:-1])
31     #Luego de leer todas las lineas de la serie de tiempo...
32     nueva_ruta = nuevo_path + nombre
33     #Creamos la carpeta que tendra todas las series de tiempo
     procesadas, solo si no existe.
34     if not os.path.exists(nuevo_path):
35         os.makedirs(nuevo_path)
36     #Ahora guardamos en un archivo todas las series que vayamos
     procesando.
37     with open(nueva_ruta, 'w') as f:
38         for linea in lineas_procesadas:
39             f.write("%s\n" % linea)
40 print("Datos procesados!")

```

Código 5: Código en lenguaje Python utilizado para el pre-procesamiento de datos.

Complementando el trabajo en general, también se aplicó el siguiente código número 6 para poder obtener a priori la lista de clases según el index del dataset recibido como

entrada. Aunque, esta lista no es un pre-requisito para poder trabajar con los datasets, si facilita ciertos procesos para evaluar el funcionamiento correcto del código.

```
1 import numpy as np
2 import pandas as pd
3 from os import listdir, path, mkdir
4 import os.path
5 from os.path import isfile, join
6
7 #path_index = ".\Archivos_profe\DATOSOGLE\query.txt" #Archivo donde se
     enlistan informacion de las series de tiempo. ID CLASE PATH P_1
     errorT1 N.
8 path_index = ".\Archivos_profe\DATOSOGLE\OGLE_dataset.dat" #Archivo
     donde se enlistan informacion de las series de tiempo. ID CLASE PATH
     P_1 errorT1 N.
9 columnas_index = ['ID', 'Class', 'Path', 'N'] #Agregar columnas que
     sean necesarias.
10
11 #print(path_index)
12 #print(columnas_index)
13
14 df_index = pd.read_csv(path_index, sep=',', skiprows=1, names=
     columnas_index)
15 #print(df_index)
16 lista_clases = df_index['Class'].unique().tolist()
17 print(lista_clases)
```

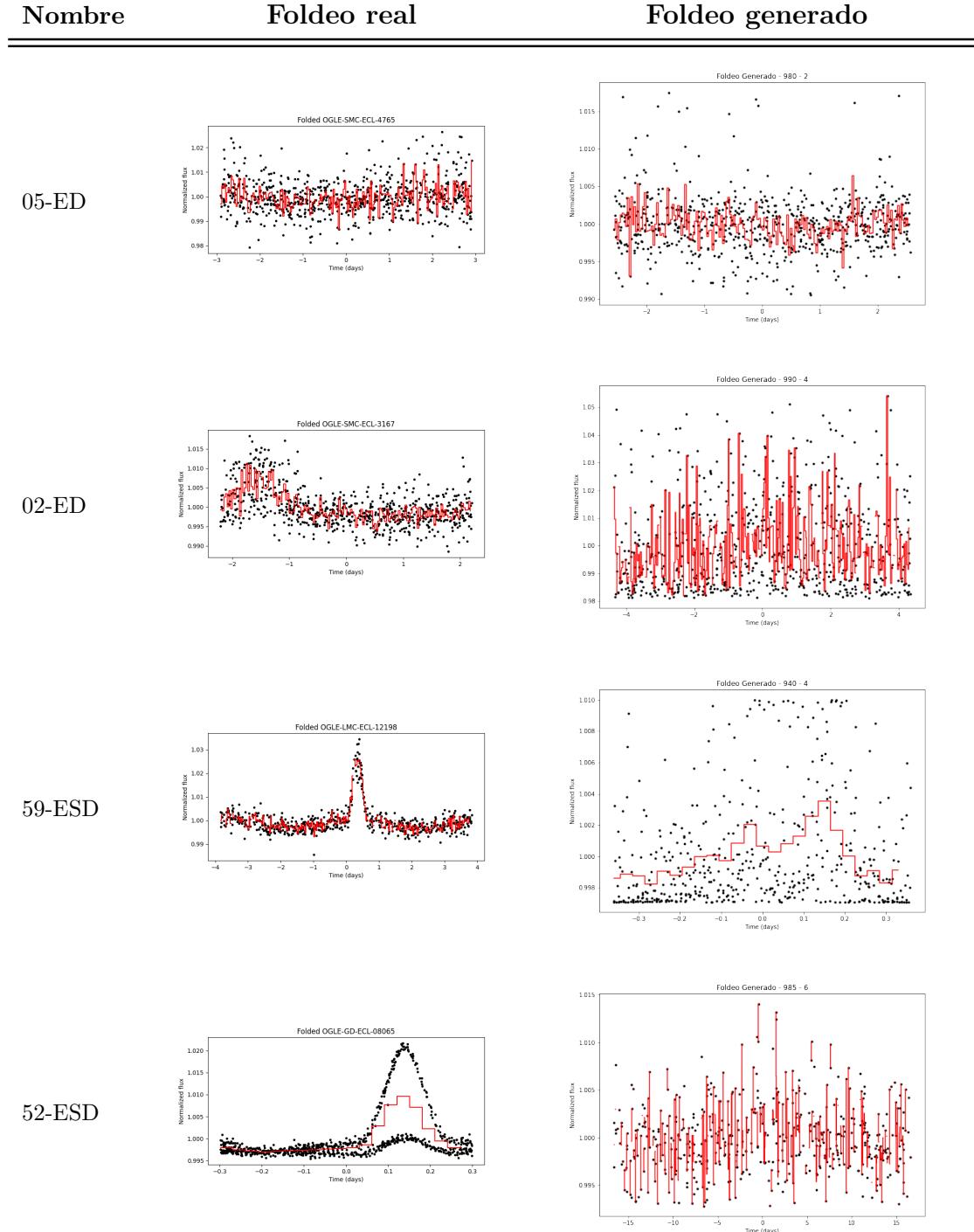
Código 6: Código en lenguaje Python utilizado para obtener la lista de clases que se encuentra en el dataset en cuestión.

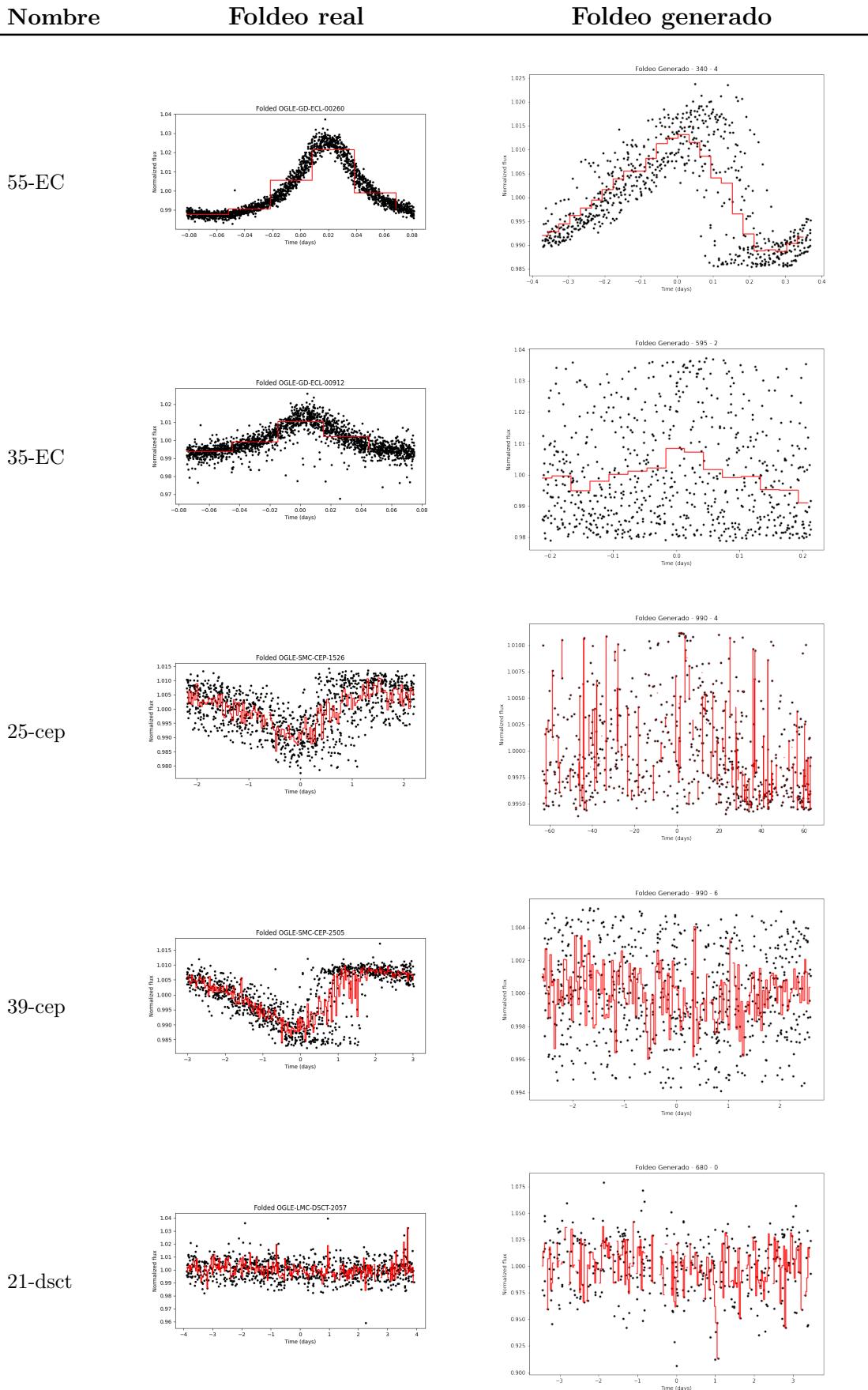
6.3. Primer experimento con Generador de una capa

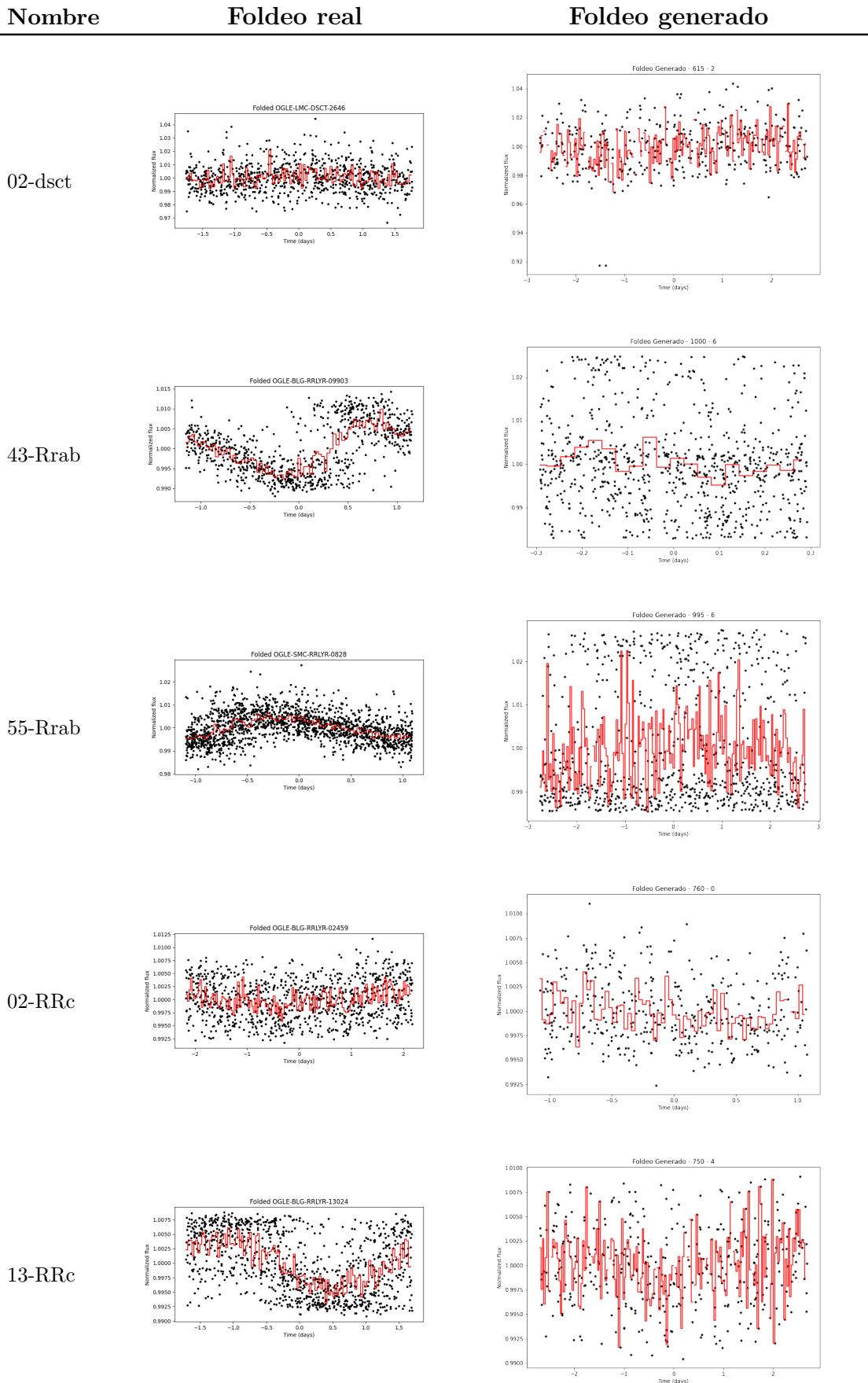
6.3.1. OGLE-III

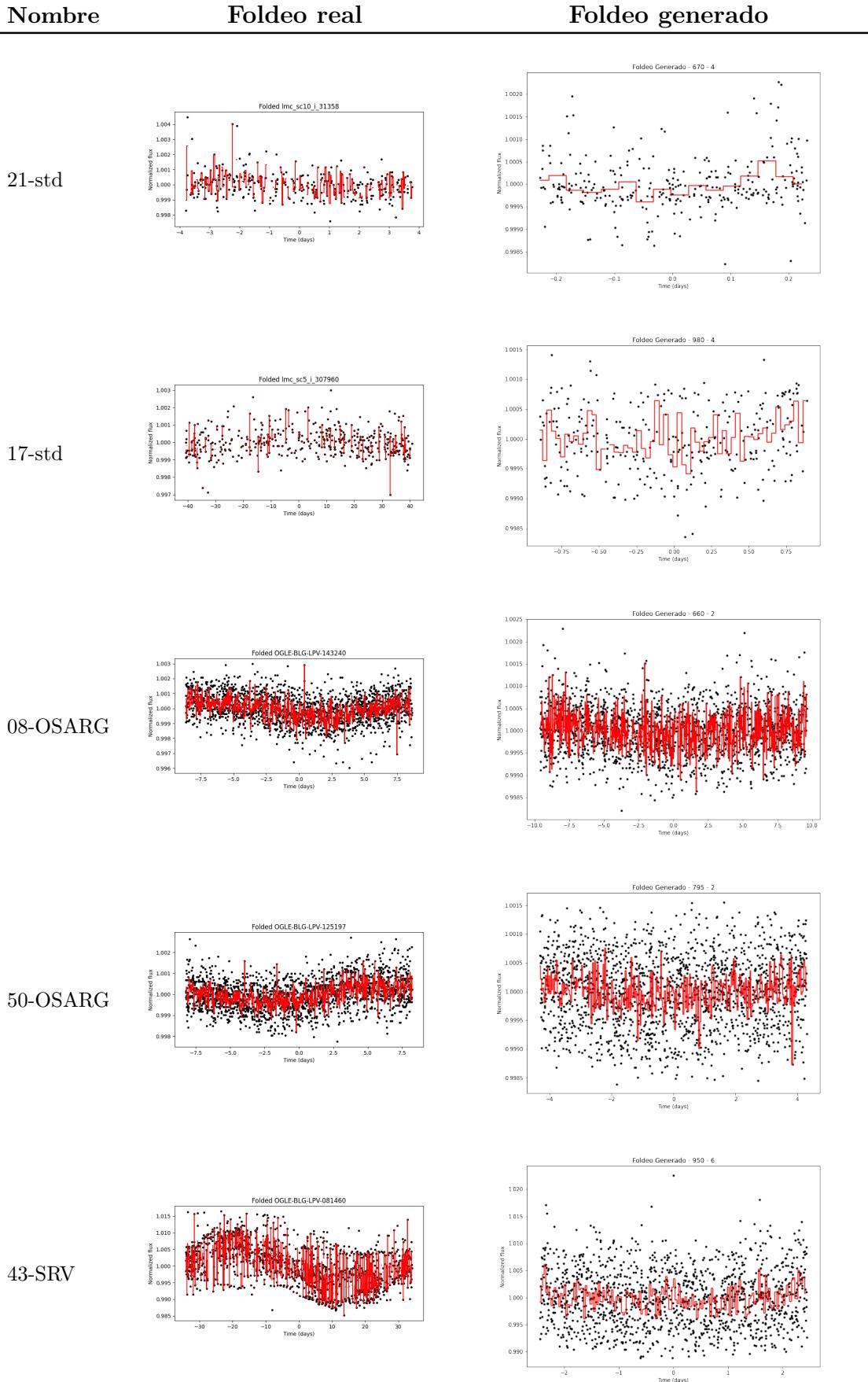
De la tabla número (5) se pueden extraer algunas instancias donde el foldeo se puede considerar muy bueno. Estas instancias corresponden a **05-ED** y **55-EC**, donde están respaldadas por el *score* de *accuracy* que tuvieron de la *LSTM Score*. Siendo **48,43 %** y **61,54 %** de *accuracy* respectivamente. Aquí hay que recordar que entre más cercano estén los valores de *accuracy* al 0,5 o 50 %, mejores son los resultados obtenidos. Por esto también, se definió la métrica para representar de mejor manera el resultado de los datos, definidos por la fórmula número (8). Luego, se puede ver que la instancia **13-RRc** tiene bastante ruido en los foldeos y no tiene tanta similitud al foldeo real, teniendo además, un *accuracy* de **91,25 %**, respaldando que los datos obtenidos no fueron muy buenos.

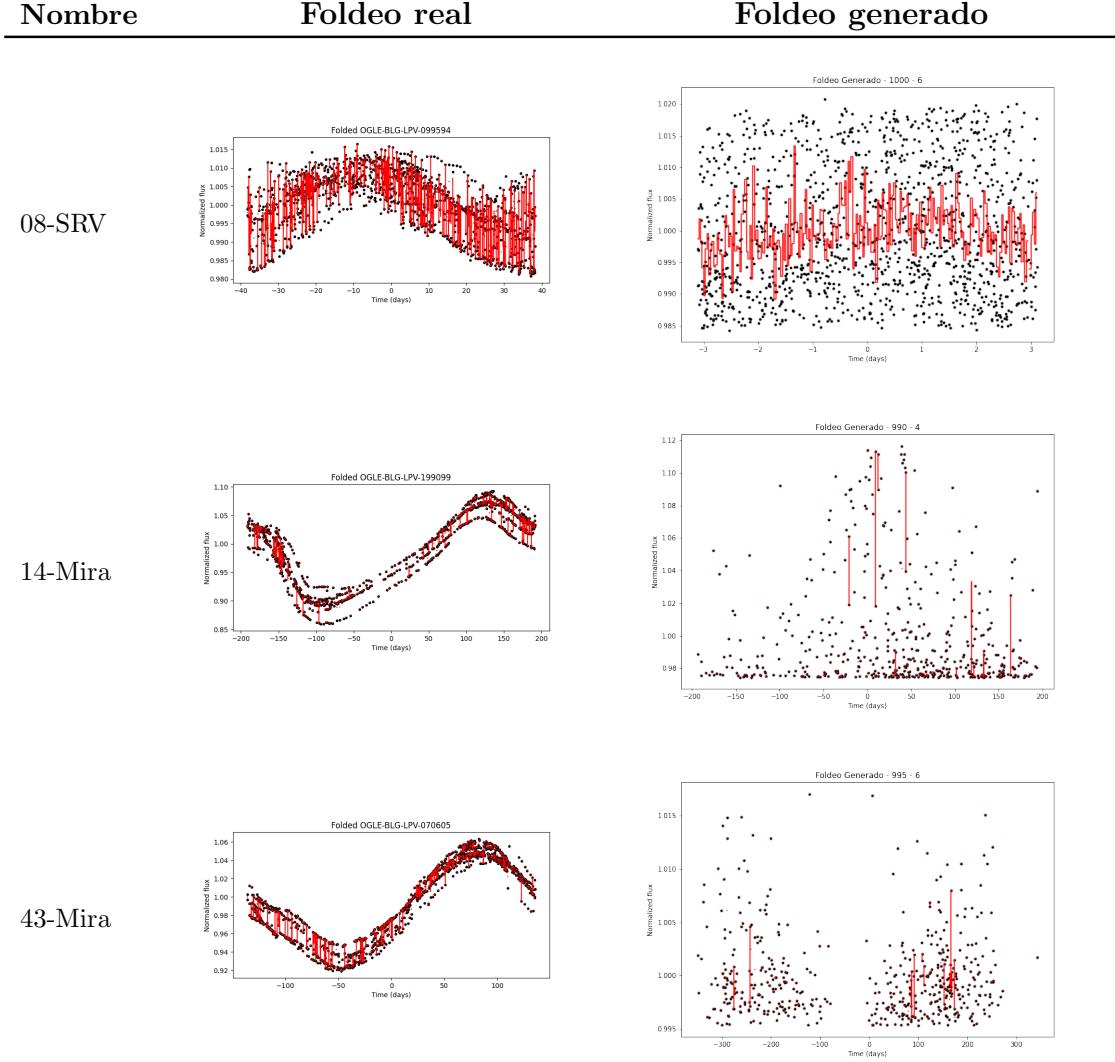
Tabla 5: Comparación entre foldeo real y generado de las instancias obtenidas por la GAN con generador de una capa del dataset **OGLE-III**.











