FUNDAMENTOS DE DEEP LEARNING

PROYECTO

ENTREGA FINAL

Por

JUAN DIEGO ARROYAVE AGUIRRE

1036673963

CARLOS ANDRÉS OSUNA FLÓREZ

1214724934

Para:
RAUL RAMOS POLLAN

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA
2024

1. PREÁMBULO

Antes de empezar con el desarrollo de los ítems que debe de contener el trabajo, se abordar algunos aspectos importantes.

Primero, se presentan algunos gráficos de las series de tiempo para escenarios estables (etiquetados con el número binario “0”) y para escenarios inestables (etiquetados con el número binario “1”).

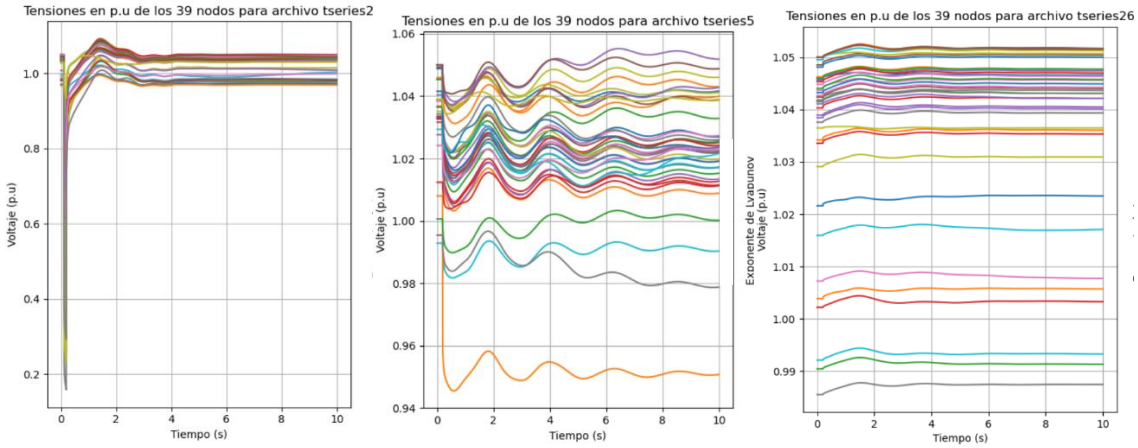


Figura 1. Ejemplo de algunas series de tiempo donde el comportamiento del sistema eléctrico es estable (etiqueta=0)

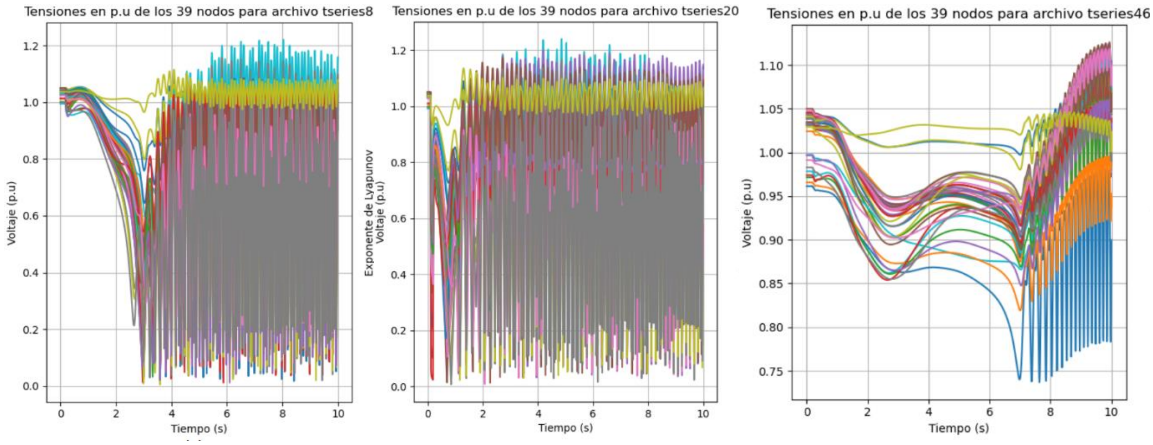


Figura 2. Ejemplo de algunas series de tiempo donde el comportamiento del sistema eléctrico es inestable (etiqueta=1)

