Modelos de predicción para detectar pacientes con enfermedades cardiacas empleando redes neuronales y las bibliotecas *Pytorch* y *TensorFlow*

Prediction models to detect patients with heart diseases using neural networks and the *Pytorch* and *TensorFlow* libraries

Nelson del Castillo Collazo
IIMAS, UNAM, Ciudad de México, México
nelson.delcastillo@iimas.unam.mx
ORCID: 0000-0002-4187-5511

Adalberto Joel Durán Ortega
IIMAS, UNAM, Ciudad de México, México
joel.duran@iimas.unam.mx
ORCID: 0000-0003-4272-7294

D. Fabián García Nocetti
IIMAS, UNAM, Ciudad de México, México
fabian.garcia@iimas.unam.mx
ORCID: 0000-0001-9861-610X

doi: https://doi.org/10.36825/RITI.12.26.010

Recibido: Septiembre 19, 2024
Aceptado: Noviembre 11, 2024

Resumen: Se emplean las Redes Neuronales para generar modelos de predicción con el fin de determinar si un individuo tiene padecimientos cardiacos, se utilizan las bibliotecas *Pytorch* y *TensorFlow* aplicados a un conjunto de datos de pacientes relacionados con enfermedades del corazón, los cuales cuentan con 17 variables predictoras. El propósito de este trabajo es comparar los resultados que se obtienen empleando las dos bibliotecas antes mencionadas usando Redes Neuronales, analizando el comportamiento de las funciones de pérdida y las medidas resultantes de la matriz de confusión al crear el modelo de predicción. Se emplean las técnicas *DownSampling* y *UpSampling* para tratar el desbalance en el conjunto de datos, el cual está formado por 319,795 pacientes en total y de ellos solo 27,373 presentan enfermedades de corazón. Se encontró que para este conjunto de datos los mejores resultados con *Pytorch* se obtienen para los modelos de 100 épocas en adelante y con un tiempo de ejecución de pocos segundos, mientras que con *TensorFlow* se obtienen buenos resultados a partir de los modelos de 10 épocas, pero su tiempo de ejecución es considerablemente mayor. Se hace un análisis sobre la diferencia en tiempo de cálculo entre *Pytorch* y *TensorFlow*.

Palabras clave: *Enfermedades Cardiovasculares, Redes Neuronales, Pytorch, TensorFlow, Modelos de Predicción.*

Abstract: Neural Networks are used to generate prediction models aimed at determining whether an individual has heart conditions. The PyTorch and TensorFlow libraries are applied to a dataset of patients related to heart diseases, containing 17 predictor variables. The purpose of this work is to compare the results obtained using the aforementioned libraries with Neural Networks, analyzing the behavior of loss functions and the outcomes from the confusion matrix when creating the prediction model. DownSampling and UpSampling techniques are employed to address the imbalance in the dataset, which consists of a total of 319,795 patients, of whom only 27,373 have heart disease. It was found that for this dataset, the best results with PyTorch are achieved in models of 100 epochs and above, with execution times of only a few seconds, while TensorFlow shows good results starting from models with 10 epochs, though its execution time is considerably longer. An analysis is conducted on the difference in computation time between PyTorch and TensorFlow.

Keywords: *Heart Disease, Neural Networks, Pytorch, TensorFlow, Prediction Models.*

1. Introducción

Una de las aplicaciones de la Inteligencia Artificial más recurrente en la actualidad buscada por los seres humanos, es la de construir modelos de predicción de enfermedades a través de los datos recabados por los diferentes dispositivos presentes en el mercado, como son *Smart watches*, bases de datos y otros dispositivos *IOT* que permitan recolectar datos relevantes de los pacientes, ya sea dentro del hospital o su conexión a la *web.* El objetivo principal con la Inteligencia Artificial es, que se busca prolongar la calidad de vida, usando los datos obtenidos para evaluar su correlación con los factores de riesgo.

El propósito de este trabajo es que, a partir de un conjunto de datos de pacientes relacionados con enfermedades cardiovasculares, se comparen los resultados que se obtienen cuando se emplean las bibliotecas *Pytorch* y *TensorFlow* usando Redes Neuronales, analizando el comportamiento de las funciones de pérdida y las medidas resultantes de la matriz de confusión al crear el modelo de predicción.