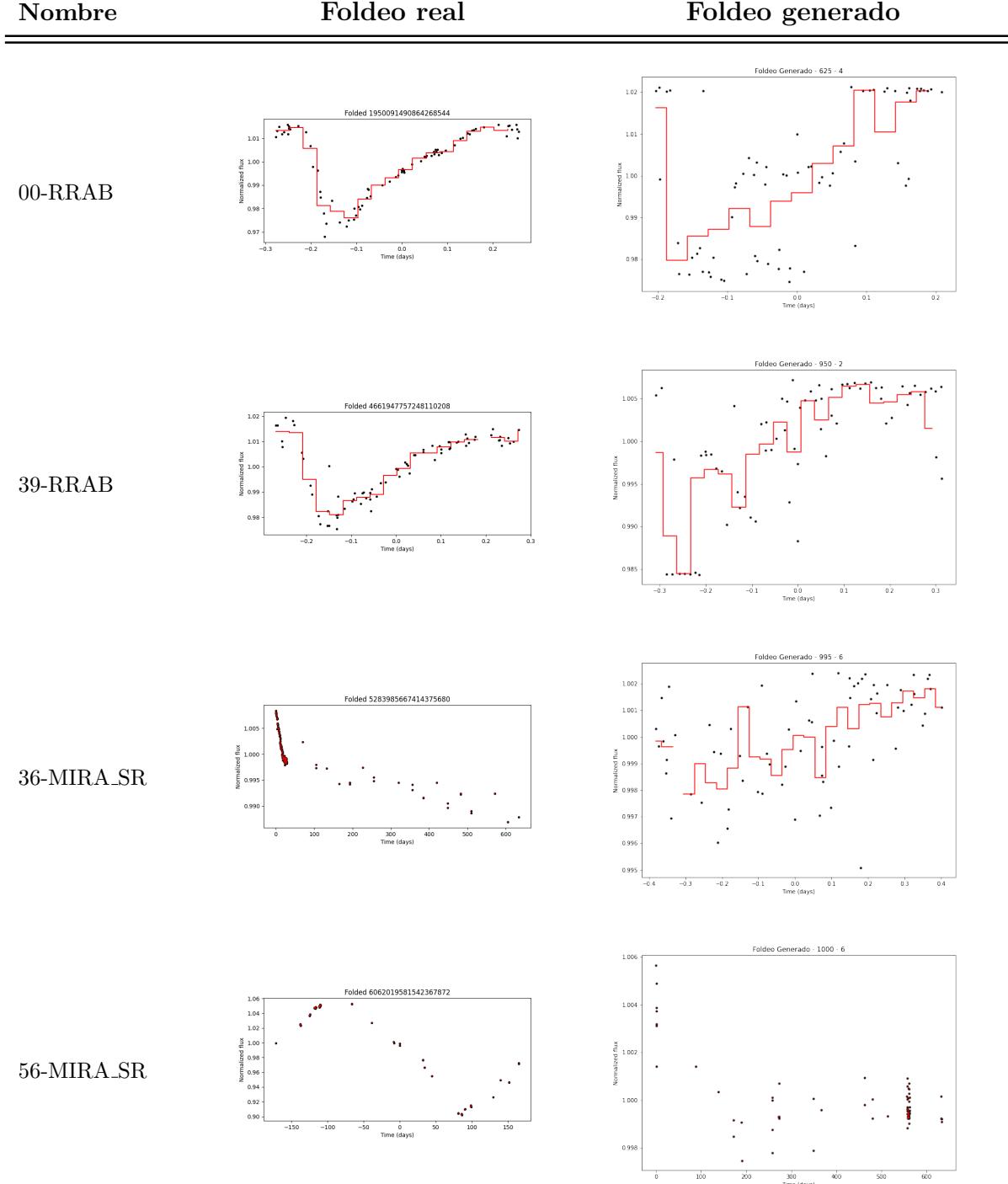
6.3.2. GAIA

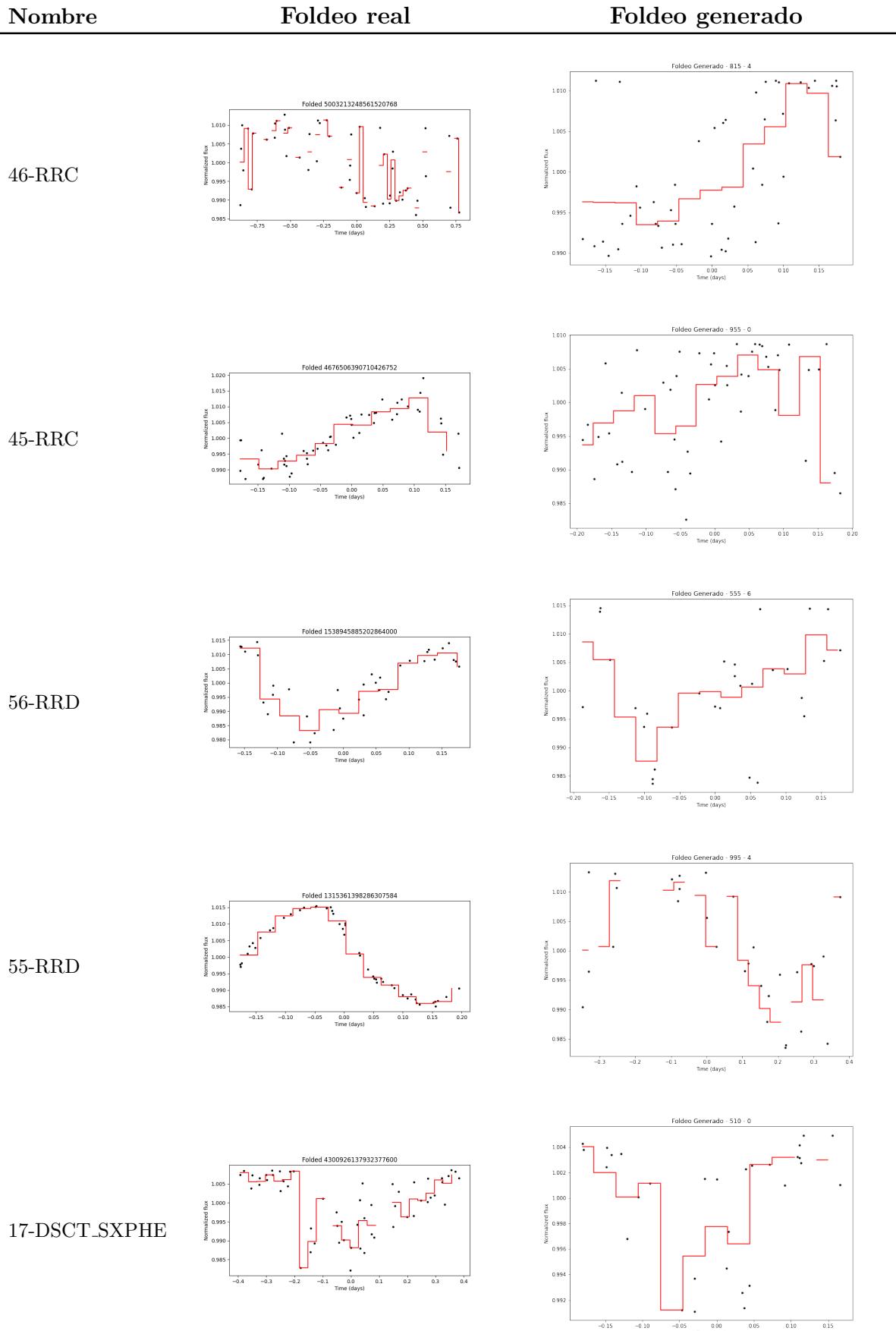
Dada la menor cantidad de mediciones que tiene este dataset comparado al **OGLE-III**, los datos generados y sus respectivos foldeos logran tener una mejor claridad, pero, también tienen resultados menos afinados. Podemos destacar las series con buenos resultados, siendo **30-DSCT_SXPHE** y **56-RRD** donde visualmente se asemejan bastante al foldeo real y son respaldadas por su *accuracy* de **52,48 %** y **40,74 %** respectivamente.

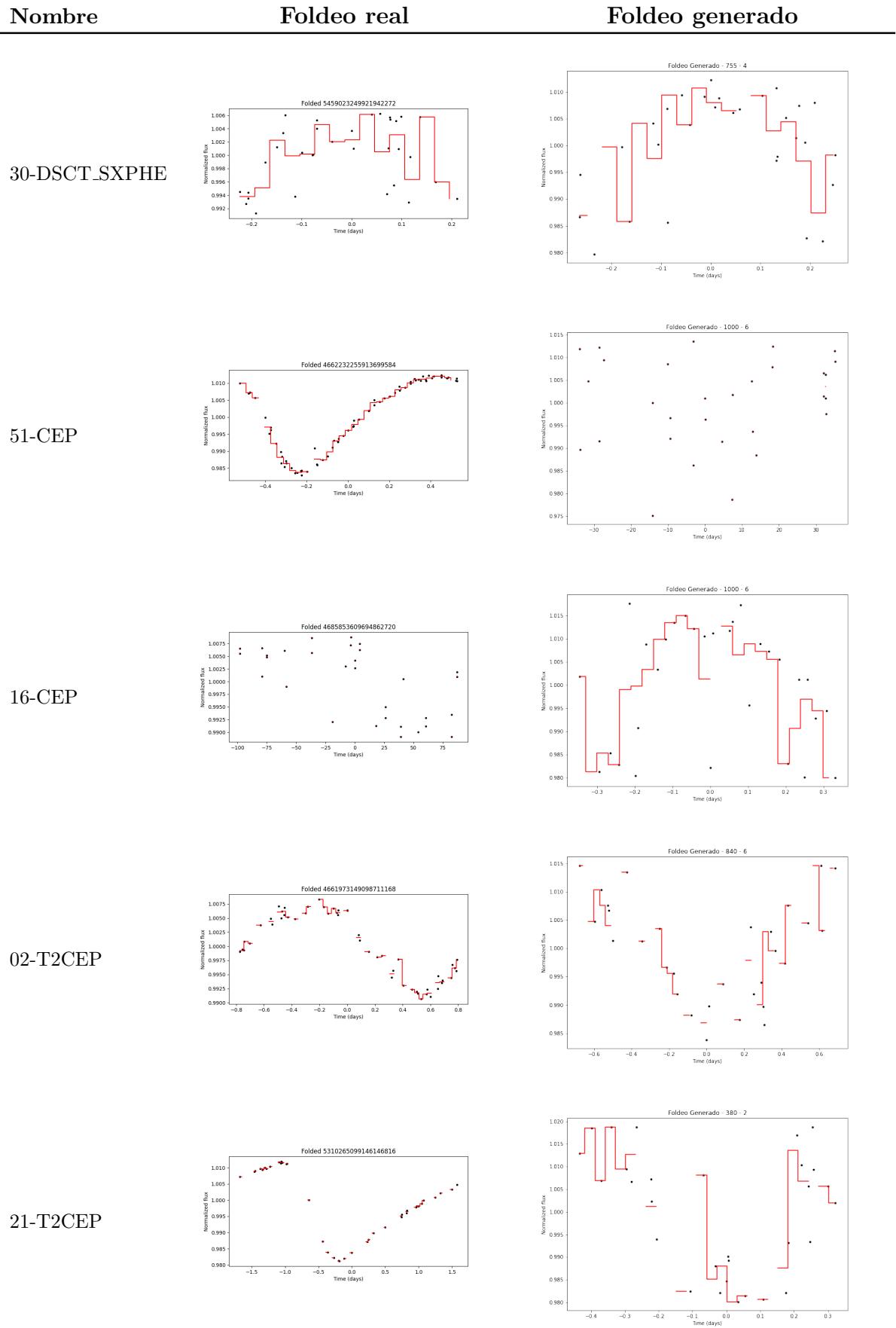
Por otro lado, si se analiza la instancia **51-CEP**, se puede apreciar que su foldeo generado no contiene ningún patrón o firma que sea similar al real, pese a que tiene un muy buen *accuracy* de **57,07 %**. Esto es parte de las instancias donde la red *GAN* no pudo generar series que entregaran resultados útiles de foldeo, ni durante su entrenamiento o posterior a ella. Sin embargo, aunque visualmente no tenga buenos resultados, para la *LSTM Score*, los datos generados si fueron lo suficientemente reales para que la categorizara con un buen *accuracy*. Este escenario ocurre en todos los dataset, donde también existen los casos que la *LSTM Score* entregó un *accuracy* muy bajo o muy elevado, demostrando que, no solo la red *GAN* no pudo generar datos que tuvieran visualmente un foldeo cercano al real, sino que, también no fue capaz de generar datos similares a los reales.

Si se analiza la clase **RRAB**, se puede ver que los foldeos de ambas instancias, intentan seguir el patrón del foldeo real, aunque no sea del todo fluido. Sin embargo, el *accuracy %* obtenido de estas instancias es de **89,1 %** y **25,2 %** respectivamente. Este escenario también ocurre en varias ocasiones, donde el foldeo generado es cercano al real, pero, las instancias no logran tener un buen *accuracy*. No obstante, esto no implica necesariamente que sean considerados malas instancias.

Tabla 6: Comparación entre foldeo real y generado de las instancias obtenidas por la **GAN** con generador de una capa del dataset **GAIA**.



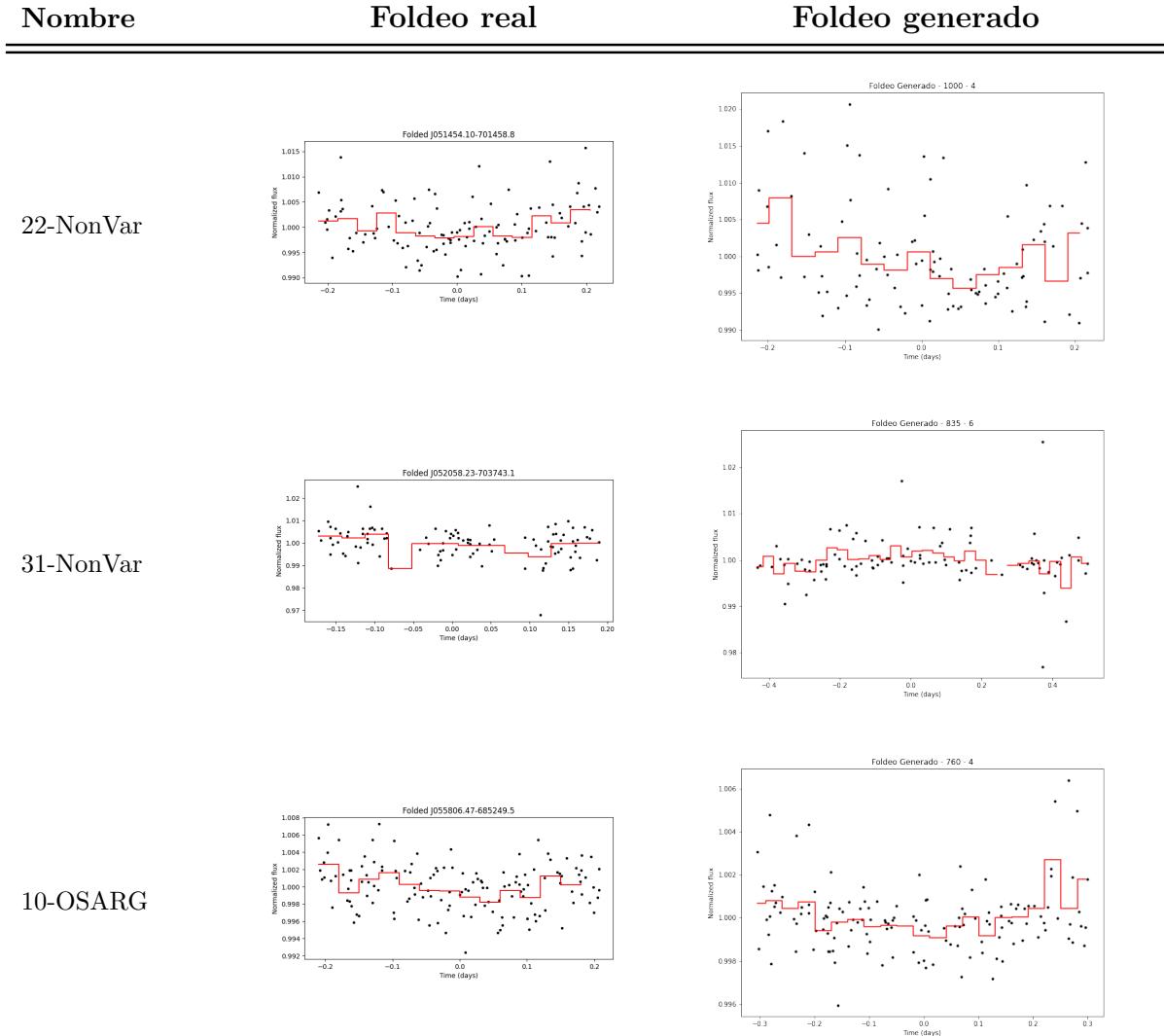


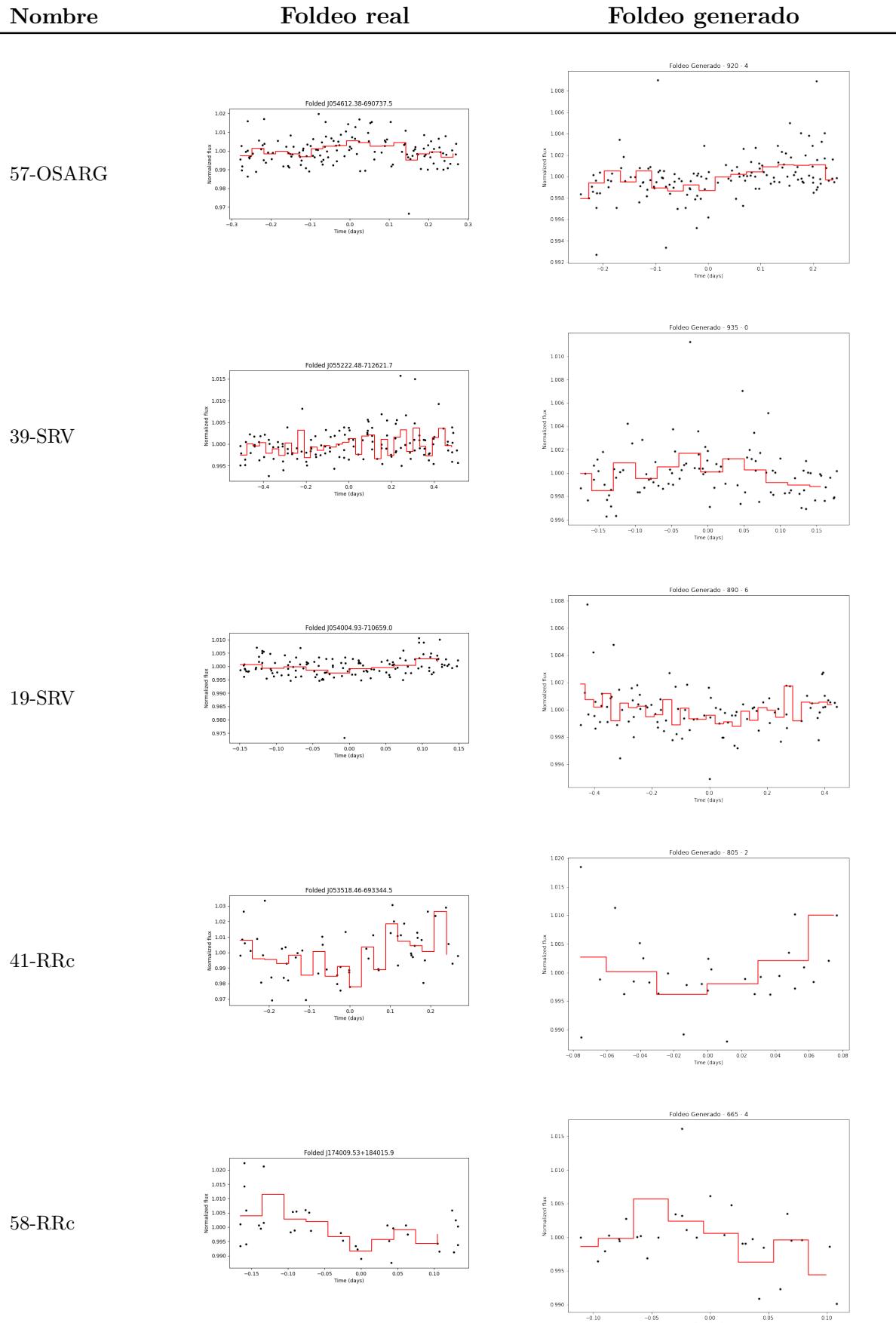


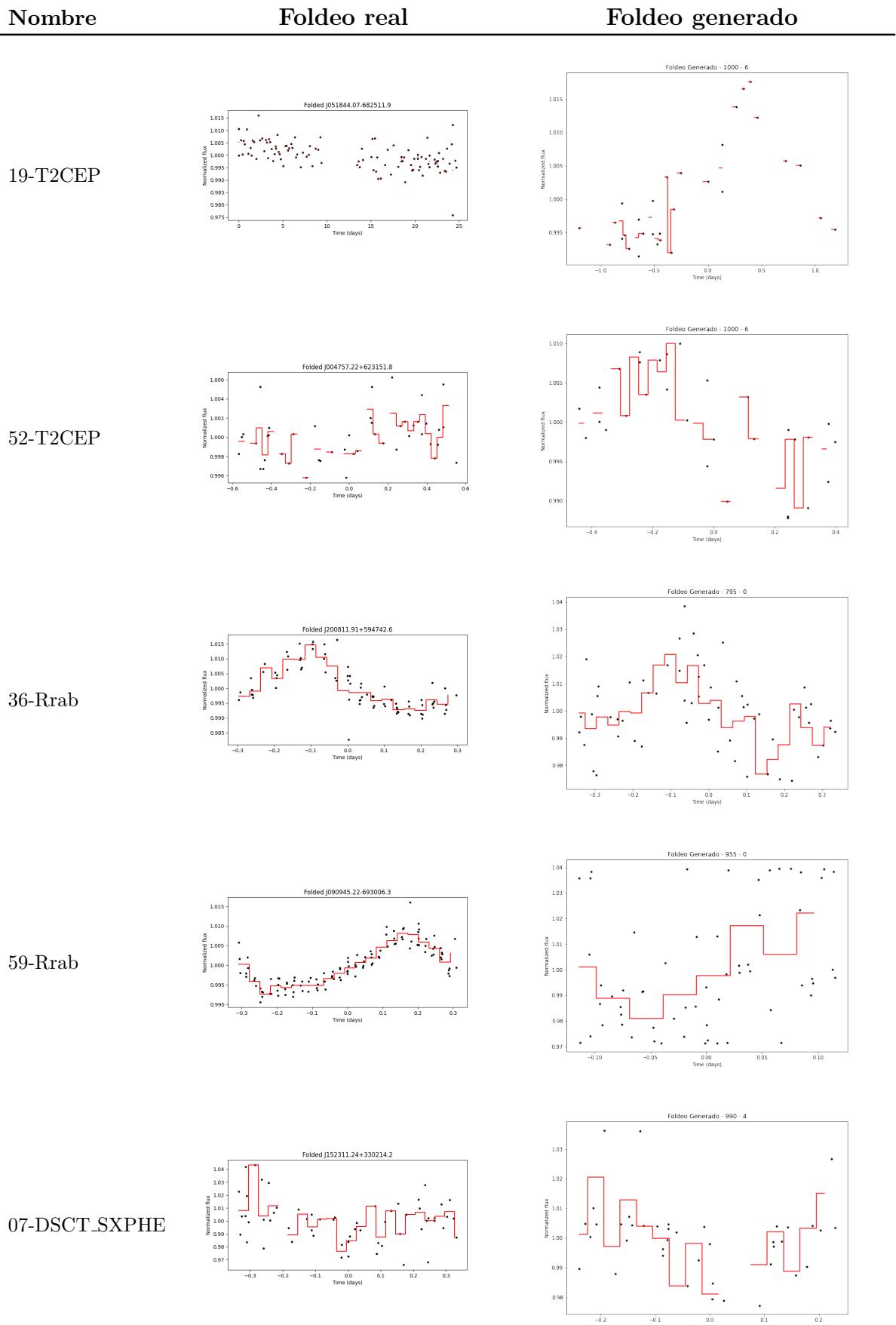
6.3.3. WISE