Como se observa en los ejemplos de la Figura 1, un sistema eléctrico se etiqueta como estable cuando después de haber sido sometido a una perturbación debido a una condición operativa inicial determinada, es capaz de mantener las tensiones constantes en todas las barras del sistema [1]. Por otro lado, en la Figura 2 se presentan ejemplos donde el sistema eléctrico es inestable, es decir, luego de sometida la perturbación, el sistema no es capaz de mantener constante las tensiones en las barras, sino que, presentan una oscilación muy grande. Este tipo de análisis es muy importante monitorearlos y aún más importante, tener herramientas rápidas que puedan predecir con antelación el estatus del sistema y dejar un margen de tiempo de respuesta más amplio para adoptar medidas correctivas decisivas.

El segundo tema importante, es que, si bien se simularon 10.000 casos, algunas series de tiempo extrajeron resultados erróneos, por lo cual no se tendrán en cuenta en el análisis.

Además, para el análisis de este trabajo, se utilizaron las primeras 3.500 series de tiempo excluyendo a su vez las series de tiempo que tienen datos erróneos o defectuosos como se mencionó anteriormente. Se utilizan 3.200 para Train y Test (80% para Train y 20% para Test) y las 300 series de tiempo restante se utilizan como datos de validación. Los datos de las series de tiempo se encuentran publicadas en el siguiente link: https://github.com/Juandi11/Series_de_tiempo el cual se utiliza en el notebook de la solución.

Por último, se quiere explicar cuáles son las métricas de desempeño que se van a analizar en el trabajo. Para el problema de clasificación se han propuesto varias métricas de desempeño, una de las formas más utilizadas y apropiadas para valorar el desempeño de las herramientas de clasificación es la matriz de confusión. A partir de la información que entrega dicha matriz se plantean otras métricas de desempeño.

La matriz de confusión se define como una tabla de filas y columnas donde se representan las etiquetas reales (objetivos) y las etiquetas de predicción (salidas) entregadas por una herramienta de clasificación. Se utiliza la matriz de confusión para evaluar el desempeño de la herramienta de clasificación donde se reporta los conteos de las predicciones correctas e incorrectas, de la siguiente forma: Verdadero positivo (True Positive, TP) es predicción positiva y la etiqueta era positiva; falso positivo (False Positive, FP) es predicción positiva y la etiqueta era negativa; verdadero negativo (True Negative, TN) es predicción negativa y la etiqueta era negativa; falso negativo (False Negative, FN) es predicción negativa y la etiqueta era positiva [2]. En la Figura 3 se presenta un ejemplo de cómo se conforma una matriz de confusión para un problema de clasificación binaria.

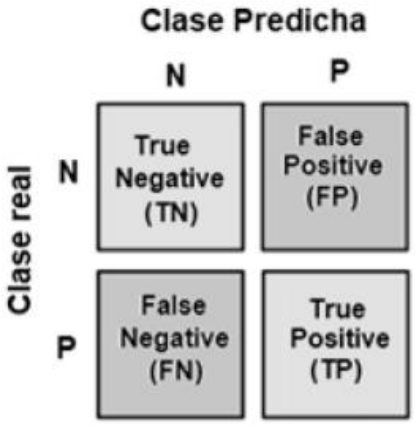


Figura 3. Matriz de confusión para un problema de clasificación binaria [2]

Para el problema de detección de la estabilidad que se trata en este trabajo (predicción del estatus estable o inestable del comportamiento del sistema) se tienen entonces que el negativo (0) es la etiqueta estable y el positivo (1) es la etiqueta inestable, por lo cual la matriz de confusión queda definida de la siguiente forma: Los TN son el número de casos que son verdaderos estables (N_{VE}); los FP son el número de casos falsos inestables (N_{FI}), es decir, casos estables clasificados como inestables; los FN son el número de casos que son falsos estables (N_{FE}), es decir, casos inestables clasificados como estables; el TP son el número de casos que son verdaderos inestables (N_{VI}), es decir, casos inestables

clasificados como inestables. En la Tabla 1 se presenta la matriz de confusión para el problema de clasificación o predicción del estatus de estabilidad de tensión.

Tabla 1. Matriz de confusión para el problema de clasificación del estatus de estabilidad de tensión

Matriz de confusión	**Estable (predicción)**	**Inestable (predicción)**
Estable (real)	N_{VE} (Verdadero estable)	N_{FI} (Falso inestable)
Inestable (real)	N_{FE} (Falso estable)	N_{VI} (Verdadero inestable)
Total	Casos estables (N_E)	Casos inestables (N_I)

A partir de la información entregada por la matriz de confusión se definen las siguientes métricas a tener en cuenta para la evaluación del desempeño de los modelos de clasificación usados en este trabajo [2]:

Exactitud: mide la proporción en la detección de forma correcta de los casos tanto estables como inestables:

$$Acc = \frac{N_{VE} + N_{VI}}{N_E + N_I} \quad (1)$$

Precisión: mide la exactitud en la detección de los casos que son inestables:

$$Precisión = \frac{N_{VI}}{N_{VI} + N_{FI}} \quad (2)$$

Recall: mide la proporción de los casos que son correctamente detectados como inestables por el clasificador, esta métrica también se conoce como la tasa de verdaderos positivos o sensibilidad:

$$Recall = \frac{N_{VI}}{N_{VI} + N_{FE}} \quad (3)$$

F1 score: se define como la media armónica entre la Precisión y el Recall, esta media armónica le da mucho más peso a los valores bajos, como resultado F1 score solo obtiene un valor alto si y solo si la precisión y el Recall son altos. Los valores del F1 score están entre cero a uno, donde cero es la peor calificación y uno la mejor calificación.

$$F1\ score = 2 * \frac{Precisión * Recall}{Precisión + Recall} \quad (4)$$

2. DESCRIPCIÓN DE LA ESTRUCTURA DE LOS NOTEBOOKS ENTREGADOS

a) Importación de librerías necesarias para la implementación

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
import tensorflow as tf
from tqdm import tqdm
import pandas as pd
import numpy as np
from tensorflow import keras
```

Figura 4. Sección con importación de librerías para ejecución de script

b) Lectura de datos de las series de tiempo y etiquetas de cada serie de tiempo

```
[3]  !git clone https://github.com/Juandi11/Series_de_tiempo.git #Se lee la URL donde se alojaron las series de tiempo
```

```
[71] malos = [156, 751, 947, 1314, 1586, 1739, 1742, 2075, 2286, 2528, 2747, 2939, 2961, 3030, 3226, 3245, 4276,
              4357, 4724, 4725, 4799, 4855, 5154, 5224, 5281, 5370, 5497, 5577, 5609, 5717, 5740, 5945, 6049,
              6366, 7607, 7754, 7909, 7950, 8099, 8307, 8331, 8556, 8826, 9038, 9276, 9510, 9984] #series de tiempo con datos erróneos

     cantidad=3200 #Cantidad de series de tiempo utilizadas para Train y Test

     Apredecir = [] # aqui van las series utilizadas para Train y Test, excluyendo las series de tiempo malas o con datos erróneos
     for i in range(1, cantidad+1):
       if (not i in malos): # se excluyen las que están malas
         Apredecir.append(i) #se agregan a la lista vacía

     s = [] # aqui se guardan las series ya procesadas

     tiempo_05= 62 # ventana de 0.5 segundos
     tiempo2 = 242 # ventana de 2 segundos
     tiempo4 = 482 # ventana de 4 segundos
     tiempo5 = 602 # ventana de 5 segundos
     tiempo6 = 722 # ventana de 6 segundos

     for i in tqdm(Apredecir):
       ser = pd.read_csv(f'/content/Series_de_tiempo/tseries_CSV/tseries{i}.csv', sep = ',') # se leen los archivos del github
       ser = ser.iloc[1:tiempo2,1:].apply(pd.to_numeric).to_numpy() # se toman solo los valores numericos indicados en la ventana de tiempo dada
       s.append(ser) # se añade a la lista de series
     ser = np.array(s) # se pasa a un array el total de series
     del(s) #Se elimina la lista s