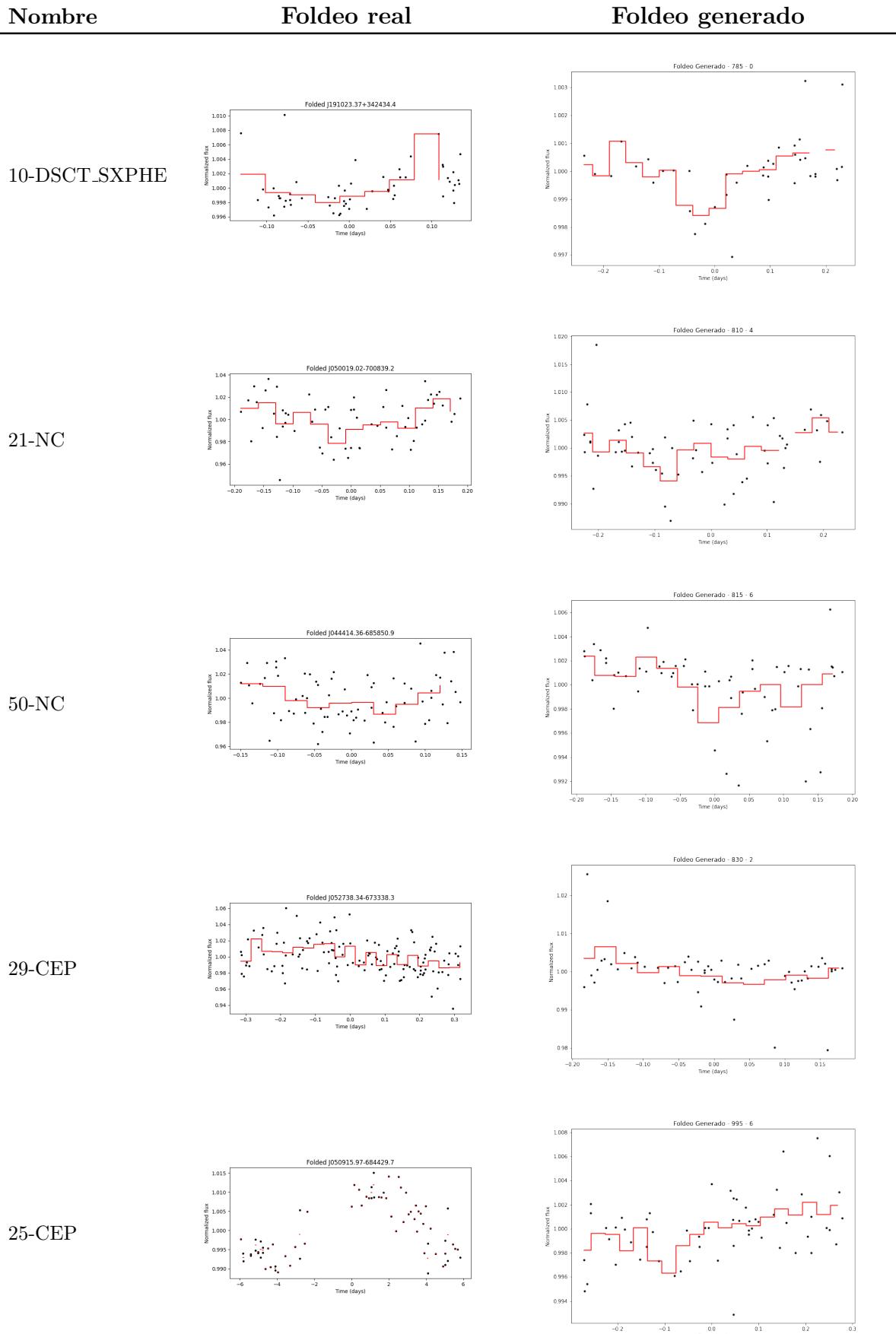
Precisamente, en este dataset se puede apreciar que los foldeos reales son en su mayoría, más planos a los vistos hasta entonces. Esto, de alguna forma ayuda a que los datos generados por la red *GAN* sean propensos a tener mayor similitud a los datos reales. Podemos catalogar como buenas instancias la **31-NonVar** y **57-OSARG**, ya que, estas instancias tienen curvas bastante planas, pero, tienen muy poco ruido, haciendo que visualmente sean bien similares al foldeo real. Por otro lado, si se analiza el *accuracy* obtenido, **59,17 %** y **76,85 %** respectivamente, de la primera instancia se entiende que los datos generados junto a la muestra entregada a la *LSTM Score* para ser evaluada, estuvieron bastante buenos pudieron ser identificados y categorizados correctamente. A diferencia, la segunda instancia, donde el *accuracy* es más alto, podría implicar que la *LSTM Score* no fue capaz de categorizar correctamente los datos recibidos, identificando los datos generados como reales, causando un aumento en su *accuracy* final.

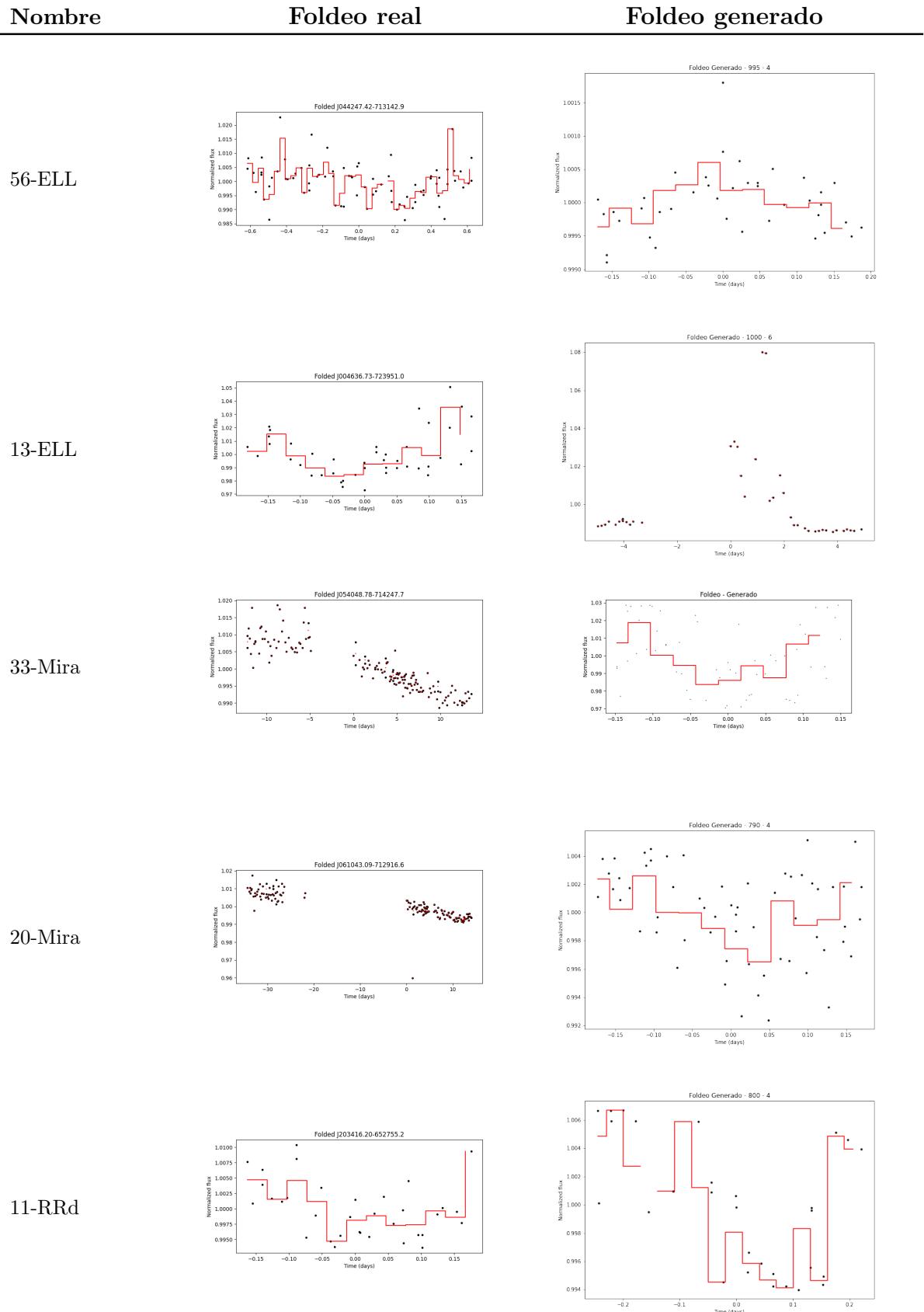
Tabla 7: Comparación entre foldeo real y generado de las instancias obtenidas por la **GAN** con generador de una capa del dataset **WISE**.

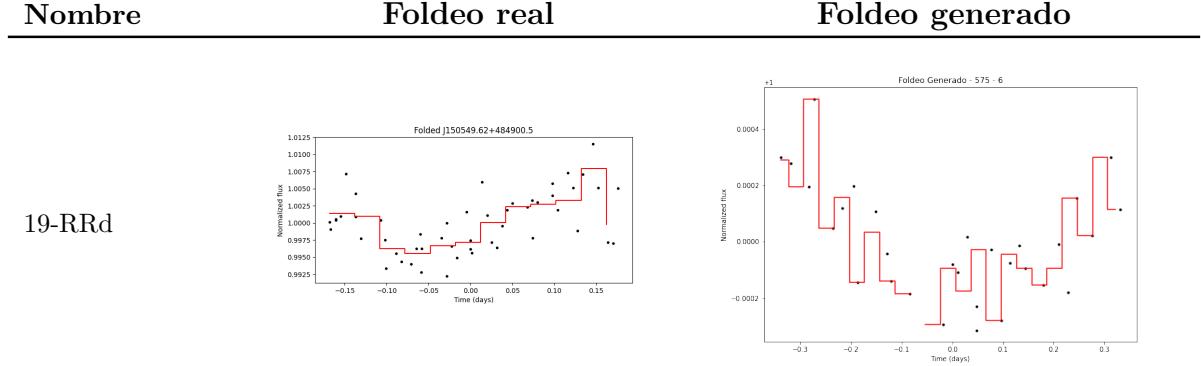










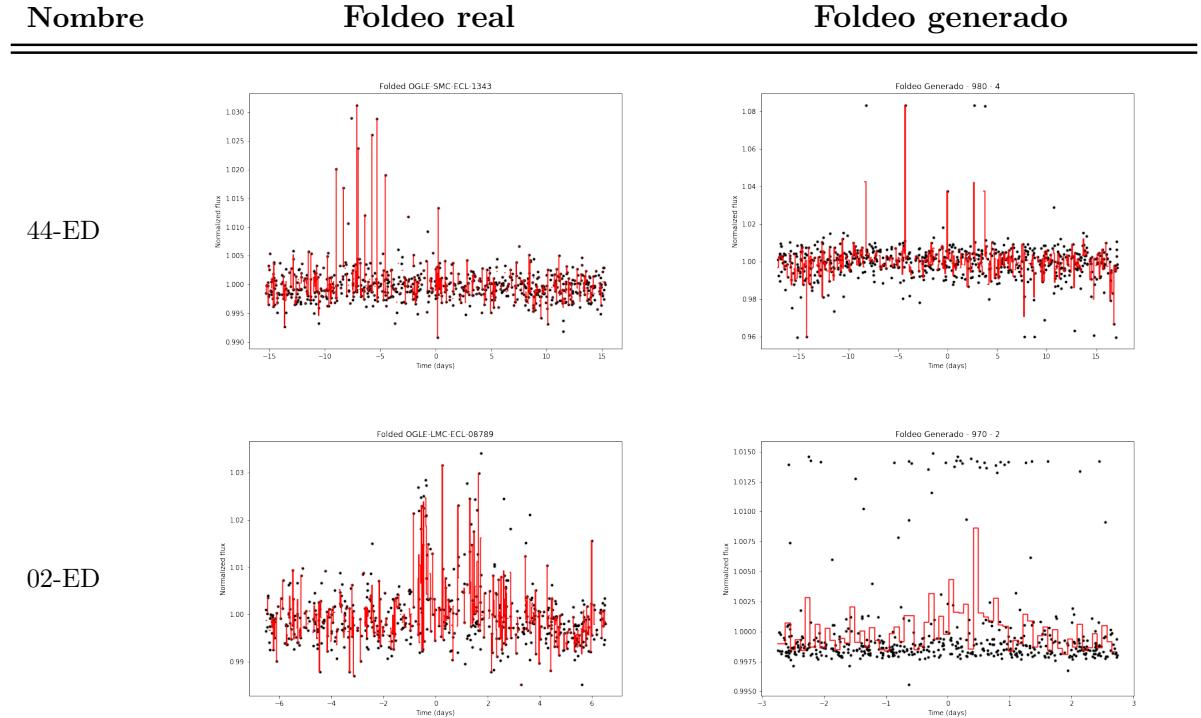


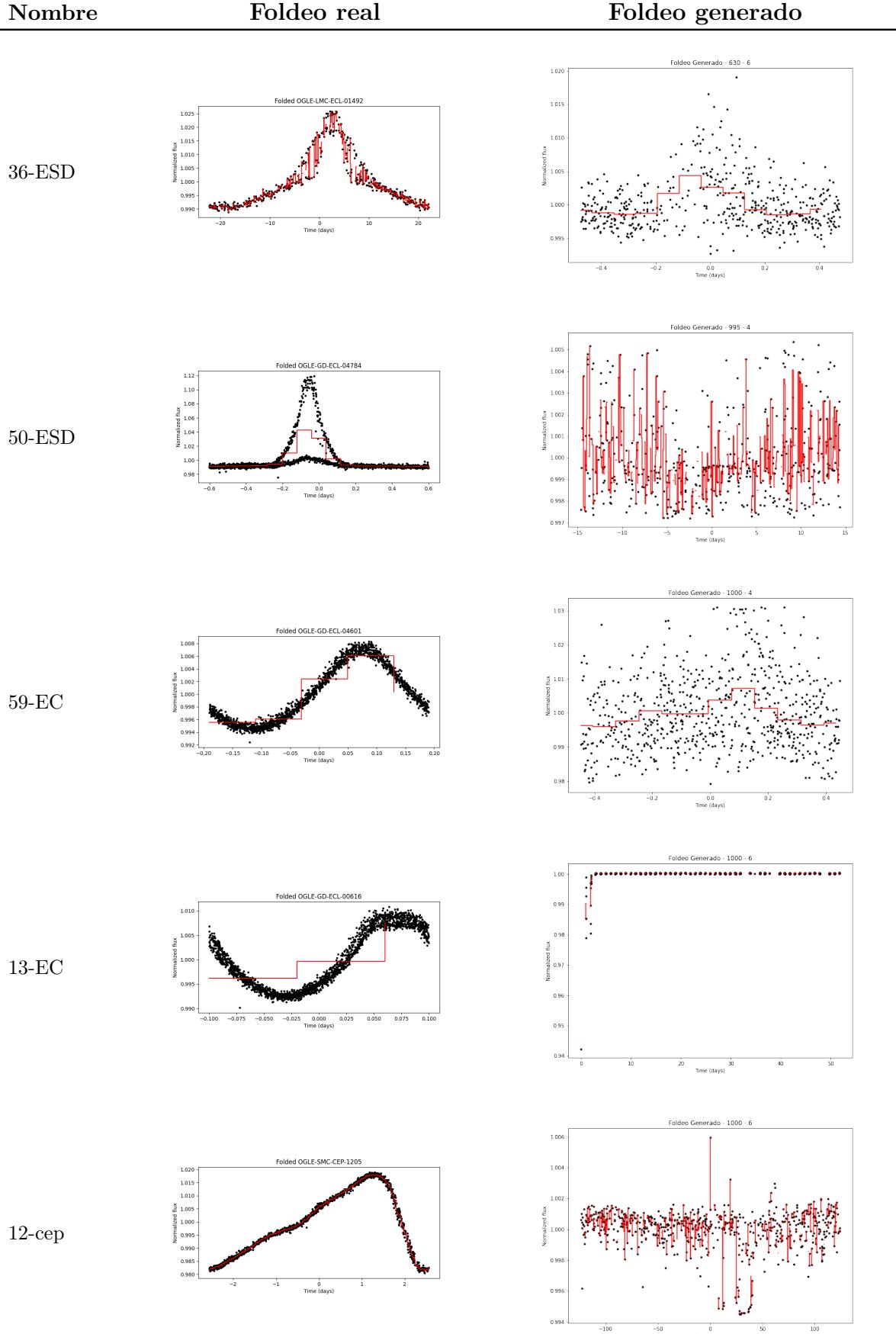
6.4. Segundo experimento con Generador de dos capas

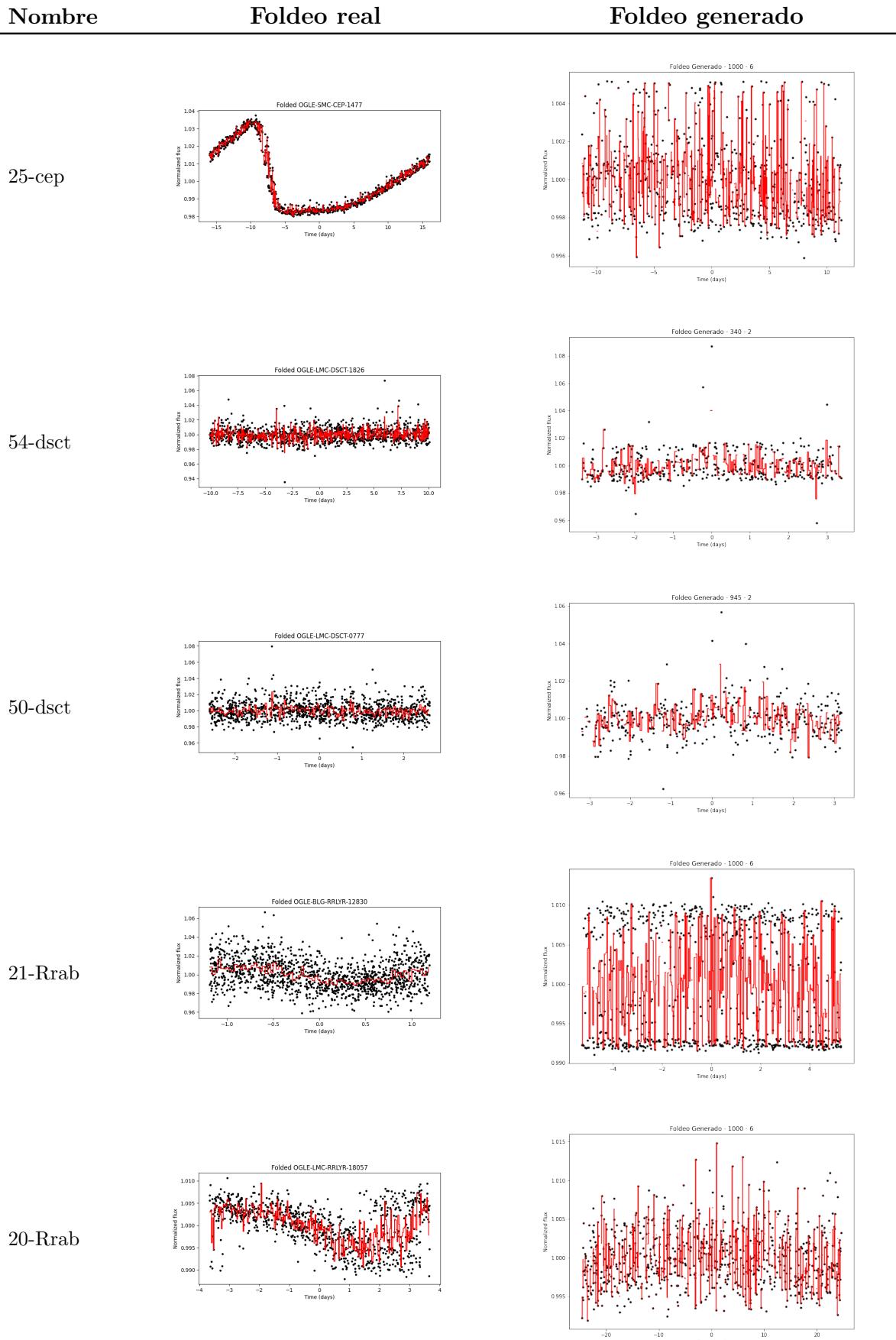
6.4.1. OGLE-III

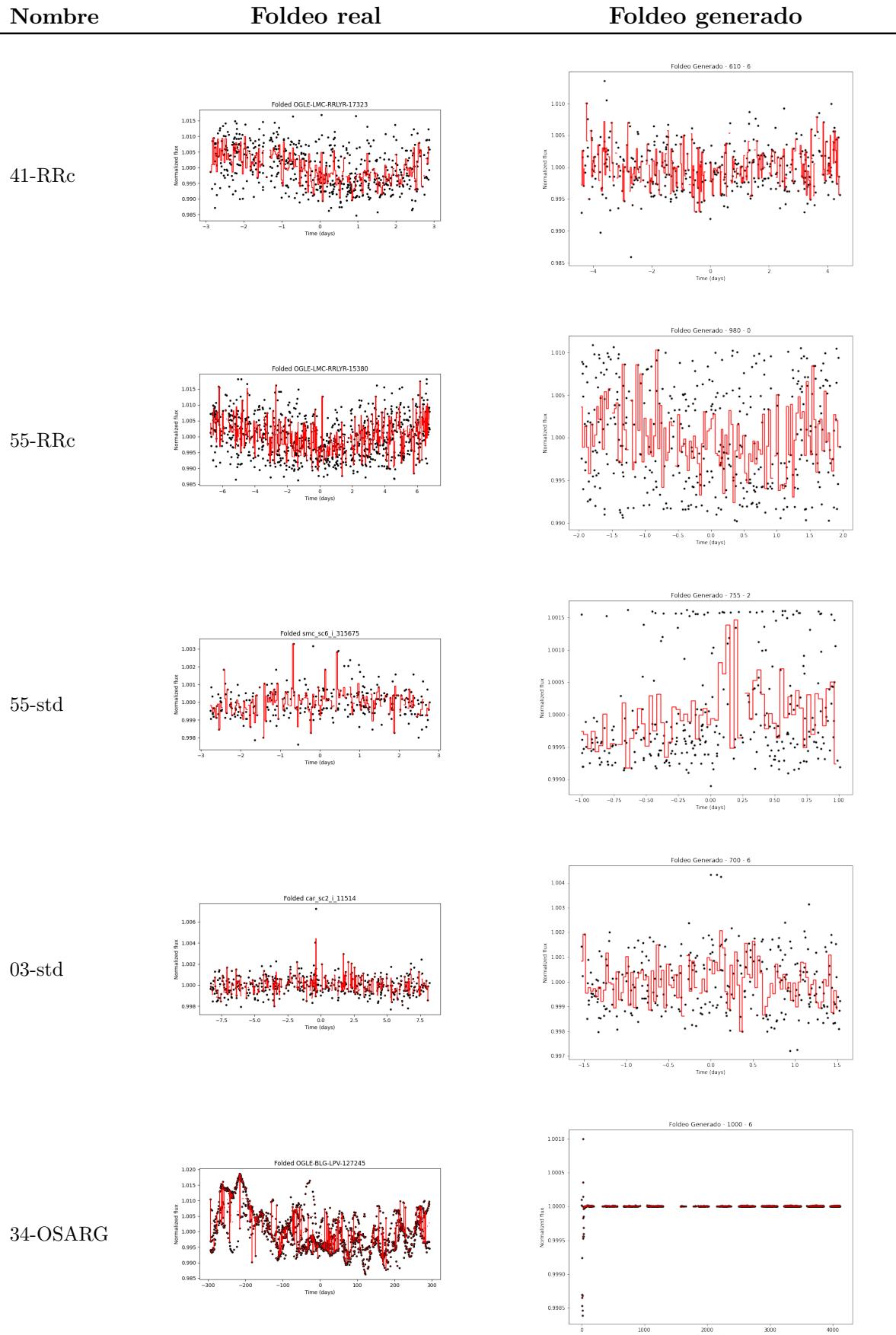
En este dataset, dicha configuración del modelo *GAN*, no entregó muy buenos resultados, ya que mayoritariamente, los foldeos generados aparentan ser bastante planos y con mucho ruido. Se puede apreciar en la instancia **36-ESD** que presenta un *accuracy* de **40,55 %**, donde el foldeo generado intenta seguir la pequeña curvatura del foldeo real, pero, no es capaz de seguirla completamente, teniendo también, bastante ruido de por medio. La instancia **29-SRV**, sigue un camino similar a la anterior, ya que del foldeo generado se puede apreciar la curvatura generada junto a su foldeo, pese a que ambos presentan un foldeo ruidoso y no tan definido como otras instancias. El *accuracy* obtenido en este caso fue de **65,49 %**.