     # se lee el archivo de etiquetas de ceros (0s) y unos (1s) de las series a usar en Train y Test
     y = pd.read_excel('/content/Series_de_tiempo/tseries_CSV/Lyapunov_analysis_10000_cases_mios_clasif.xlsx')['Lyapunov'][Apredecir].to_numpy(float)
```

Figura 5. Sección con lectura de datos para Train, Test y etiquetas

c) Partición de los datos para el conjunto de entrenamiento (train), prueba (test) y balanceo de datos

```
[72] X_train, X_test, y_train, y_test = train_test_split(ser, y, test_size=0.2, random_state=42) #Se parte el conjunto de datos en Train y Test
```

```
[73] #Sección de balanceo de datos
     s1,s2,s3 = X_train.shape[0],X_train.shape[1],X_train.shape[2]
     print(s1,s2,s3)
     X_train = np.reshape(X_train,(s1, s2*s3)) #Se hace una redimensión de los datos para entrarlos al proceso de balanceo
     print(X_train.shape)

     #Importo nueva librería
     from imblearn.combine import SMOTEENN

     smenn=SMOTEENN()

     # Balanceo de datos
     X_resampled, y_train = smenn.fit_resample(X_train, y_train)

     #Se vuelve a las dimensiones originales de los datos de Train
     X_resampled = np.reshape(X_resampled, (X_resampled.shape[0], s2, s3))
```

Figura 6. Sección de partición de datos para Train, Test y balanceo de datos

Debido a que se cuenta con pocas etiquetas de 1 (sistemas inestables) en comparación con las etiquetas de 0 (sistemas estables) utilizadas para el sistema de entrenamiento, es

necesario balancear los datos para garantizar que en la predicción se tenga un porcentaje alto en la detección de casos inestables. Para esto se utiliza el método de balanceo SMOTEENN la cual es una técnica de sobremuestreo de minorías sintéticas. Es una técnica utilizada en el aprendizaje automático para abordar conjuntos de datos desequilibrados [3].

d) Creación del modelo a usar

```
[77] #Se crea el modelo
     model = Sequential([
         LSTM(35, input_shape=(ser.shape[1], ser.shape[2]), return_sequences=True),
         LSTM(35),
         Dense(1, activation='sigmoid')
     ])
     #Métricas a utilizar
     Metrics = [
           keras.metrics.BinaryAccuracy(name='Exactitud'),
           keras.metrics.Precision(name='Precisión'),
           keras.metrics.Recall(name='Recall'),
           ]

     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=Metrics)
```

Figura 7. Modelo a usar en el trabajo.

e) Entrenamiento del modelo

```
[79] # Entrenamiento del modelo
     model.fit(X_resampled, y_train, epochs=20)
```

Figura 8. Entrenamiento del modelo.

f) Etapa de prueba (test) del modelo

```
[80] # Prueba de Test
     predictions = model.predict(X_test)
     p = predictions > .5
     r = y_test.reshape(y_test.shape[0], 1)
     print(100*np.mean(p == r)) # promedio de porcentaje de predicciones correctas
```

```
[ ] #Resultados de Test
    print(confusion_matrix(y_test, p.astype(int)))
    report = classification_report(y_test, p.astype(int))
    print("\nInforme de clasificación:")
    print(report)
```

Figura 9. Etapa de test y resultados con matriz de confusión

g) Lectura de datos para validación

```
[82] #Lectura de datos para validación
     cantidad_val1=cantidad+1
     cantidad_val2=cantidad+300 #Se utilizan otras 300 series de tiempo más para la validación