Tabla 8: Comparación entre foldeo real y generado de las instancias obtenidas por la **GAN** con generador de dos capas del dataset **OGLE-III**.



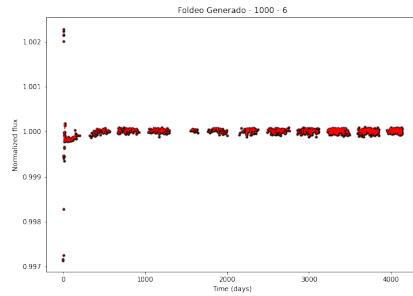
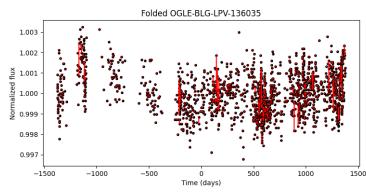




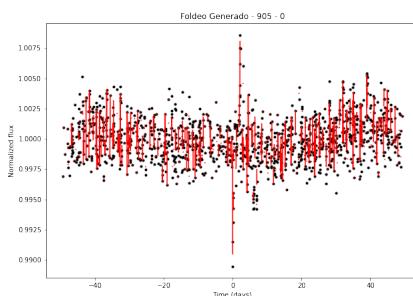
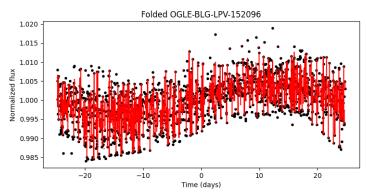


Nombre**Foldeo real****Foldeo generado**

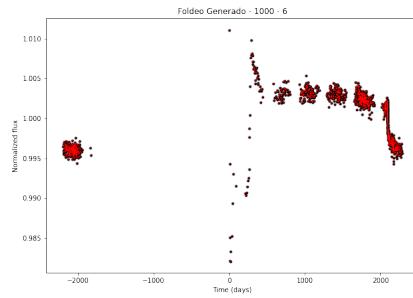
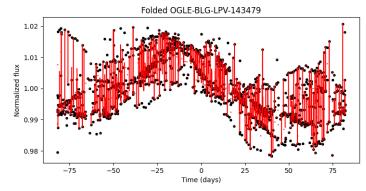
23-OSARG



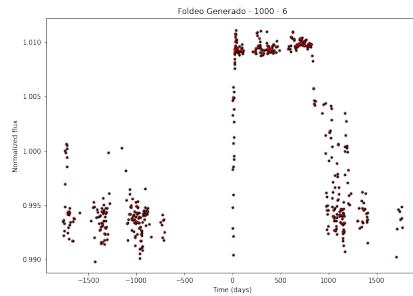
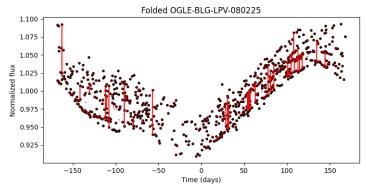
29-SRV



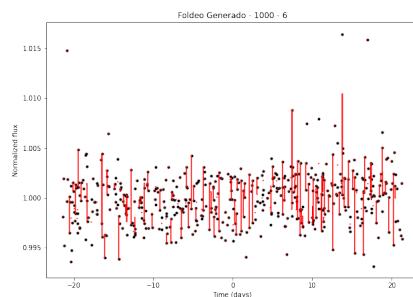
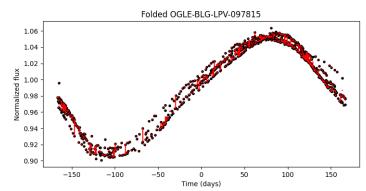
29-SRV



25-Mira



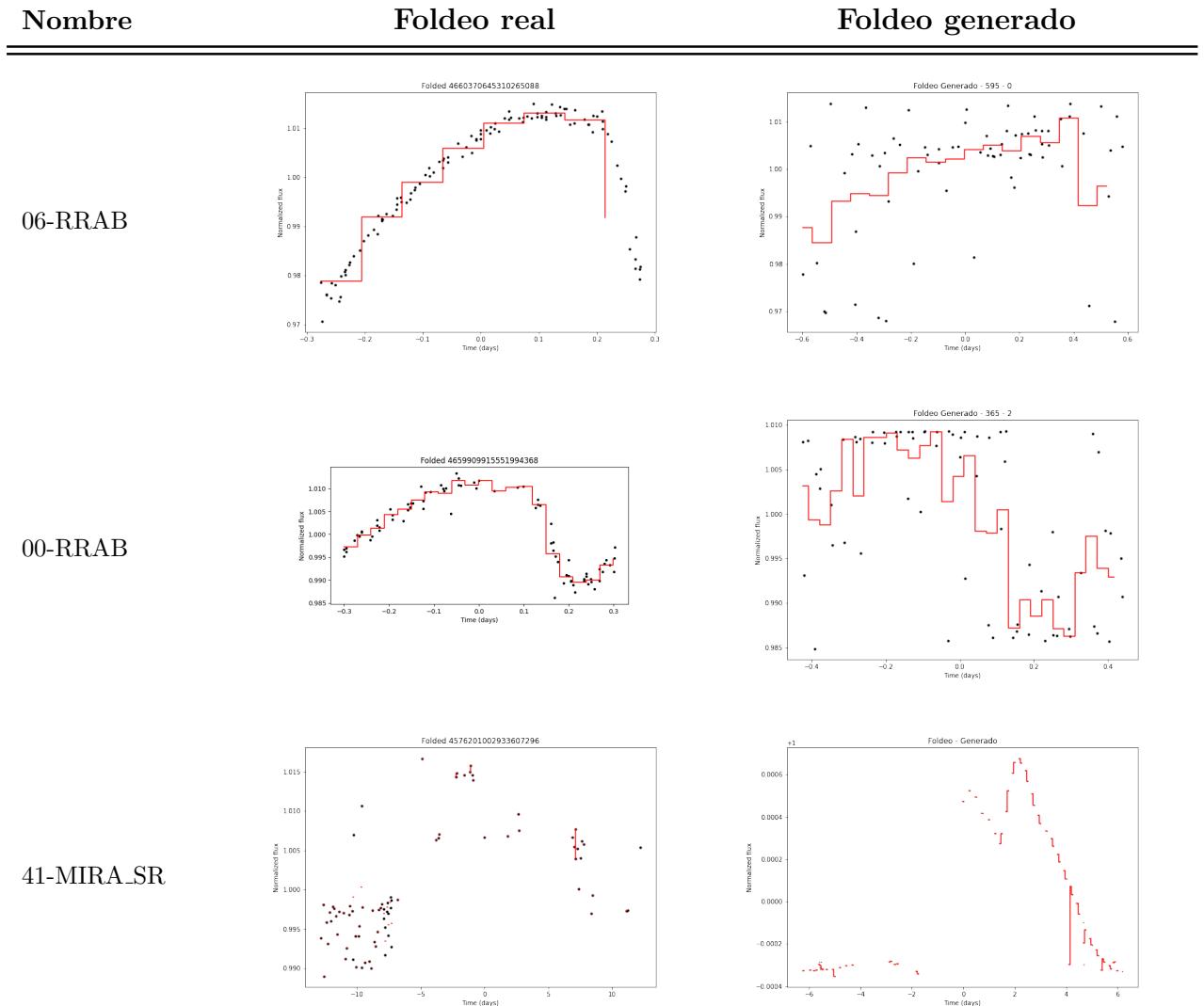
46-Mira

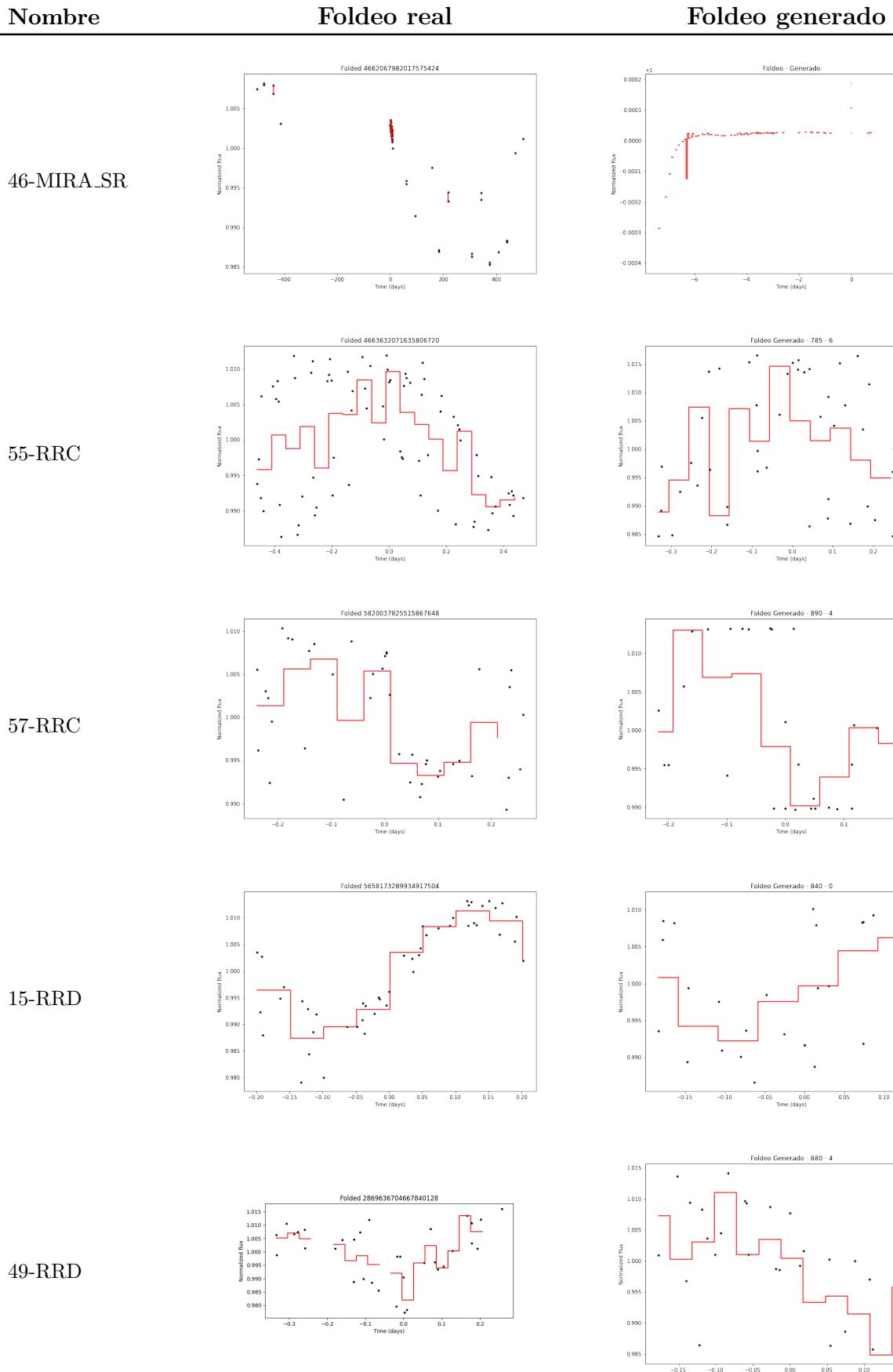


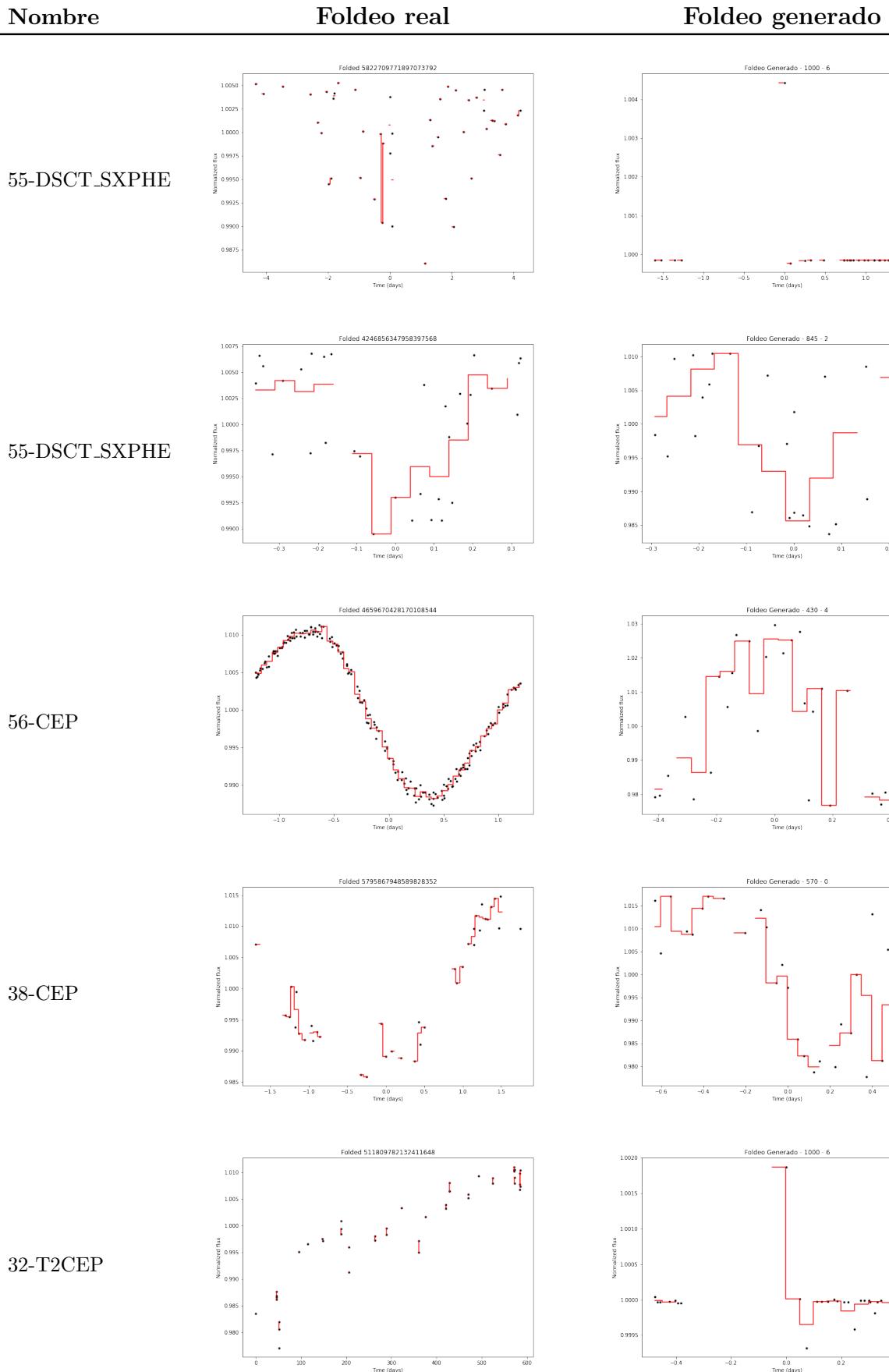
6.4.2. GAIA

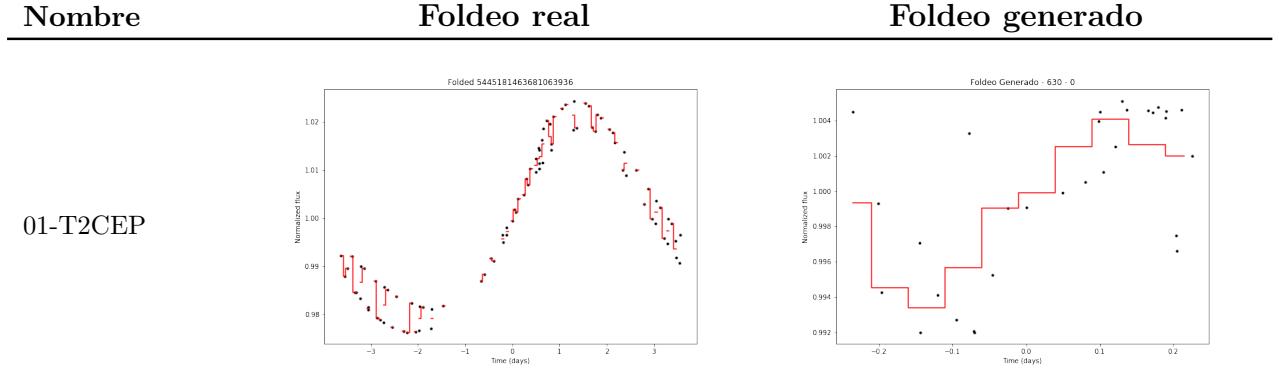
El dataset de GAIA, no tiene foldeos muy definidos debido a la cantidad de mediciones que tiene cada serie, por esto, los foldeos generados suelen estar más desorganizados y sin curvas muy marcadas. Por ejemplo, la instancia **57-RRC** con *accuracy* de **62,58 %**, tiene un foldeo generado bien similar al real, pero, en ambos casos la curva del foldeo no es tan definida como casos de otros datasets. Esto se debe a la poca cantidad de muestras que se tienen, dificultando la construcción de un foldeo más fluido, obteniendo estas cambios de direcciones más marcados. El caso de la instancia **55-DSCT_SXPHE**, presenta un *accuracy* de **86,08 %**, que se puede ver que no tiene un foldeo real muy bueno, y por consiguiente, tampoco se tiene un foldeo generado. Esto se debe a que sencillamente los datos usados para esa instancia no fueron de muy buena calidad, por lo que tampoco se pudo generar datos que pudiesen dar algún indicio de foldeo.

Tabla 9: Comparación entre foldeo real y generado de las instancias obtenidas por la **GAN** con generador de dos capas del dataset **GAIA**.





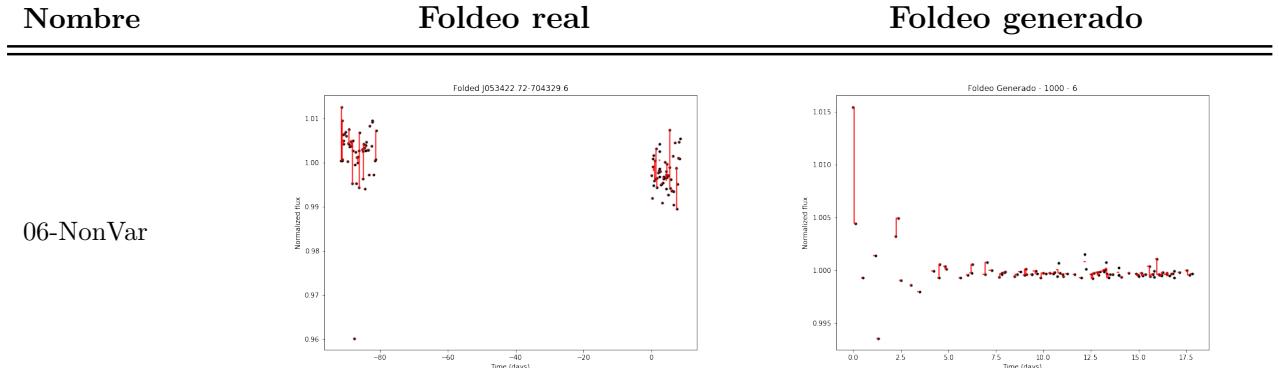


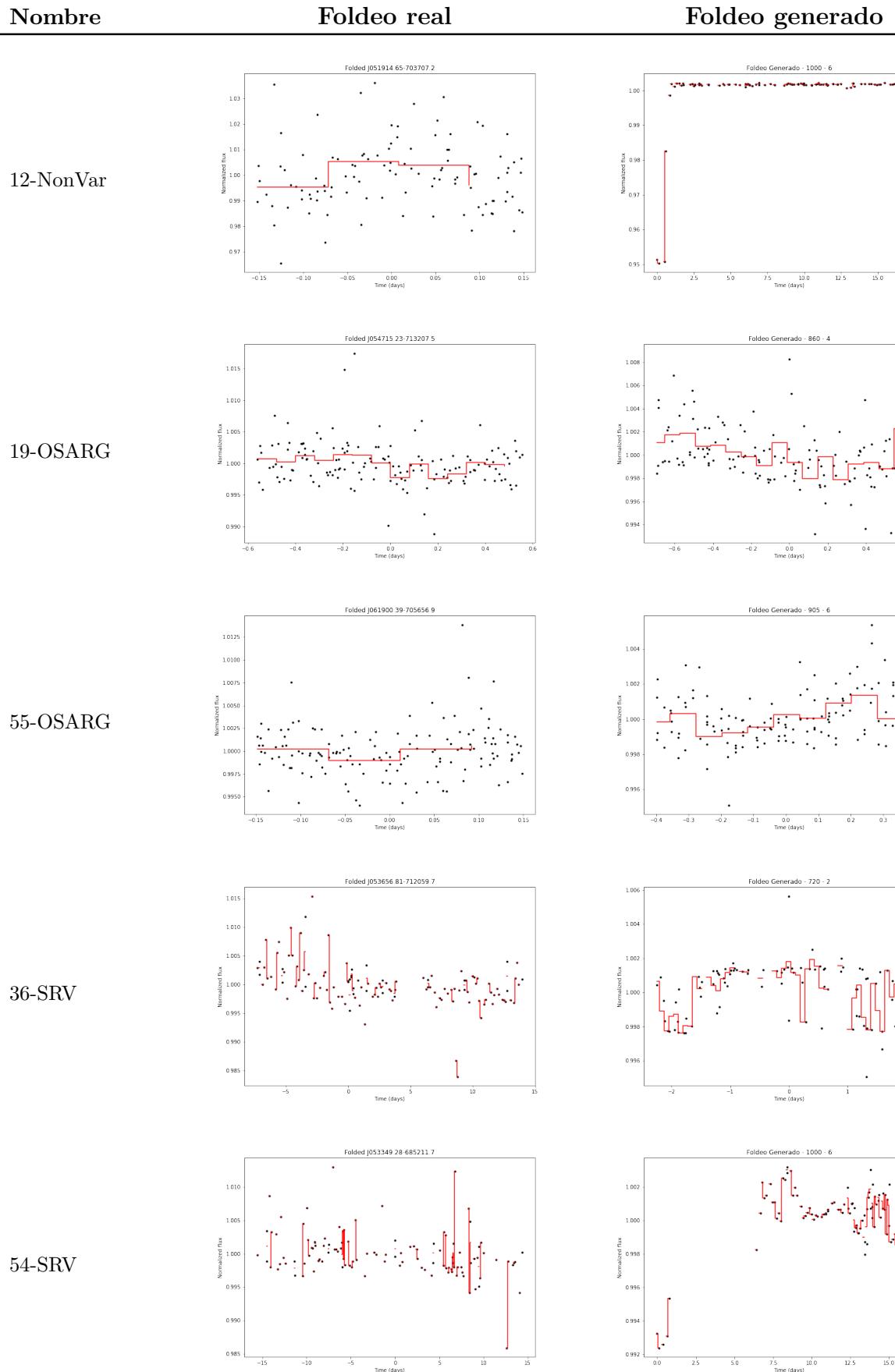


6.4.3. WISE

Este último dataset, presenta una mayor cantidad de mediciones que el anterior, pero, aun así, logran existir instancias donde no se pueden foldear los datos reales. Como es el caso de la instancia **06-NonVar** con un *accuracy* de **34,85 %**, no se puede apreciar un foldeo claro, ya que probablemente los valores de tiempo no fueron suficientes para poder marcar y definir un patrón en los cambios de magnitudes. Por otra parte, si se analiza la instancia **11-RRc**, se puede ver que el foldeo generado es bastante similar al real, e incluso, presenta menos ruido de por medio. Sin embargo, su *accuracy* de **13,58 %**, implica que los datos generados no fueron muy buenos, ya que, mayoritariamente, están catalogados como falsos. Adicionalmente, se pueden ver las instancias **56-ELL** y **56-DSCT_SXPHE** con *accuracy* de **37 %** y **34,68 %** respectivamente, donde no tienen un foldeo de datos reales muy marcado, pero, su foldeo de datos generados si presentan una curva. Este escenario, se puede dar debido a que las muestras de datos seleccionadas para mostrar el foldeo real no fueron lo suficientemente buenas o de una calidad suficiente.

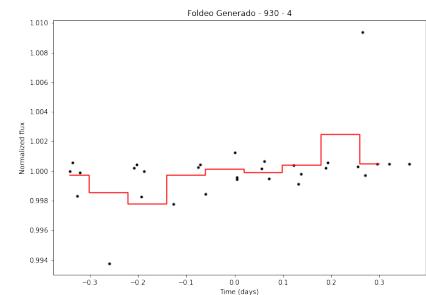
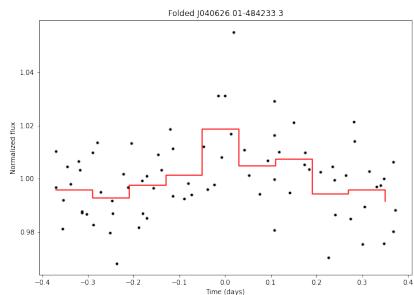
Tabla 10: Comparación entre foldeo real y generado de las instancias obtenidas por la **GAN** con generador de dos capas del dataset **WISE**.



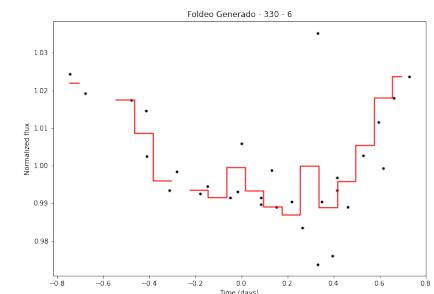
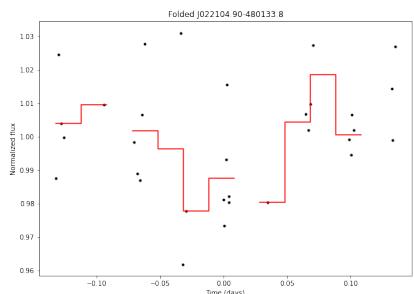


Nombre	Foldeo real	Foldeo generado
--------	-------------	-----------------

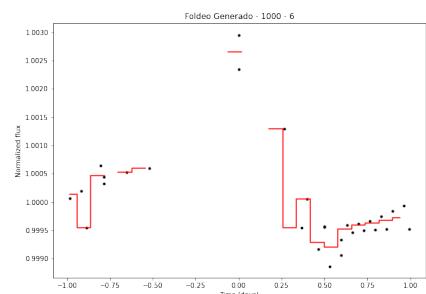
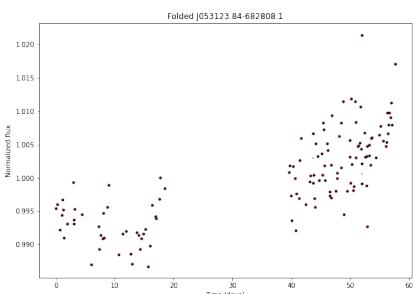
46-RRc



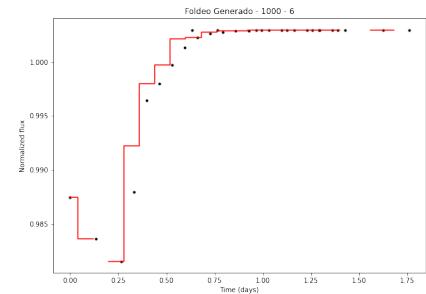
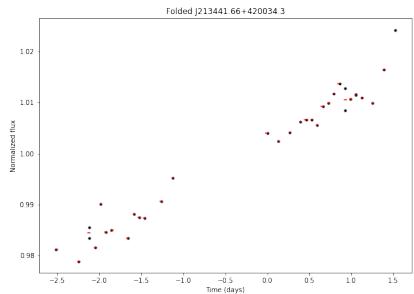
11-RRc



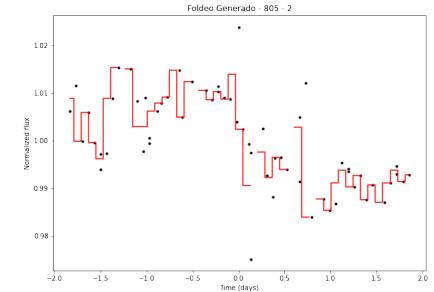
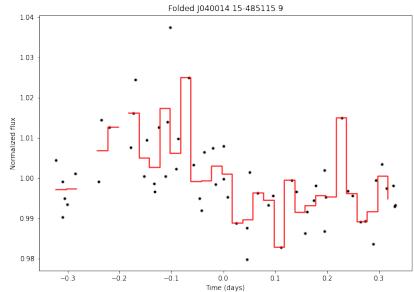
07-T2CEP

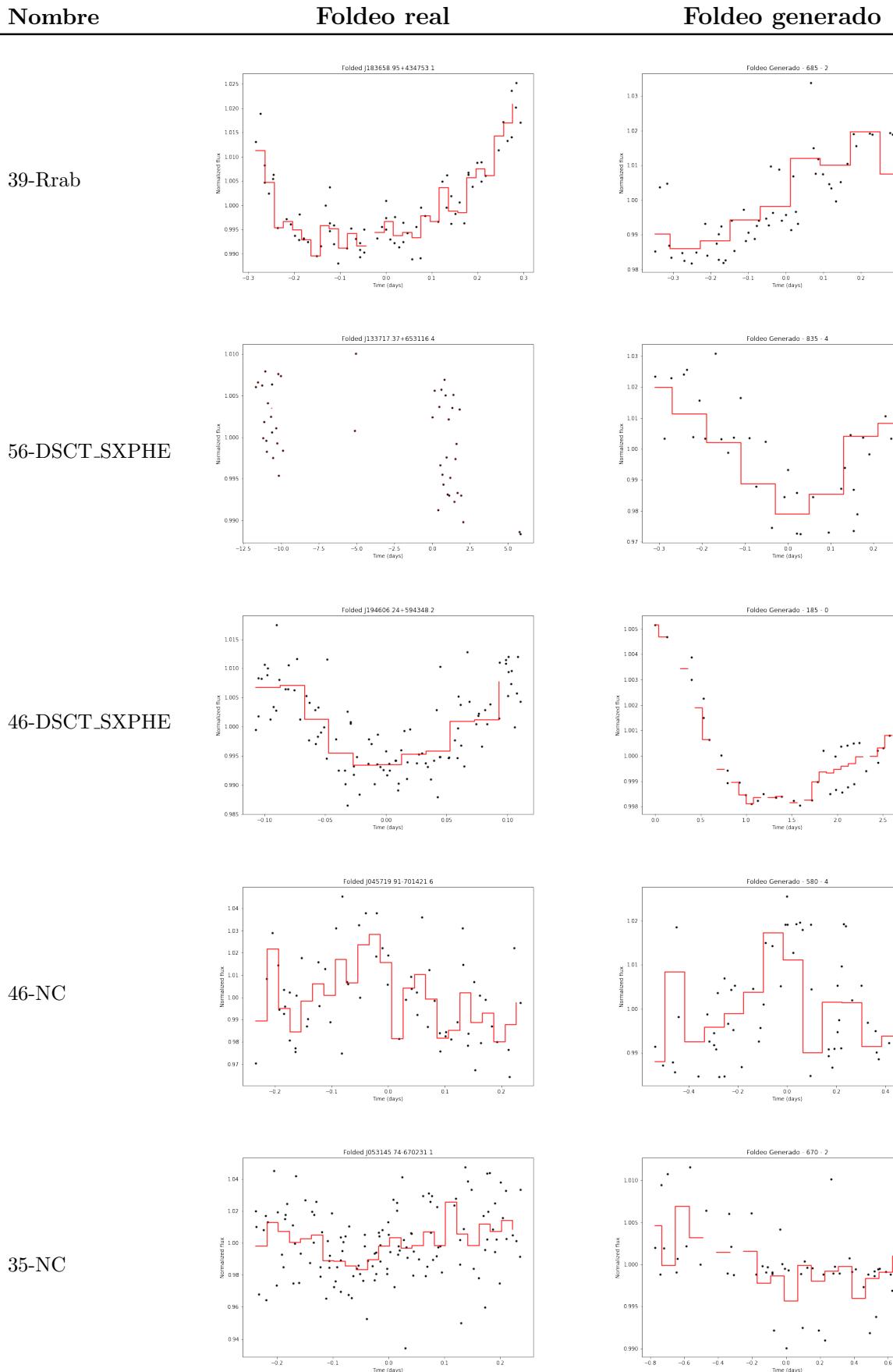


59-T2CEP



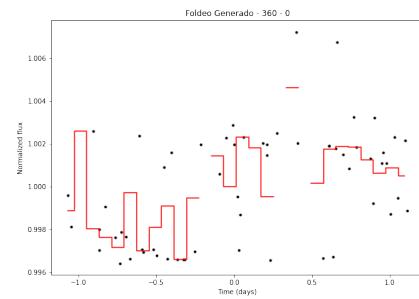
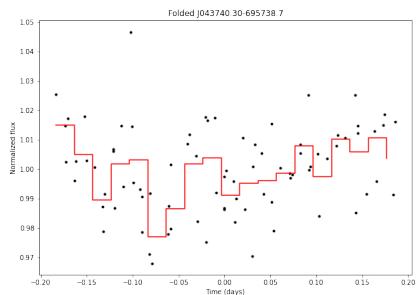
14-Rrab



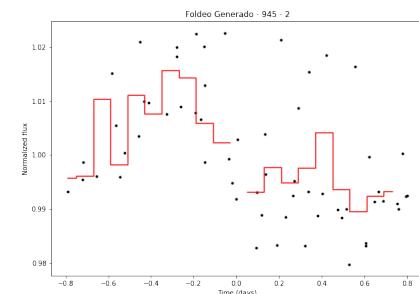
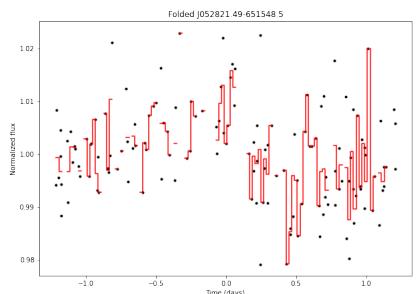


Nombre**Foldeo real****Foldeo generado**

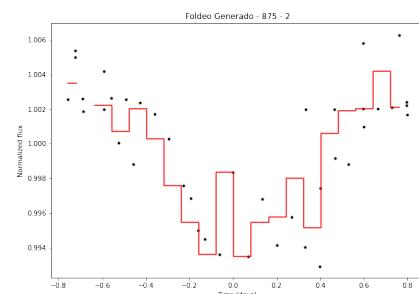
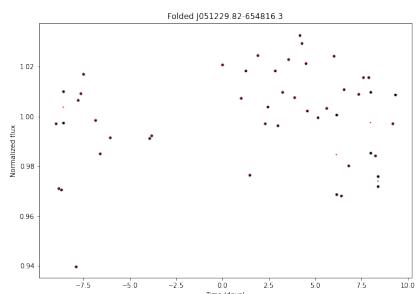
02-CEP



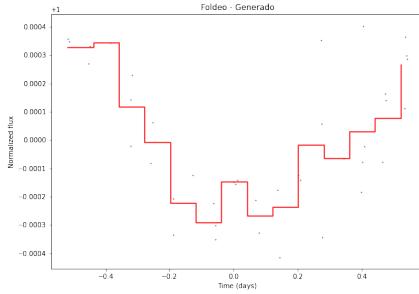
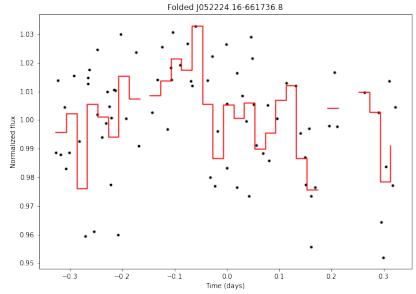
47-CEP



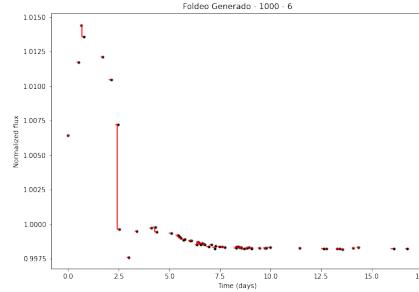
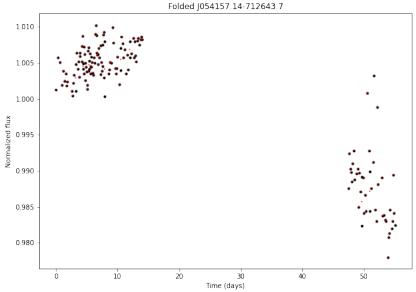
56-ELL

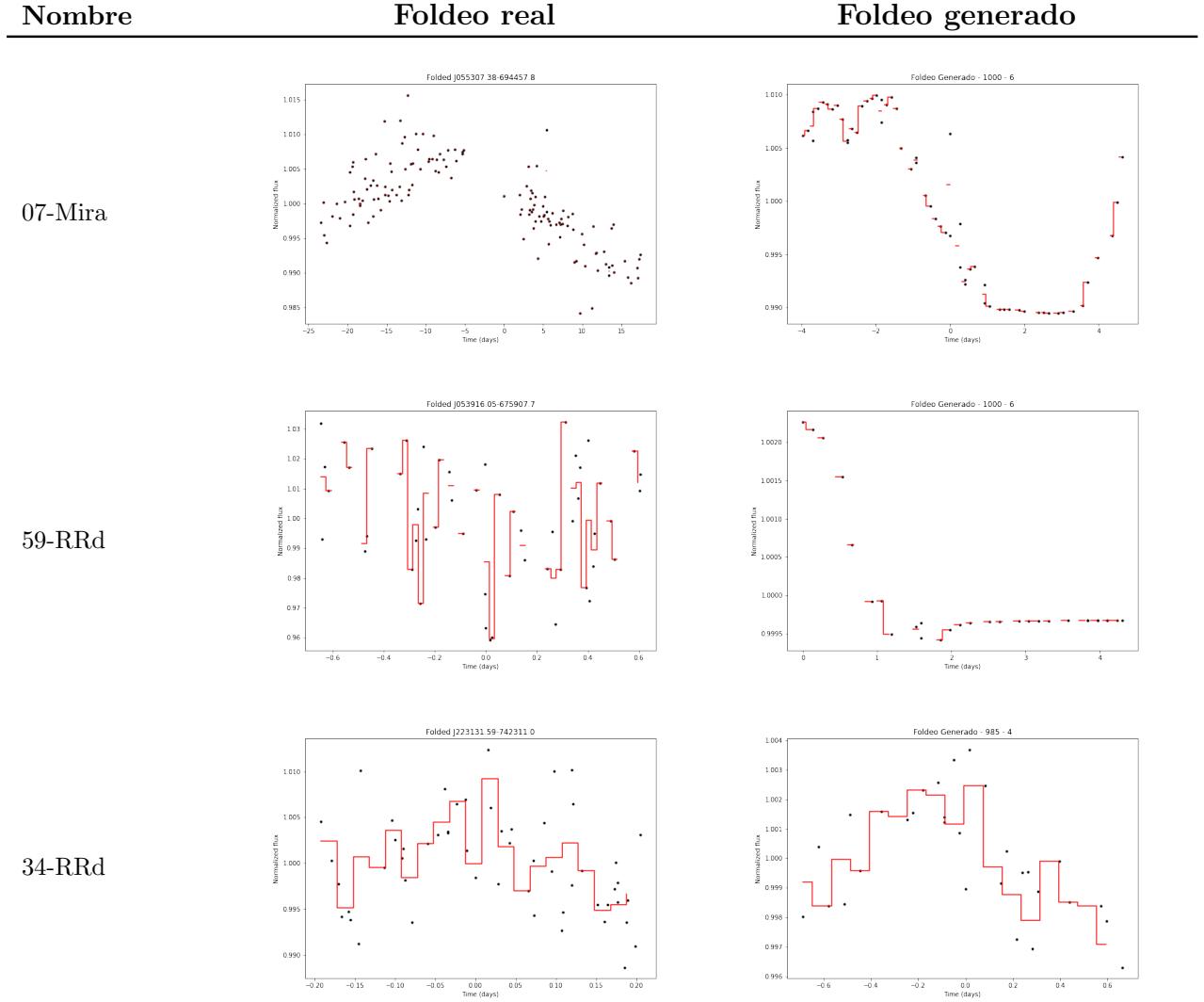


25-ELL



40-Mira





6.5. Comparación *IAR*

En esta sección, se van a comparar los resultados de *accuracy* de la red *GAN* de los tres dataset versus el *accuracy* obtenido del modelo de *iAR*. Dada la naturaleza de la librería, se optó por realizar dos experimentos para realizar las comparaciones. En el primer experimento, se implementó la librería sin modificar el proceso de generación de las series de tiempo, es decir, solo se aplicaron los procesamientos pertinentes a los datos, pero, se mantuvo la generación de series intacta. Y en el segundo, se forzó a generar series de tiempo que tuvieran un valor **mu** o promedio mayor a 1. Se definió este límite, ya que, cuando el *IAR* generaba series que no eran similares a las reales, ni de forma visual ni numérica, los parámetros **phi**, **mu** y **sigma** [11] resultaban tener valores muy cercanos a 0, por ende, para poder comparar los resultados con un caso ideal de series generadas, se decidió generar series de tiempo que tuvieran un valor de **mu** mayor a 1, provocando que el promedio de las magnitudes ahora fuera bien cercano a 15, el valor promedio de la magnitud de las series reales.