     Avalidacion = [] # aqui van las series que se van a utilizar para validación
     for i in range(cantidad_val1, cantidad_val2+1):
       if (not i in malos): # se excluyen las que están malas
         Avalidacion.append(i) #Se agregan a la lista vacía

     s2 = [] # aqui se guardan las series ya procesadas

     for i in tqdm(Avalidacion):
       ser2 = pd.read_csv(f'/content/Series_de_tiempo/tseries_CSV/tseries{i}.csv', sep = ',') # se leen los archivos del github
       ser2 = ser2.iloc[1:tiempo2,1:].apply(pd.to_numeric).to_numpy() # se toman solo los valores numericos indicados en la ventan de tiempo dada
       s2.append(ser2) # se añade a la lista de series
     ser2 = np.array(s2) # se pasa a un array el total de series
     del(s2) #se elimina la lista s2

     # se lee el archivo de etiquetas de ceros (0s) y unos (1s) de las series a usar para Validación
     y2 = pd.read_excel('/content/Series_de_tiempo/tseries_CSV/Lyapunov_analysis_10000_cases_mios_clasif.xlsx')['Lyapunov'][Avalidacion].to_numpy(float)
```

Figura 10. Sección con lectura de datos para validación

h) Etapa de validación del modelo

```
# Prueba de Validación
predictions_valid = model.predict(ser2)
p2 = predictions_valid > .5 # los que tienen probabilidad mayor a 50% de ser 1 se ponen como 1, el resto es 0
r2 = y2.reshape(y2.shape[0], 1)
print(100*np.mean(p2 == r2)) # promedio de porcentaje de predicciones correctas
```

```
[86] #Resultados de Validación
     print(confusion_matrix(y2, p2.astype(int)))
     report2 = classification_report(y2, p2.astype(int))
     print("\nInforme de clasificación:")
     print(report2)
```

Figura 11. Etapa de validación y resultados con matriz de confusión

3. DESCRIPCIÓN DE TU SOLUCIÓN (ARQUITECTURA, PREPROCESADO, ETC)

Se entiende por arquitectura a la forma como se define el ingreso y salida de la información a la red. La propuesta de arquitectura para la Red Neuronal Recurrente (RNN) del tipo LSTM usada en este trabajo se conoce como arquitectura “many to one”, en donde las mediciones previas de la serie de tiempo son usadas para la clasificación de la condición final de la secuencia.

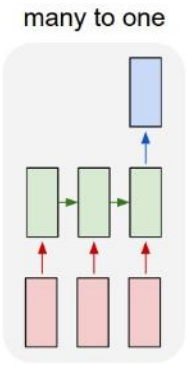


Figura 12. Arquitectura “many to one” o muchos a uno.

Esta arquitectura consiste en que las capas de redes están apiladas, y a su vez, estas son usadas para capturar las diferentes características en varios niveles de capas desde la entrada hasta la salida de la red. Por ejemplo, la red RNN-LSTM utilizada en el trabajo es el que se muestra en la Figura 13, donde se tiene una primer capa de entrada LSTM con 35 neuronas y con la cantidad de muestras de la ventana escogida (para el caso de la

Figura 13 es de 241 muestras que corresponde a una ventana de 2 segundos), luego se tiene una segunda capa LSTM también con 35 neuronas y por último se tiene la capa de salida “Dense” con la función de activación tipo “sigmoid”.

```
[77] #Se crea el modelo
     model = Sequential([
         LSTM(35, input_shape=(ser.shape[1], ser.shape[2]), return_sequences=True),
         LSTM(35),
         Dense(1, activation='sigmoid')
     ])
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_8 (LSTM)               (None, 241, 35)           10500

 lstm_9 (LSTM)               (None, 35)                9940

 dense_4 (Dense)             (None, 1)                 36

=================================================================
Total params: 20476 (79.98 KB)
Trainable params: 20476 (79.98 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figura 13. Arquitectura del modelo usado con una RNN-LSTM.

Las redes RNN-LSTM son diseñadas para tomar secuencias de n pasos de tiempo de las mediciones como entradas, la arquitectura interna de cada celda LSTM y la funcionalidad de cada compuerta no lineal, le permite a la red LSTM utilizar y pasar la información desde el principio hasta el final de la secuencia. La capa final de bloques LSTM solo pasa la salida hacia adelante a lo largo de la secuencia de tiempo, y la salida en el tiempo t es una capa de red completamente conectada (FC) con una función de activación Sigmoide para la clasificación del tipo binaria.

4. DESCRIPCIÓN DE LAS ITERACIONES QUE HICISTE

Las iteraciones y resultados que se muestran a continuación se realizan con el conjunto de datos de Test. Las iteraciones que se realizaron fueron básicamente:

- Sensibilidad del número de neuronas de las capas del modelo utilizado

Resultados de matriz de confusión con 35 neuronas en cada capa y ventana completa de 10 segundos

```
[[510  12]
 [ 33  83]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.94      0.98      0.96       522
         1.0       0.87      0.72      0.79       116

    accuracy                           0.93       638
   macro avg       0.91      0.85      0.87       638
weighted avg       0.93      0.93      0.93       638
```

Figura 14. Métricas de desempeño con matriz de confusión para modelo con 35 neuronas

Resultados de matriz de confusión con 39 neuronas en cada capa y ventana completa de 10 segundos

```
[[512  10]
 [ 40  76]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.93      0.98      0.95       522
         1.0       0.88      0.66      0.75       116

    accuracy                           0.92       638
   macro avg       0.91      0.82      0.85       638
weighted avg       0.92      0.92      0.92       638
```

Figura 15. Métricas de desempeño con matriz de confusión para modelo con 39 neuronas

Resultados de matriz de confusión con 110 neuronas en cada capa y ventana completa de 10 segundos

```
[[495  27]
 [ 30  86]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.94      0.95      0.95       522
         1.0       0.76      0.74      0.75       116

    accuracy                           0.91       638
   macro avg       0.85      0.84      0.85       638
weighted avg       0.91      0.91      0.91       638
```

Figura 16. Métricas de desempeño con matriz de confusión para modelo con 110 neuronas

De las métricas evaluadas en la matriz confusión, se puede observar que da mejores resultados con 35 neuronas en cada capa. Por lo cual se escoge esta cantidad de neuronas para el trabajo.

- Luego se realiza una sensibilidad del tamaño de ventana de tiempo o número de muestras, es decir, las series de tiempo tienen una ventana total de 10 segundos, entonces lo que se hizo fue disminuir la ventaja de tiempo y ver el comportamiento de las predicciones.

Resultados de matriz de confusión con ventana de tiempo de 6 segundos:

```
[[511  11]
 [ 32  84]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.94      0.98      0.96       522
         1.0       0.88      0.72      0.80       116

    accuracy                           0.93       638
   macro avg       0.91      0.85      0.88       638
weighted avg       0.93      0.93      0.93       638
```

Figura 17. Métricas de desempeño con matriz de confusión para ventana de 6 segundos

Resultados de matriz de confusión con ventana de tiempo de 4 segundos:

```
[[509  13]
 [ 30  86]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.94      0.98      0.96       522
         1.0       0.87      0.74      0.80       116

    accuracy                           0.93       638
   macro avg       0.91      0.86      0.88       638
weighted avg       0.93      0.93      0.93       638
```

Figura 18. Métricas de desempeño con matriz de confusión para ventana de 4 segundos

Resultados de matriz de confusión con ventana de tiempo de 2 segundos:

```
[[515   7]
 [ 37  79]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96       522
         1.0       0.92      0.68      0.78       116

    accuracy                           0.93       638
   macro avg       0.93      0.83      0.87       638
weighted avg       0.93      0.93      0.93       638
```

Figura 19. Métricas de desempeño con matriz de confusión para ventana de 2 segundos