Entonces, las tablas comparativas de cada subsección a continuación, muestran los promedios de *accuracy* de la red *GAN* de los tres dataset v/s el *accuracy* obtenido del

modelo de *iAR* con el fin de comparar el rendimiento que tuvo la red *GAN* con un modelo previamente utilizado en otros trabajos y experimentos [11] [12].

Existen ciertos campos donde no hay valores descritos, por ejemplo, en la tabla 11, la clase **ED**, no presenta valores para los dataset de **GAIA** ni **WISE**, ya que, estos dataset no contienen dicha clase, por lo que se dejaron en blanco para visualizar de mejor manera los datos.

6.5.1. Sin forzar

Las tablas número 11 y 12, muestran los resultados del *accuracy* de las dos instancias ejecutadas por clase según el dataset, teniendo en cuenta que la primera tabla corresponde al generador de una capa, y la segunda al generador de dos capas. Por otro lado, en las tablas 13 y 14, con una y dos capas respectivamente, están los resultados de la métrica obtenidos con la fórmula (8) de cada instancia ejecutada.

La siguiente tabla, muestra el *accuracy* obtenido de cada instancia y dataset realizadas de una capa generadora. Por ejemplo, podemos ver la clase **DSCT_SXPHE**, que no existe en el dataset de **OGLE-III**, pero si en el de **GAIA** y **WISE**. De estos valores presentes, se puede apreciar que la primera instancia tuvo buenos resultados, tanto en el modelo de la *GAN* e *iAR*, cuando se ejecutó con los datos de **GAIA**, teniendo **0,589** y **0,612** respectivamente. Sin embargo, con el dataset de **WISE**, no se tuvieron resultados tan favorables, **0,851** y **0,833** respectivamente. Ahora, dada la cantidad de razones y variables que puedan afectar los resultados, la segunda instancia busca justamente nivelarlos, donde, efectivamente podemos ver que los resultados de la segunda instancia ahora resultaron ser mucho más similares y cercanos al valor 0,5. Siendo **0,525** y **0,462** del modelo *GAN* e *iAR* del dataset de **GAIA** respectivamente, y **0,412** y **0,442** para el dataset de **WISE**.

Tabla 11: Tabla con los resultados del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con una capa generadora sin forzar la generación de series del IAR.

Modelo	Clase	Accuracy			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED	0,484			0,484
IAR		0,538			0,538
GAN		0,393			0,393
IAR		0,528			0,528
GAN	ESD	0,589			0,589
IAR		0,466			0,466
GAN		0,396			0,396
IAR		0,417			0,417
GAN	EC	0,615			0,615
IAR		0,515			0,515

	GAN	0,605	0,605	
	IAR	0,214	0,214	
	GAN	0,336	0,571	0,578
	IAR	0,288	0,671	0,790
	GAN	0,833	0,401	0,819
	IAR	0,987	0,552	0,792
	GAN	0,035		0,035
	IAR	0,032		0,032
	GAN	0,491		0,491
	IAR	0,809		0,809
	GAN	0,210	0,891	0,780
	IAR	0,119	0,860	0,794
	GAN	0,342	0,252	0,520
	IAR	0,236	0,104	0,628
	GAN	0,588	0,569	0,583
	IAR	0,495	0,211	0,500
	GAN	0,913	0,840	0,794
	IAR	0,902	0,489	0,849
	GAN	0,513		0,513
	IAR	0,266		0,266
	GAN	0,860		0,860
	IAR	0,769		0,769
	GAN	0,454	0,090	0,272
	IAR	0,528	0,058	0,293
	GAN	0,860	0,769	0,814
	IAR	0,538	0,849	0,693
	GAN	0,421	0,470	0,446
	IAR	0,434	0,606	0,520
	GAN	0,573	0,473	0,523
	IAR	0,682	0,692	0,687
	GAN	0,300	0,768	0,534
	IAR	0,360	0,740	0,550
	GAN	0,553	0,232	0,392

IAR		0,478	0,174	0,326
GAN		0,761		0,761
IAR		0,808		0,808
GAN	MIRA_SR	0,510		0,510
IAR		0,553		0,553
GAN		0,407	0,502	0,455
IAR		0,193	0,452	0,322
GAN	RRD	0,626	0,412	0,519
IAR		0,377	0,213	0,295
GAN		0,589	0,851	0,720
IAR		0,612	0,833	0,723
GAN	DSCT_SXPHE	0,525	0,412	0,469
IAR		0,462	0,442	0,452
GAN		0,429	0,448	0,438
IAR		0,237	0,382	0,309
GAN	T2CEP	0,591	0,307	0,449
IAR		0,788	0,555	0,672
GAN			0,615	0,615
IAR			0,872	0,872
GAN	NonVar		0,592	0,592
IAR			0,314	0,314
GAN			0,488	0,488
IAR			0,193	0,193
GAN	NC		0,483	0,483
IAR			0,275	0,275
GAN			0,300	0,300
IAR			0,422	0,422
GAN	ELL		0,257	0,257
IAR			0,276	0,276

Analizando la misma clase anterior, ahora podemos ver los resultados con respecto al generador de dos capas. La primera instancia de la clase **DSCT_SXPHE** en el dataset de **GAIA**, presenta peores resultados que en el experimento de una capa, por lo menos en el modelo de la *GAN*, ya que el valor actual de **0,861** resulta mucho mayor que el valor anterior de **0,589**. Sin embargo, con el modelo de *iAR*, estos valores son bien similares,

siendo **0,612** y **0,644** con un generador de una capa y dos capas respectivamente. Esta similitud se debe a que el modelo *iAR* no presenta cambios en su estructura, mientras que la *GAN*, justamente se le modificó la cantidad de capas ocultas en su Generador, pasando de una a dos capa, explicando así el cambio que tuvieron los resultados.

Tabla 12: Tabla con los resultados del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras sin forzar la generación de series del IAR.

Modelo	Clase	Accuracy			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED		0,763		0,763
IAR			0,834		0,834
GAN			0,694		0,694
IAR			0,810		0,810
GAN	ESD		0,406		0,406
IAR			0,012		0,012
GAN			0,305		0,305
IAR			0,405		0,405
GAN	EC		0,054		0,054
IAR			0,036		0,036
GAN			0,872		0,872
IAR			0,869		0,869
GAN	cep	0,400	0,493	0,615	0,503
IAR		0,016	0,408	0,747	0,391
GAN		0,259	0,459	0,594	0,437
IAR		0,332	0,321	0,337	0,330
GAN	dsct		0,449		0,449
IAR			0,714		0,714
GAN			0,752		0,752
IAR			0,745		0,745
GAN	Rrab	0,562	0,092	0,869	0,508
IAR		0,360	0,082	0,477	0,306
GAN		0,556	0,136	0,445	0,379
IAR		0,745	0,507	0,681	0,644
GAN		0,229	0,278	0,457	0,321

IAR		0,472	0,215	0,722	0,470
GAN		0,477	0,626	0,136	0,413
IAR		0,533	0,577	0,105	0,405
GAN		0,399			0,399
IAR		0,455			0,455
GAN	std	0,498			0,498
IAR		0,141			0,141
GAN		0,250		0,362	0,306
IAR		0,254		0,769	0,512
GAN	OSARG	0,409		0,204	0,306
IAR		0,511		0,638	0,575
GAN		0,655		0,267	0,461
IAR		0,615		0,525	0,570
GAN	SRV	0,507		0,507	0,507
IAR		0,482		0,530	0,506
GAN		0,579		0,481	0,530
IAR		0,572		0,382	0,477
GAN	MIRA	0,349		0,702	0,525
IAR		0,300		0,696	0,498
GAN			0,952		0,952
IAR			0,958		0,958
GAN	MIRA_SR		0,563		0,563
IAR			0,780		0,780
GAN			0,322	0,277	0,300
IAR			0,450	0,363	0,407
GAN	RRD		0,455	0,242	0,348
IAR			0,660	0,477	0,568
GAN			0,861	0,347	0,604
IAR			0,644	0,562	0,603
GAN	DSCT_SXPHE		0,546	0,704	0,625
IAR			0,897	0,767	0,832
GAN			0,529	0,472	0,501
IAR	T2CEP		0,504	0,367	0,435

GAN		0,724	0,489	0,606
IAR		0,830	0,346	0,588
GAN		0,349	0,349	
IAR		0,622	0,622	
GAN	NonVar	0,865	0,865	
IAR		0,849	0,849	
GAN		0,780	0,780	
IAR		0,771	0,771	
GAN	NC	0,508	0,508	
IAR		0,371	0,371	
GAN		0,370	0,370	
IAR		0,628	0,628	
GAN	ELL	0,598	0,598	
IAR		0,524	0,524	

Recordando la métrica propuesta para conocer la similitud de las series generadas y reales, denotadas por la fórmula número (8), podemos obtener las siguientes tablas, 13 y 14 para una y dos capas respectivamente. Estas tablas, permiten realizar un mejor análisis sobre los valores de *accuracy* que se obtuvieron. Por ejemplo, al seguir con la clase **DSCT_SXPHE**, en el experimento de una capa, podemos identificar que el modelo GAN tuvo un mejor rendimiento que el modelo de *iAR* en el dataset de **GAIA**, debido a que los valores obtenidas por la métrica de la *GAN* son menores que del *iAR*, siendo, **0,089** v/s **0,112** en la primera instancia respectivamente.

Tabla 13: Tabla con la métrica del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con una capa generadora sin forzar la generación de series del IAR.

Modelo	Clase	Métrica			Promedio
		OGLE-III	GAIA	WISE	
GAN		0,016			0,016
IAR		0,038			0,038
GAN	ED	0,107			0,107
IAR		0,028			0,028
GAN		0,089			0,089
IAR		0,034			0,034
GAN	ESD	0,104			0,104
IAR		0,083			0,083

GAN		0,115		0,115
IAR		0,015		0,015
GAN	EC	0,105		0,105
IAR		0,286		0,286
GAN		0,164	0,071	0,078
IAR		0,212	0,171	0,290
GAN	cep	0,333	0,099	0,319
IAR		0,487	0,052	0,292
GAN		0,465		0,465
IAR	dsct	0,468		0,468
GAN		0,009		0,009
IAR		0,309		0,309
GAN		0,290	0,391	0,280
IAR	Rrab	0,381	0,360	0,294
GAN		0,158	0,248	0,020
IAR		0,264	0,396	0,128
GAN		0,088	0,069	0,083
IAR	RRc	0,005	0,289	0,000
GAN		0,413	0,340	0,294
IAR		0,402	0,011	0,349
GAN		0,013		0,013
IAR	std	0,234		0,234
GAN		0,360		0,360
IAR		0,269		0,269
GAN		0,046		0,411
IAR	OSARG	0,028		0,442
GAN		0,360		0,269
IAR		0,038		0,314
GAN		0,079		0,193
IAR	SRV	0,066		0,054
GAN		0,073		0,106
IAR		0,182		0,086
GAN		0,201		0,050
GAN		0,268		0,187
GAN		0,234		0,234

IAR		0,140	0,240	0,190
GAN		0,053	0,268	0,161
IAR		0,022	0,326	0,174
GAN		0,261		0,261
IAR	MIRA_SR	0,308		0,308
GAN		0,010		0,010
IAR		0,053		0,053
GAN		0,093	0,002	0,047
IAR	RRD	0,307	0,048	0,178
GAN		0,126	0,088	0,107
IAR		0,123	0,287	0,205
GAN		0,089	0,351	0,220
IAR	DSCT_SXPHE	0,112	0,333	0,223
GAN		0,025	0,088	0,056
IAR		0,038	0,058	0,048
GAN		0,071	0,052	0,062
IAR	T2CEP	0,263	0,118	0,191
GAN		0,091	0,193	0,142
IAR		0,288	0,055	0,172
GAN			0,115	0,115
IAR	NonVar		0,372	0,372
GAN			0,092	0,092
IAR			0,186	0,186
GAN			0,013	0,013
IAR	NC		0,307	0,307
GAN			0,017	0,017
IAR			0,225	0,225
GAN			0,200	0,200
IAR	ELL		0,078	0,078
GAN			0,243	0,243
IAR			0,224	0,224

Analizando el experimento con dos capas de la misma clase y dataset mencionados, podemos ver un incremento en los valores de la métrica, teniendo ahora, **0,361** y **0,144** para la primera instancia de *GAN* e *iAR*. En cambio, anteriormente obtuvimos los valores

de **0,089** y **0,112**. Se puede notar que la *GAN* tuvo peor rendimiento, pero, el modelo *iAR* mantuvo uno similar. Esto se debe a que entre estas iteraciones, el modelo *iAR* no fue modificado, ya que se mantuvo el parámetro de no forzar la generación de series.

Tabla 14: Tabla con la métrica del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras sin forzar la generación de series del IAR.

Modelo	Clase	Métrica			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED		0,263		0,263
IAR			0,334		0,334
GAN			0,194		0,194
IAR			0,310		0,310
GAN	ESD		0,095		0,095
IAR			0,488		0,488
GAN			0,195		0,195
IAR			0,095		0,095
GAN	EC		0,447		0,447
IAR			0,464		0,464
GAN			0,372		0,372
IAR			0,369		0,369
GAN	cep	0,100	0,007	0,115	0,074
IAR		0,484	0,092	0,247	0,274
GAN		0,241	0,041	0,094	0,125
IAR		0,168	0,179	0,163	0,170
GAN	dsct		0,052		0,052
IAR			0,214		0,214
GAN			0,252		0,252
IAR			0,245		0,245
GAN	Rrab	0,062	0,408	0,369	0,280
IAR		0,140	0,418	0,023	0,194
GAN		0,056	0,364	0,055	0,158
IAR		0,245	0,007	0,181	0,144
GAN	RRc		0,271	0,223	0,043
IAR			0,028	0,285	0,222
					0,178

	GAN	0,023	0,126	0,364	0,171
	IAR	0,033	0,077	0,395	0,168
	GAN	0,101			0,101
	IAR	0,045			0,045
	GAN	0,002			0,002
	IAR	0,359			0,359
	GAN	0,250		0,138	0,194
	IAR	0,246		0,269	0,258
	GAN	0,091		0,296	0,194
	IAR	0,011		0,138	0,075
	GAN	0,155		0,233	0,194
	IAR	0,115		0,025	0,070
	GAN	0,007		0,007	0,007
	IAR	0,018		0,030	0,024
	GAN	0,079		0,019	0,049
	IAR	0,072		0,118	0,095
	GAN	0,152		0,202	0,177
	IAR	0,200		0,196	0,198
	GAN		0,452		0,452
	IAR		0,458		0,458
	GAN		0,063		0,063
	IAR		0,280		0,280
	GAN		0,178	0,223	0,201
	IAR		0,050	0,137	0,093
	GAN		0,045	0,258	0,152
	IAR		0,160	0,023	0,092
	GAN		0,361	0,153	0,257
	IAR		0,144	0,062	0,103
	GAN		0,046	0,204	0,125
	IAR		0,397	0,267	0,332
	GAN		0,029	0,028	0,028
	IAR		0,004	0,133	0,068
	GAN		0,224	0,011	0,117

IAR		0,330	0,154	0,242
GAN		0,152	0,152	
IAR		0,122	0,122	
GAN	NonVar	0,365	0,365	
IAR		0,349	0,349	
GAN		0,280	0,280	
IAR		0,271	0,271	
GAN	NC	0,008	0,008	
IAR		0,129	0,129	
GAN		0,130	0,130	
IAR		0,128	0,128	
GAN	ELL	0,098	0,098	
IAR		0,024	0,024	

En la siguiente tabla, se muestran los valores promedio de las métricas obtenidas por cada clase del experimento de una capa, según el dataset utilizado. Adicionalmente, se agregaron las desviaciones estándar de cada una para complementar los resultados.

Para el caso de la clase **DSCT_SXPHE**, se puede observar que el modelo de la *GAN* tuvo mejor rendimiento que el *iAR* en los dos dataset, teniendo un promedio y desviación estándar de **0,057** con **0,032** y **0,219** con **0,132** para **GAIA** y **WISE** respectivamente. En ciertos casos, como este, se puede apreciar que la desviación estándar es muy elevada, correspondiendo a un 56 % del promedio. En efecto, estos valores resultan ser elevados debido a las variaciones que ocurren entre las instancias de una misma clase. Si retomamos los valores de las métricas, tenemos **0,089** y **0,025**, dada la diferencia entre ellas, se obtiene un valor elevado de la desviación estándar.

Tabla 15: Tabla con el promedio de métricas y desviaciones estándar de las clases según los datasets, obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con una capa generadora sin forzar la generación de series del IAR.

Modelo	Clase	OGLE-III		GAIA		WISE	
		AVG métrica	STD métrica	AVG métrica	STD métrica	AVG métrica	STD métrica
GAN	ED	0,062	0,046				
IAR		0,033	0,005				
GAN	ESD	0,097	0,008				
IAR		0,059	0,025				
GAN	EC	0,110	0,005				
IAR		0,151	0,136				
GAN	cep	0,248	0,084	0,085	0,014	0,199	0,120
IAR		0,349	0,138	0,111	0,059	0,291	0,001
GAN	dsct	0,237	0,228				
IAR		0,389	0,079				
GAN	Rrab	0,224	0,066	0,319	0,071	0,150	0,130
IAR		0,322	0,059	0,378	0,018	0,211	0,083
GAN	RRc	0,250	0,162	0,204	0,135	0,189	0,106
IAR		0,203	0,199	0,150	0,139	0,175	0,174
GAN	std	0,187	0,173				
IAR		0,251	0,018				
GAN	OSARG	0,203	0,157			0,340	0,071
IAR		0,033	0,005			0,396	0,047

	GAN		0,076	0,003			0,028	0,001
	IAR	SRV	0,124	0,058			0,149	0,043
	GAN		0,127	0,074			0,268	0,000
	IAR	MIRA	0,081	0,059			0,283	0,043
	GAN				0,136	0,126		
	IAR	MIRA_SR			0,180	0,127		
	GAN				0,109	0,017	0,045	0,043
	IAR	RRD			0,215	0,092	0,168	0,119
	GAN				0,057	0,032	0,219	0,132
	IAR	DSCT_SXPHE			0,075	0,037	0,195	0,138
	GAN				0,081	0,010	0,123	0,071
	IAR	T2CEP			0,276	0,013	0,087	0,032
	GAN						0,103	0,011
	IAR	NonVar					0,279	0,093
	GAN						0,015	0,002
	IAR	NC					0,266	0,041
	GAN						0,221	0,021
	IAR	ELL					0,151	0,073

Finalizando con la clase **DSCT_SXPHE**, podemos ver que ahora en el experimento de dos capas generadoras, la desviación estándar es mucho mayor, debido a que existe una variación mayor entre las métricas de sus instancias. Ahora, de un promedio de **0,203**, se obtiene una desviación estándar de **0,158**, que corresponde a un 78 % del promedio.

Tabla 16: Tabla con el promedio de métricas y desviaciones estándar de las clases según los datasets, obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras sin forzar la generación de series del IAR.

Modelo	Clase	OGLE-III		GAIA		WISE	
		AVG métrica	STD métrica	AVG métrica	STD métrica	AVG métrica	STD métrica
GAN	ED	0,228	0,034				
IAR		0,322	0,012				
GAN	ESD	0,145	0,050				
IAR		0,291	0,197				
GAN	EC	0,409	0,037				
IAR		0,417	0,048				
GAN	cep	0,171	0,071	0,024	0,017	0,104	0,010
IAR		0,326	0,158	0,136	0,044	0,205	0,042
GAN	dsct	0,152	0,100				
IAR		0,230	0,016				
GAN	Rrab	0,059	0,003	0,386	0,022	0,212	0,157
IAR		0,192	0,052	0,213	0,205	0,102	0,079
GAN	RRc	0,147	0,124	0,174	0,048	0,203	0,161
IAR		0,031	0,002	0,181	0,104	0,308	0,086
GAN	std	0,052	0,050				
IAR		0,202	0,157				
GAN	OSARG	0,170	0,079			0,217	0,079
IAR		0,129	0,117			0,204	0,066

	GAN		0,081	0,074			0,120	0,113
	IAR	SRV	0,067	0,049			0,027	0,003
	GAN		0,115	0,036			0,111	0,092
	IAR	MIRA	0,136	0,064			0,157	0,039
	GAN				0,258	0,194		
	IAR	MIRA_SR			0,369	0,089		
	GAN				0,111	0,066	0,241	0,018
	IAR	RRD			0,105	0,055	0,080	0,057
	GAN				0,203	0,158	0,178	0,025
	IAR	DSCT_SXPHE			0,271	0,127	0,165	0,103
	GAN				0,126	0,098	0,019	0,008
	IAR	T2CEP			0,167	0,163	0,144	0,010
	GAN						0,258	0,107
	IAR	NonVar					0,236	0,114
	GAN						0,144	0,136
	IAR	NC					0,200	0,071
	GAN						0,114	0,016
	IAR	ELL					0,076	0,052

A fin de cuentas, podemos apreciar la siguiente tabla que muestra el promedio de las métricas obtenidas por de cada dataset y definidas por su modelo. Por ejemplo, el modelo de *iAR* fue bastante parejo a lo largo de los tres dataset, mientras que el modelo de la *GAN*, fue bastante más variado, teniendo desde un promedio de **0,142** con una capa, hasta un máximo de **0,182** en el experimento de dos capas generadoras, y ambas, por coincidencia, en el dataset de **GAIA**. La última columna, **GAN Ideal**, corresponde al promedio de las mejores instancias seleccionadas de la *GAN*, es decir, si por cada clase, se selecciona el modelo con mejor rendimiento (*GAN* de una o dos capas generadoras), se obtendría en promedio, los valores detallados de la tabla. Lógicamente, el promedio de los mejores casos debiese ser menor que los resultados obtenidos en realidad, como es el caso del valor **0,115**, que resulta ser menor que el modelo del *iAR*, y los resultados reales del modelo *GAN*.

Tabla 17: Promedio de las métricas por dataset

Dataset	AVG iAR	GAN 1 Capa	GAN 2 Capa	AVG GAN	GAN Ideal
OGLE	0,197	0,165	0,157	0,161	0,110
GAIA	0,202	0,142	0,183	0,162	0,129
WISE	0,190	0,158	0,160	0,159	0,106
Promedio	0,196	0,155	0,167	0,161	0,115

6.5.2. Con forzar

Las tablas número 18 y 19, muestran los resultados del *accuracy* de las dos instancias ejecutadas por clase según el dataset, teniendo en cuenta que la primera tabla corresponde al generador de una capa, y la segunda al generador de dos capas. Por otro lado, en las tablas 20 y 21, con una y dos capas respectivamente, están los resultados de la métrica obtenidos con la fórmula (8) de cada instancia ejecutada.

Analizando la siguiente tabla, de la primera instancia correspondiente a la clase **RRc**, se puede ver que tuvo excelentes resultados en función del *accuracy*. Ya que, el modelo de la *GAN* obtuvo un valor de **0,588**, mientras que el valor del *iAR* fue de **0,527**. Esta instancia corresponde a la **02-RRc** de la tabla 5, donde se puede ver que el foldeo generado no es tan similar al real, ya que puede tener datos generados correctamente, los cuales, no necesariamente implican que va a tener un buen foldeo.

Tabla 18: Tabla con los resultados del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la *GAN* con una capa generadora forzando la generación de series del IAR.

Modelo	Clase	Accuracy			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED				0,484
					0,515

GAN		0,393		0,393
IAR		0,608		0,608
GAN		0,589		0,589
IAR		0,750		0,750
GAN	ESD	0,396		0,396
IAR		0,398		0,398
GAN		0,615		0,615
IAR	EC	0,157		0,157
GAN		0,605		0,605
IAR		0,299		0,299
GAN		0,336	0,571	0,578
IAR	cep	0,268	0,575	0,698
GAN		0,833	0,401	0,819
IAR		0,986	0,338	0,747
GAN		0,035		0,035
IAR	dsct	0,029		0,029
GAN		0,491		0,491
IAR		0,757		0,757
GAN		0,210	0,891	0,780
IAR	Rrab	0,089	0,858	0,748
GAN		0,342	0,252	0,520
IAR		0,219	0,189	0,482
GAN		0,588	0,569	0,583
IAR	RRc	0,527	0,357	0,573
GAN		0,913	0,840	0,794
IAR		0,858	0,761	0,809
GAN		0,513		0,513
IAR	std	0,438		0,438
GAN		0,860		0,860
IAR		0,699		0,699
GAN		0,454	0,090	0,272
IAR	OSARG	0,297	0,079	0,188
GAN		0,860	0,769	0,814

IAR		0,743	0,823	0,783
GAN		0,421	0,470	0,446
IAR	SRV	0,699	0,661	0,680
GAN		0,573	0,473	0,523
IAR		0,479	0,446	0,462
GAN		0,300	0,768	0,534
IAR	MIRA	0,369	0,474	0,422
GAN		0,553	0,232	0,392
IAR		0,464	0,148	0,306
GAN		0,761		0,761
IAR	MIRA_SR	0,771		0,771
GAN		0,510		0,510
IAR		0,460		0,460
GAN		0,407	0,502	0,455
IAR	RRD	0,197	0,542	0,369
GAN		0,626	0,412	0,519
IAR		0,592	0,227	0,409
GAN		0,589	0,851	0,720
IAR	DSCT_SXPHE	0,582	0,858	0,720
GAN		0,525	0,412	0,469
IAR		0,685	0,634	0,659
GAN		0,429	0,448	0,438
IAR	T2CEP	0,438	0,443	0,441
GAN		0,591	0,307	0,449
IAR		0,746	0,606	0,676
GAN			0,615	0,615
IAR	NonVar		0,862	0,862
GAN			0,592	0,592
IAR			0,500	0,500
GAN			0,488	0,488
IAR	NC		0,164	0,164
GAN			0,483	0,483
IAR			0,193	0,193

GAN		0,300	0,300
IAR		0,277	0,277
GAN	ELL	0,257	0,257
IAR		0,273	0,273

La siguiente tabla muestra el mismo detalle anterior, pero con la diferencia de tener el generador con dos capas. Esto resulta en un modelo distinto, y por consiguiente, resultados distintos, aunque, no necesariamente dichos resultados deban estar tan alejados entre si. Por ejemplo, si se analiza la primera instancia de la clase **cep** del modelo de la *GAN*, podemos ver que a lo largo de los tres dataset obtuvo un promedio de **0,503**, mientras que la instancia homóloga con un generador de una capa, tuvo un promedio de **0,495**. Aunque la diferencia entre ellas sea muy pequeña, el mejor resultado sería el de **0,503** por estar más cercano al valor 0,5.

En contraste, la primera instancia de la clase **RRD** del modelo de una capa, tuvo un promedio de **0,455** y **0,369** para *GAN* e *iAR* respectivamente. Mientras que la primera instancia en el modelo de dos capas, obtuvo un promedio de **0,300** y **0,272** para *GAN* e *iAR* respectivamente. Entonces, por lo visto, la instancia de dos capas resultó tener peores resultados que la de una capa.

Tabla 19: Tabla con los resultados del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras forzando la generación de series del IAR.

Modelo	Clase	Accuracy			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED	0,763			0,763
IAR		0,750			0,750
GAN	ESD	0,694			0,694
IAR		0,732			0,732
GAN	EC	0,406			0,406
IAR		0,017			0,017
GAN	cep	0,305			0,305
IAR		0,471			0,471
GAN		0,054			0,054
IAR		0,056			0,056
GAN		0,872			0,872
IAR		0,871			0,871
GAN		0,400	0,493	0,615	0,503
IAR		0,014	0,569	0,543	0,375
GAN		0,259	0,459	0,594	0,437

IAR		0,266	0,476	0,498	0,413
GAN		0,449			0,449
IAR		0,856			0,856
GAN	dsct	0,752			0,752
IAR		0,759			0,759
GAN		0,562	0,092	0,869	0,508
IAR	Rrab	0,334	0,081	0,859	0,425
GAN		0,556	0,136	0,445	0,379
IAR		0,673	0,143	0,493	0,436
GAN		0,229	0,278	0,457	0,321
IAR	RRc	0,380	0,152	0,565	0,365
GAN		0,477	0,626	0,136	0,413
IAR		0,405	0,516	0,182	0,368
GAN		0,399			0,399
IAR	std	0,247			0,247
GAN		0,498			0,498
IAR		0,065			0,065
GAN	OSARG	0,250		0,362	0,306
IAR		0,357		0,517	0,437
GAN		0,409		0,204	0,306
IAR		0,802		0,321	0,562
GAN		0,655		0,267	0,461
IAR	SRV	0,425		0,495	0,460
GAN		0,507		0,507	0,507
IAR		0,473		0,613	0,543
GAN		0,579		0,481	0,530
IAR	MIRA	0,506		0,420	0,463
GAN		0,349		0,702	0,525
IAR		0,231		0,672	0,451
GAN			0,952		0,952
IAR	MIRA_SR		0,956		0,956
GAN			0,563		0,563

IAR		0,544	0,544
GAN		0,322	0,277
IAR		0,265	0,280
GAN	RRD	0,455	0,242
IAR		0,520	0,302
GAN		0,861	0,347
IAR	DSCT_SXPHE	0,839	0,391
GAN		0,546	0,704
IAR		0,681	0,696
GAN		0,529	0,472
IAR	T2CEP	0,495	0,458
GAN		0,724	0,489
IAR		0,765	0,447
GAN		0,349	0,349
IAR	NonVar	0,359	0,359
GAN		0,865	0,865
IAR		0,785	0,785
GAN		0,780	0,780
IAR	NC	0,765	0,765
GAN		0,508	0,508
IAR		0,513	0,513
GAN		0,370	0,370
IAR	ELL	0,356	0,356
GAN		0,598	0,598
IAR		0,584	0,584

Siguiendo los ejemplos anteriores, la siguiente tabla mostrará los valores obtenidos a partir de la métrica definida de los modelos con una capa generadora. La clase **cep**, presenta un valor de la métrica de **0,164** en la primera instancia correspondiente al dataset de **OGLE-III**. Dicha instancia, corresponde a la que tuvo mejor rendimiento para el dataset, debido a que es la que presenta un valor más cercano a 0, lo que implica, que es la instancia que logra tener la mayor similitud entre sus series reales y generadas. En contraste, la otra instancia de la misma clase, presenta una métrica de **0,333**, bastante mayor a la primera, lo que significa que dicha instancia tuvo un mal rendimiento teniendo series de tiempo muy distintas.

Tabla 20: Tabla con la métrica del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con una capa generadora forzando la generación de series del IAR.

Modelo	Clase	Métrica			Promedio
		OGLE-III	GAIA	WISE	
GAN	ED		0,016		0,016
IAR			0,015		0,015
GAN	ESD		0,107		0,107
IAR			0,108		0,108
GAN	EC		0,089		0,089
IAR			0,250		0,250
GAN	cep		0,104		0,104
IAR			0,102		0,102
GAN	EC		0,115		0,115
IAR			0,343		0,343
GAN	cep		0,105		0,105
IAR			0,201		0,201
GAN	dsct	0,164	0,071	0,078	0,104
IAR		0,232	0,075	0,198	0,168
GAN	dsct	0,333	0,099	0,319	0,250
IAR		0,486	0,162	0,247	0,298
GAN	Rrab	0,465			0,465
IAR		0,471			0,471
GAN	Rrb	0,009			0,009
IAR		0,257			0,257
GAN	RRc	0,290	0,391	0,280	0,320
IAR		0,411	0,358	0,248	0,339
GAN	RRc	0,158	0,248	0,020	0,142
IAR		0,281	0,311	0,018	0,203
GAN	RRc	0,088	0,069	0,083	0,080
IAR		0,027	0,143	0,073	0,081
GAN	RRc	0,413	0,340	0,294	0,349
IAR		0,358	0,261	0,309	0,309
GAN		0,013			0,013

IAR		0,062	0,062
GAN		0,360	0,360
IAR		0,199	0,199
GAN		0,046	0,411
IAR	OSARG	0,203	0,421
GAN		0,360	0,269
IAR		0,243	0,323
GAN		0,079	0,030
IAR	SRV	0,199	0,161
GAN		0,073	0,027
IAR		0,021	0,054
GAN		0,201	0,268
IAR	MIRA	0,131	0,026
GAN		0,053	0,268
IAR		0,036	0,352
GAN			0,194
IAR	MIRA_SR	0,261	0,261
GAN		0,271	0,271
IAR		0,010	0,010
GAN		0,040	0,040
IAR	RRD	0,093	0,002
GAN		0,303	0,042
IAR		0,126	0,088
GAN		0,092	0,273
IAR	DSCT_SXPHE	0,089	0,351
GAN		0,082	0,358
IAR		0,025	0,088
GAN		0,185	0,134
IAR	T2CEP	0,071	0,052
GAN		0,062	0,057
IAR		0,091	0,193
GAN		0,246	0,106
IAR	NonVar	0,115	0,115
GAN		0,362	0,362

GAN		0,092	0,092
IAR		0,000	0,000
GAN		0,013	0,013
IAR		0,336	0,336
GAN	NC	0,017	0,017
IAR		0,307	0,307
GAN		0,200	0,200
IAR		0,223	0,223
GAN	ELL	0,243	0,243
IAR		0,227	0,227

Observando nuevamente la clase **RRD**, podemos ver que entre el modelo de una y dos capas, se obtiene un mejor rendimiento con el primero, ya que, en promedio, las primeras instancias de una capa tienen una métrica de **0,047** y **0,107**, mientras que las de dos capas tienen **0,201** y **0,152** para la primera y segunda instancia respectivamente. Adicionalmente, se debe destacar el rendimiento que se obtuvo en la primera instancia teniendo una capa generadora, ya que, se obtuvo un valor de la métrica de **0,002**, lo que corresponde a un valor muy bajo que esta casi en el valor ideal.

Existe otra clase que obtuvo valores ideales en su rendimiento, teniendo una métrica de **0,000**, es decir, logró obtener un *accuracy* directamente de **0,5** aproximadamente. Esta clase, corresponde a la **NonVar**, la cual, en su segunda instancia de *iAR*, logró tener dichos valores. Incluso, la segunda instancia del modelo *GAN*, también tuvo excelentes resultados, teniendo una métrica de **0,092**.

Tabla 21: Tabla con la métrica del *accuracy* obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras forzando la generación de series del IAR.

Modelo	Clase	Métrica			Promedio
		OGLE-III	GAIA	WISE	
GAN		0,263			0,263
IAR		0,250			0,250
GAN	ED	0,194			0,194
IAR		0,232			0,232
GAN		0,095			0,095
IAR		0,483			0,483
GAN	ESD	0,195			0,195
IAR		0,029			0,029
GAN		0,447			0,447

IAR		0,444		0,444
GAN		0,372		0,372
IAR		0,371		0,371
GAN		0,100	0,007	0,115
IAR		0,486	0,069	0,043
GAN	cep	0,241	0,041	0,094
IAR		0,234	0,024	0,002
GAN		0,052		0,052
IAR	dsct	0,356		0,356
GAN		0,252		0,252
IAR		0,259		0,259
GAN		0,062	0,408	0,369
IAR	Rrab	0,166	0,419	0,359
GAN		0,056	0,364	0,055
IAR		0,173	0,357	0,007
GAN		0,271	0,223	0,043
IAR	RRc	0,120	0,348	0,065
GAN		0,023	0,126	0,364
IAR		0,095	0,016	0,318
GAN		0,101		0,101
IAR	std	0,253		0,253
GAN		0,002		0,002
IAR		0,435		0,435
GAN		0,250		0,138
IAR	OSARG	0,143		0,017
GAN		0,091		0,296
IAR		0,302		0,240
GAN		0,155		0,233
IAR		0,075		0,005
GAN	SRV	0,007		0,007
IAR		0,027		0,113
GAN		0,079		0,019
IAR	MIRA	0,006		0,080
				0,043

	GAN	0,152	0,202	0,177
	IAR	0,269	0,172	0,221
	GAN	0,452	0,452	
	IAR	0,456	0,456	
MIRA_SR	GAN	0,063	0,063	
	IAR	0,044	0,044	
	GAN	0,178	0,223	0,201
RRD	IAR	0,235	0,220	0,228
	GAN	0,045	0,258	0,152
	IAR	0,020	0,198	0,109
DSCT_SXPHE	GAN	0,361	0,153	0,257
	IAR	0,339	0,109	0,224
	GAN	0,046	0,204	0,125
	IAR	0,181	0,196	0,188
T2CEP	GAN	0,029	0,028	0,028
	IAR	0,005	0,042	0,024
	GAN	0,224	0,011	0,117
	IAR	0,265	0,053	0,159
NonVar	GAN		0,152	0,152
	IAR		0,141	0,141
	GAN		0,365	0,365
	IAR		0,285	0,285
NC	GAN		0,280	0,280
	IAR		0,265	0,265
	GAN		0,008	0,008
	IAR		0,013	0,013
ELL	GAN		0,130	0,130
	IAR		0,144	0,144
	GAN		0,098	0,098
	IAR		0,084	0,084

En la siguiente tabla, se muestran los valores promedio de las métricas obtenidas por cada clase del experimento de una capa, según el dataset utilizado. Adicionalmente, se agregaron las desviaciones estándar de cada una para complementar los resultados.

Para este caso, de la clase **NonVar**, se puede observar que el modelo de la *GAN* tuvo mejor rendimiento que el *iAR* en el dataset de **WISE**, teniendo un promedio y desviación estándar de **0,103** con **0,011** respectivamente. Para este caso la desviación estándar no es muy elevada con respecto al promedio de la métrica, ya que corresponde a un **10%** aproximadamente del promedio. En cambio, si analizamos el *iAR* para la misma clase, podemos ver que su promedio de métrica es de **0,181**, pero, su desviación estándar es de **0,181**. Estos valores son idénticos, porque una instancia de esta clase tuvo un *accuracy* de **0,5**, provocando que la métrica fuera de **0,000**, haciendo que la desviación entre las dos instancias tuviera el valor de la métrica de la primera instancia.

Tabla 22: Tabla con el promedio de métricas y desviaciones estándar de las clases según los datasets, obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con una capa generadora forzando la generación de series del IAR.