Resultados de matriz de confusión con ventana de tiempo de 0.5 segundos:

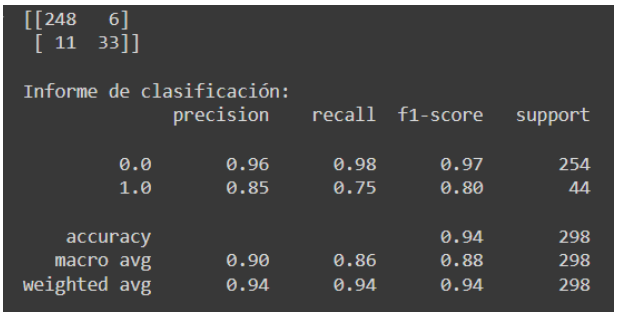
```
[[512  10]
 [ 31  85]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.94      0.98      0.96       522
         1.0       0.89      0.73      0.81       116

    accuracy                           0.94       638
   macro avg       0.92      0.86      0.88       638
weighted avg       0.93      0.94      0.93       638
```

Figura 20. Métricas de desempeño con matriz de confusión para ventana de 0.5 segundos

De las iteraciones realizadas en la variación de la ventana, se puede concluir que todas presentan un adecuado desempeño en las métricas, sin embargo, se decide utilizar una ventana de 2 segundos, ya que, es la que mayor precisión tiene al momento de identificar casos inestables (etiqueta=1).

5. DESCRIPCIÓN DE LOS RESULTADOS

El modelo finalmente utilizado es un red RNN-LSTM, como se mostró en la Figura 13, la cual contiene 2 capas LSTM de 35 neuronas cada una y una capa de salida “Dense” con función de activación “sigmoid”. Con esta red utilizada, para el conjunto de datos de Test, se obtuvieron los resultados mostrados en la sección anterior en la Figura 19. Además, también se hizo la evaluación para un conjunto de datos escogido para validación. Los resultados de la matriz de confusión se muestran a continuación:



```
[[248   6]
 [ 11  33]]

Informe de clasificación:
              precision    recall  f1-score   support

         0.0       0.96      0.98      0.97       254
         1.0       0.85      0.75      0.80        44

    accuracy                           0.94       298
   macro avg       0.90      0.86      0.88       298
weighted avg       0.94      0.94      0.94       298
```

Figura 21. Métricas de desempeño con matriz de confusión para conjunto de validación y modelo final

De la figura anterior se puede concluir lo siguiente:

- El modelo propuesto tiene una precisión del 96% en predecir bien los sistemas eléctricos estables (etiqueta=0) y una precisión del 85% en predecir correctamente los sistemas inestables (etiqueta=1).
- La exactitud del modelo es del 94% y se puede calcular con la ecuación (1) presentada en este documento en la sección 1.

$$Acc = \frac{N_{VE} + N_{VI}}{N_E + N_I} \quad (5)$$

$$Acc = \frac{248 + 33}{254 + 44} \quad (6)$$

$$Acc = \frac{281}{298} = 0.94 = 94\% \quad (7)$$

- En [4] donde se realiza un trabajo similar de clasificación de estabilidad de tensión, se obtuvo un porcentaje de exactitud por encima de 90% y de precisión por encima del 70%, por lo cual, se puede concluir de forma general que el modelo desarrollado en este trabajo cuenta con métricas bastantes altas y que son aceptables.

6. REFERENCIAS

[1] P. Kundur, «Definition and classification of power system stability,» *IEEE Transactions on Power System,* pp. 1387-1401, 2004.

[2] J. Patterson y A. Gibson, Deep Learning: A Practitioner's Approach. First Edition, 2017.

[3] «SMOTE for Imbalanced Classification with Python,» [En línea]. Available: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[4] W. M. V. Acevedo, «EVALUACIÓN DE LA ESTABILIDAD DE TENSIÓN DE LARGO Y CORTO ALCANCE USANDO HERRAMIENTAS DE INTELIGENCIA ARTIFICIAL,» San Juan, 2023.