Modelo	Clase	OGLE-III		GAIA		WISE	
		AVG de la métrica	STD métrica	AVG métrica	STD métrica	AVG métrica	STD métrica
GAN	ED	0,062	0,046				
IAR		0,061	0,046				
GAN	ESD	0,097	0,008				
IAR		0,176	0,074				
GAN	EC	0,110	0,005				
IAR		0,272	0,071				
GAN	cep	0,248	0,084	0,085	0,014	0,199	0,120
IAR		0,359	0,127	0,118	0,043	0,222	0,024
GAN	dsct	0,237	0,228				
IAR		0,364	0,107				
GAN	Rrab	0,224	0,066	0,319	0,071	0,150	0,130
IAR		0,346	0,065	0,334	0,024	0,133	0,115
GAN	RRc	0,250	0,162	0,204	0,135	0,189	0,106
IAR		0,192	0,165	0,202	0,059	0,191	0,118
GAN	std	0,187	0,173				
IAR		0,130	0,068				
GAN	OSARG	0,203	0,157			0,340	0,071
IAR		0,223	0,020			0,372	0,049

	GAN		0,076	0,003			0,028	0,001
	IAR	SRV	0,110	0,089			0,108	0,054
	GAN		0,127	0,074			0,268	0,000
	IAR	MIRA	0,083	0,048			0,189	0,163
	GAN				0,136	0,126		
	IAR	MIRA_SR			0,155	0,115		
	GAN				0,109	0,017	0,045	0,043
	IAR	RRD			0,197	0,106	0,158	0,116
	GAN				0,057	0,032	0,219	0,132
	IAR	DSCT_SXPHE			0,134	0,052	0,246	0,112
	GAN				0,081	0,010	0,123	0,071
	IAR	T2CEP			0,154	0,092	0,081	0,025
	GAN						0,103	0,011
	IAR	NonVar					0,181	0,181
	GAN						0,015	0,002
	IAR	NC					0,321	0,015
	GAN						0,221	0,021
	IAR	ELL					0,225	0,002

Analizando los promedios y desviaciones estándar de los modelos con dos capas, podemos apreciar la clase **cep**, la cual, en la instancia de *iAR*, tiene un promedio de **0,023** y una desviación estándar de **0,020**. Esta desviación estándar equivale a un **88%** del promedio, un valor bastante alto. Esto se debe, a que la segunda instancia de la clase obtuvo una métrica bastante baja de **0,002**, mientras que la primera fue de **0,043**. Entonces, dado el valor tan bajo de la instancia, causa que la desviación sea casi igual al promedio de ellas.

Tabla 23: Tabla con el promedio de métricas y desviaciones estándar de las clases según los datasets, obtenidos por el modelo LSTM Score de series reales v/s falsas. Se utilizó la GAN con dos capas generadoras forzando la generación de series del IAR.

Modelo	Clase	OGLE-III		GAIA		WISE	
		AVG métrica	STD métrica	AVG métrica	STD métrica	AVG métrica	STD métrica
GAN	ED	0,228	0,034				
IAR		0,241	0,009				
GAN	ESD	0,145	0,050				
IAR		0,256	0,227				
GAN	EC	0,409	0,037				
IAR		0,408	0,036				
GAN	cep	0,171	0,071	0,024	0,017	0,104	0,010
IAR		0,360	0,126	0,046	0,023	0,023	0,020
GAN	dsct	0,152	0,100				
IAR		0,307	0,048				
GAN	Rrab	0,059	0,003	0,386	0,022	0,212	0,157
IAR		0,169	0,004	0,388	0,031	0,183	0,176
GAN	RRc	0,147	0,124	0,174	0,048	0,203	0,161
IAR		0,108	0,013	0,182	0,166	0,191	0,126
GAN	std	0,052	0,050				
IAR		0,344	0,091				
GAN	OSARG	0,170	0,079			0,217	0,079
IAR		0,223	0,079			0,098	0,081

	GAN		0,081	0,074			0,120	0,113
	IAR	SRV	0,051	0,024			0,059	0,054
	GAN		0,115	0,036			0,111	0,092
	IAR	MIRA	0,138	0,132			0,126	0,046
	GAN				0,258	0,194		
	IAR	MIRA_SR			0,250	0,206		
	GAN				0,111	0,066	0,241	0,018
	IAR	RRD			0,127	0,108	0,209	0,011
	GAN				0,203	0,158	0,178	0,025
	IAR	DSCT_SXPHE			0,260	0,079	0,152	0,043
	GAN				0,126	0,098	0,019	0,008
	IAR	T2CEP			0,135	0,130	0,048	0,005
	GAN						0,258	0,107
	IAR	NonVar					0,213	0,072
	GAN						0,144	0,136
	IAR	NC					0,139	0,126
	GAN						0,114	0,016
	IAR	ELL					0,114	0,030

Dado que los experimentos de *iAR* donde se forzó la generación de series no resultó tener mejor rendimiento que el *iAR* no forzado, no se le construyó una tabla comparativa de resumen homóloga a la tabla 17. Incluso, la comparación de los valores del modelo *GAN* entre las iteraciones del *iAR* forzado y no forzado, no tienen sentido ser comparadas, ya que, corresponden a las mismas instancias, por lo que presentan los mismos valores.

7. Análisis de datos

Por un lado, los modelos son entrenados desde los datos reales que existen, en este caso son 3 datasets. Dada la importancia de los datos, en muchas ocasiones, la limpieza o extracción de los datos no suele ser la mejor, lo que inevitablemente afecta los resultados obtenidos. Ahora, sumando estos errores o variaciones que se puedan producir hasta el momento, con todos los parámetros con los cuáles los modelos son configurados, puede resultar bien difícil encontrar la combinación que entregue mejores resultados y sean utilizables para su debido análisis y trabajos posteriores. Es por esto, que de los resultados, se pueden apreciar variados gráficos con variadas representaciones de una misma clase, como es el caso de la instancia **20-Mira** con un *accuracy* de **23,18 %** perteneciente al dataset de **WISE** ejecutado con un generador de una capa, a continuación.

A simple vista, se puede apreciar que el factor de elegir una muestra al azar de la clase no logró entregar un resultado representativo del foldeo, sin embargo, al momento de foldear una muestra de los datos generados, si se logra obtener una gráfica la cual nos permite conocer a simple vista el ciclo o periodo de la estrella. En contraste, podemos ver la figura número 14, donde podemos apreciar que el foldeo sobre la muestra tampoco logró ser demostrativa de la clase, pero, al momento de analizar el foldeo sobre los datos generados, se puede apreciar una similitud de la firma obtenida. Esto puede terminar en dos opciones, la primera, es que se podría inferir que la firma para el foldeo de la clase **MIRA** es de una forma cercana a la obtenida por este experimento. Sin embargo, la segunda alternativa podría quedar en que el modelo no tuvo datos lo suficientemente limpios para poder entrenar, dado que las muestras obtenidas de los datos reales no fueron representativas.

No solo se pueden analizar los resultados obtenidos del trabajo, sino que, también se deben observar los modelos utilizados. Por lo que, tomando como ejemplo el modelo de **LSTM** que entrega un *score* del *accuracy* sobre los datos reales versus los generados, se puede dar a cuentas que este modelo no nos entregaría un *score* representativo del caso cuando las series generadas sean iguales a las series reales. Esto se debe a cómo fue construido, ya que, en el fondo, este modelo está comparando una muestra de datos que sabemos que tiene un 50 % de datos reales y el otro 50 % de datos falsos, por ende, sabemos también, que con un correcto entrenamiento el mejor resultado de *accuracy* debiese ser 0,5, por ende, la métrica establecida debiese ser 0. Ahora, si suponemos que tenemos una serie de datos que fueron generados a la perfección y pueden catalogarse como reales (aunque en el fondo sabemos que no lo son), y se los pasamos al modelo **LSTM** junto con una serie de datos reales, el modelo no sería capaz de entregar un *score* de 0,5, debido a que los supuestos datos falsos serían categorizados como 100 % reales. Entonces, su *score* sería de 1,0. Este escenario demuestra que no necesariamente un valor mayor a 0,5 es siempre un mal resultado. Por esta razón, es que se definió la métrica definida por la ecuación 8, que permite conocer que tan alejado está el *accuracy* obtenido del *accuracy* esperado, tomando en cuenta que tanto las series reales podrían considerarse falsas, como las series falsas considerarse reales. Cabe destacar que esta métrica también presenta algunas zonas con errores, ya que, para los casos de series reales, si se le entrega una muestra completamente real al modelo, y esta es capaz de detectarla como real, su *accuracy* será 1 como se mencionó anteriormente, pero, su métrica sería entonces 0,5. La premisa de la métrica indica que entre más cercano al 0 es su valor, mayor similitud tienen sus datos. Para poder justificar estas zonas donde los resultados pueden

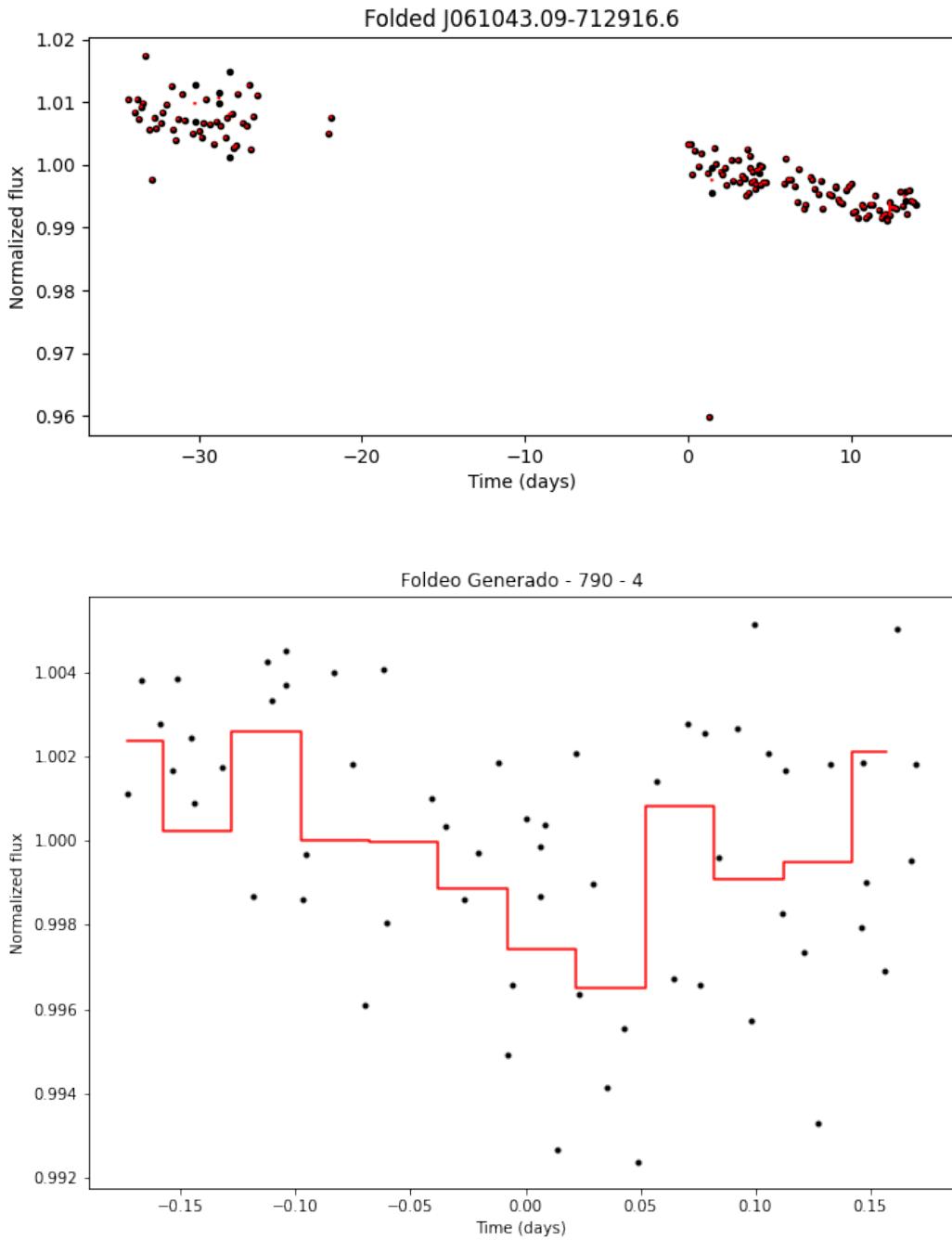


Figura 13: Ejemplo de una instancia foldeada donde no representa la clase original. La imagen superior debiese representar el foldeo original de la clase **MIRA**, mientras que la imagen inferior, muestra el foldeo obtenido de los datos generados de la clase.

ser malinterpretados, hay que destacar que el modelo necesariamente tiene que recibir de entrada una muestra equitativa de series reales y falsas, para así realizar un correcto análisis sobre los experimentos, justamente como se ha realizado en este trabajo.

Finalmente, los resultados que debiesen orientar a elegir un modelo u otro, están descritos por la tabla 17, que expresa que para los experimentos realizados en este trabajo,

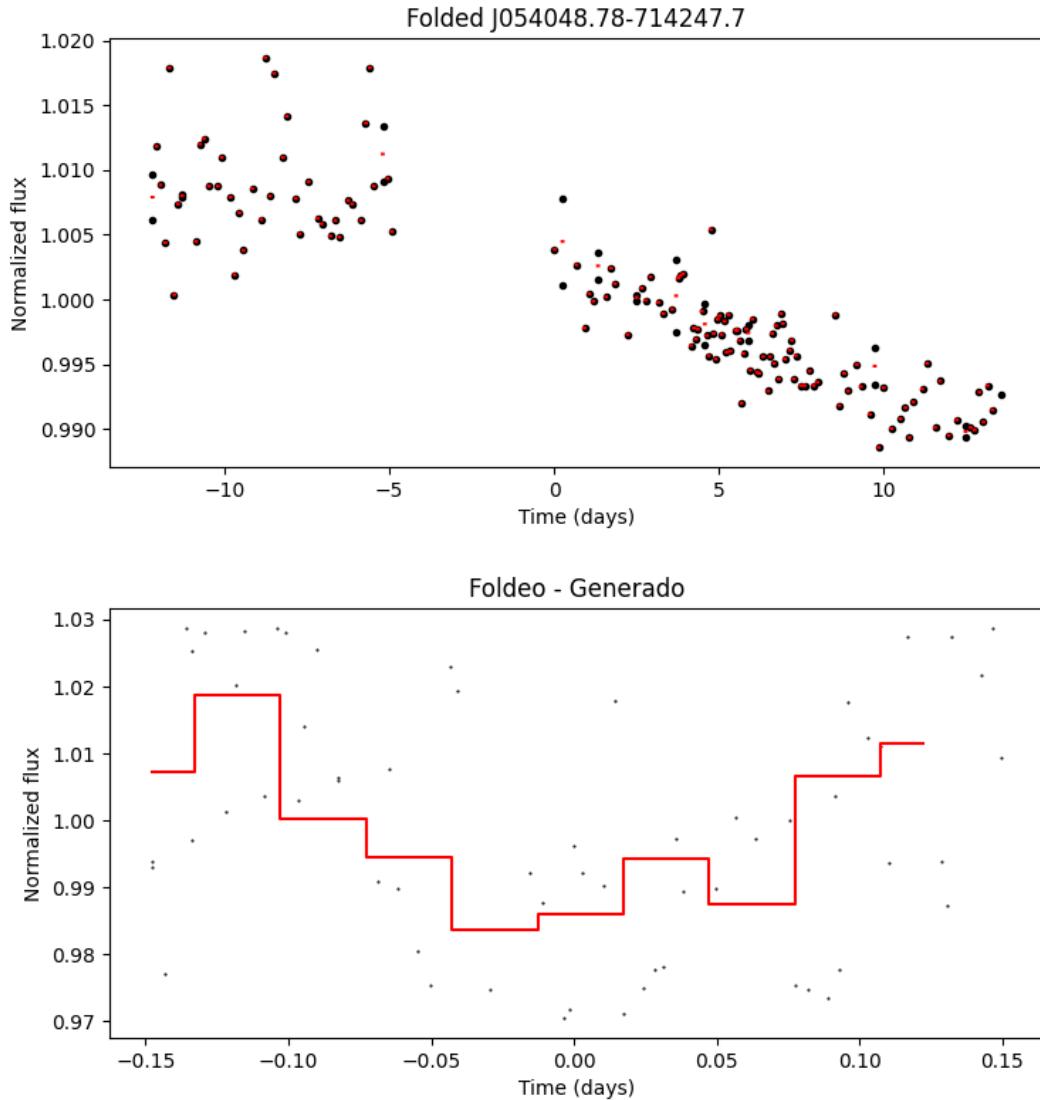


Figura 14: Ejemplo de una segunda instancia foldeada donde no representa la clase original. La imagen superior debiese representar el foldeo original de la clase MIRA, mientras que la imagen inferior muestra el foldeo obtenido de los datos generados de la clase.

el modelo ***GAN*** teniendo una capa generativa fue la que obtuvo el mejor rendimiento en general. Superando la librería de ***iAR*** con y sin forzado, e incluso, a su modelo homólogo de dos capas generativas. Ahora, si la selección del modelo se profundizara en función del dataset a utilizar, podemos ver que se tendría un mejor rendimiento, pasando desde **0,155** a **0,115** dada las métricas obtenidas. Las razones por la cual profundizar la selección mejora el resultado final, es debido a la cantidad de mediciones que traen cada dataset, y de la calidad de sus datos. Claramente, estos factores finalmente afectan a la mayoría de los modelos neuronales y no son un caso aislado que ocurre en este experimento, el origen de los datos, es sumamente importante cuando se desarrollan modelos complejos como los vistos en este trabajo.

8. Conclusión y trabajos futuros

De los resultados vistos en este trabajo, se puede apreciar que en cada dataset existen clases donde se ha podido generar correctamente series de datos que logran obtener un *accuracy* ideal junto a sus métricas que respaldan los foldeos generados frente a los reales. Los modelos descritos no son perfectos y se encuentra bien lejos de estarlo, pero, afinando los parámetros y trabajando con mayor profundidad sus arquitecturas y métodos para poder puntuar los resultados en base a una escala, se podrían esperar mejores resultados. El hecho de poder generar estas series de tiempo con el modelo de la *GAN* permite abrir las puertas para generar otras series de tiempo de otras clases y datasets, demostrando que es posible sólo si se afinan de mejor manera los modelos. Adicionalmente, el uso de estos datos generados no tiene por qué terminar aquí, ya que se podría extender el modelo de la *GAN* o condicionar con el mismo foldeo de las series generadas, para que de esta forma, en las siguientes iteraciones se generen las series de tiempo condicionadas a tener un foldeo más real. Esto permitirá acotar de mejor manera los resultados obtenidos y se esperaría que entregaran resultados con mejor rendimiento, tanto de las gráficas visuales como entregados por el modelo de la *LSTM de Score*. Existen distintas metodologías para realizar el foldeo de estrellas y calcular sus períodos. Por lo general, pertenecen a operaciones matemáticas más puras que suelen entregar resultados bien aproximados a los resultados reales. Por otro lado, estas maneras de desarrollar este tipo de experimentos, van acompañados de análisis profundos y muchas verificaciones, que suelen ser manuales y realizadas por Astrónomos o investigadores del área, dificultando un poco la labor. Son estas revisiones, análisis y comparaciones las que se busca atacar y solucionar en este trabajo, intentando acercarse lo más posible a los resultados que manualmente ya se obtienen, y que logran tener resultados suficientes para avanzar en las investigaciones. Esta problemática puede seguir desarrollándose si se analiza y se trabaja en afinar los parámetros de los modelos y el trabajo general realizado en este informe. Adicionalmente, existe la posibilidad de ser complementado por otros tipos de modelos, sea para generar datos o corroborar los datos obtenidos y clasificarlos, permitiendo así, clasificar estrellas las cuales presentan una cantidad muy baja de mediciones, dificultando sus estudios.

Referencias

- [1] Yang Wang. A mathematical introduction to generative adversarial nets (gan), 2020.
- [2] Sarro, L. M., Debosscher, J., López, M., and Aerts, C. Automated supervised classification of variable stars* - ii. application to the ogle database. *A&A*, 494(2):739–768, 2009.
- [3] Fotometría de la clase Classic Cepheid. <https://ogledb.astrouw.edu.pl/~ogle/OCVS/o.php?OGLE-LMC-CEP-0001>.
- [4] Fotometría de la clase Type II Cepheid. <https://ogledb.astrouw.edu.pl/~ogle/OCVS/o.php?OGLE-LMC-T2CEP-001>.
- [5] I.ński@, A. Udalski, M. K. Szymański, Ł. Wyrzykowski, K. Ulaczyk, R. Poleski, P. Pietrukowicz, S.łowski@, D. M. Skowron, J. Skowron, P.óz@, M. Pawlak, K. Rybicki, and A. Jacyszyn-Dobrzeniecka. The OGLE Collection of Variable Stars. Clas-

sical, Type II, and Anomalous Cepheids toward the Galactic Center. , 67(4):297–316, December 2017.

- [6] Felipe Elorrieta. iAR Package. https://github.com/felipeelorrieta/iAR/blob/master/examples/.ipynb_checkpoints/IAR_demo-checkpoint.ipynb, May 2021.
- [7] Ignacio Becker. Datasets. <https://drive.google.com/drive/folders/1Ywjz8RKq8fsqQrK3NBiFUVAs1P17y13I>, Feb 2020.
- [8] sklearn.model_selection.train_test_split. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, Oct 2021.
- [9] Geert Barentsen. A friendly package for kepler & tess time series analysis in python. <https://pypi.org/project/lightkurve/>, Sept 2021.
- [10] Lightkurve docs — lightkurve. <https://docs.lightkurve.org/#>.
- [11] Susana Eyheramendy, Felipe Elorrieta, and Wilfredo Palma. An irregular discrete time series model to identify residuals with autocorrelation in astronomical light curves. *Monthly Notices of the Royal Astronomical Society*, 481(4):4311–4322, Sep 2018.
- [12] Felipe Elorrieta, Susana Eyheramendy, and Wilfredo Palma. Discrete-time autoregressive model for unequally spaced time-series observations. *Astronomy Astrophysics*, 627:A120, Jul 2019.