Se sabe que en la era actual las bases de datos electrónicas de salud y las técnicas avanzadas de minería de datos han mejorado la capacidad para analizar patrones complejos entre signos cardíacos y enfermedades relacionadas, permitiendo diagnósticos más precisos y pronósticos más confiables [1], [2] y [3]. Sin embargo, en ocasiones los grandes volúmenes de datos vienen junto con un gran desbalance de estos, en cuyo caso se hace presente la necesidad de emplear las técnicas de Balanceo de datos. Las técnicas más documentadas hasta el momento son: *UpSampling* y *DownSampling*, las cuales permiten lidiar con problemas de desequilibrio en los conjuntos de datos utilizados para modelos predictivos [4]; en el caso de la de técnicas de balanceo tipo *UpSampling* se emplea un aumento de instancias de la clase minoritaria y para la técnica de *DownSampling* se realiza una reducción de instancias de la clase mayoritaria. Cabe resaltar que ambas técnicas han ido evolucionando y perfeccionando su desempeño junto con las herramientas y bibliotecas avanzadas de aprendizaje automático.

El creciente desarrollo de las redes neuronales artificiales no es nuevo, éste comenzó entre 1950 y 1980, con conceptos como el perceptrón y las redes multicapa, y aunque en sus inicios enfrentaron ciertas limitaciones. En los años 1990 y 2000 las técnicas como la retropropagación de errores impulsaron su resurgimiento, y a partir de las bibliotecas *TensorFlow* y *PyTorch* han facilitado su uso y acceso a herramientas para el aprendizaje profundo. Sin embargo, la ejecución de las herramientas junto con los datos se veía afectada debido a que el entrenamiento de redes se realizaba en CPUs, proceso que limitaba en gran medida su velocidad. Con la llegada de las GPU y las TPUs a partir de 2010 se ha transformado este proceso, permitiendo cálculos paralelos y reduciendo drásticamente los tiempos de entrenamiento para los modelos complejos [5]. Dichos modelos predictivos basados en redes neuronales tienen limitaciones en la gestión de incertidumbres, pero desde 2010, se han desarrollado técnicas avanzadas, como las redes neuronales bayesianas y los métodos de Monte Carlo, que ayudan a mejoran la estimación de incertidumbres y la fiabilidad de los pronósticos.

2. Materiales y métodos

Los datos fueron tomados de la página de *Kaggle* [6] los cuales están relacionados con indicadores de enfermedades cardiacas, estos corresponden a dos categorías para predecir si un individuo padece una enfermedad cardiaca o no. La cantidad de datos con las que se cuenta son 319,795 registros en total, de ellos 292,422 pertenecen a la *clase No* (individuos que no presentan enfermedades cardiacas) y 27,373 a la *clase Si* (individuos que si padecen del corazón), en la Figura 1 se puede apreciar que existe un marcado desbalance de datos.

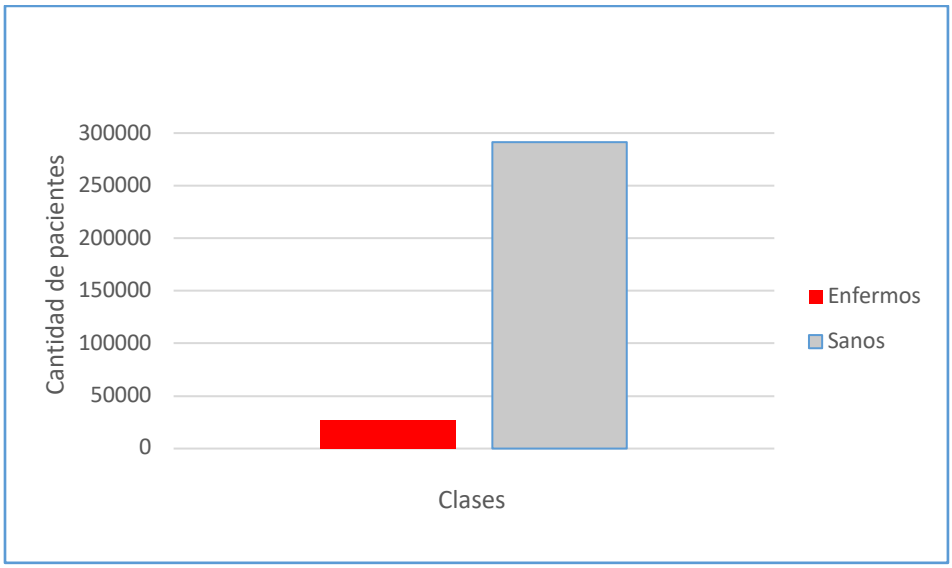


Figura 1. Cantidad de pacientes por *clase*.
Fuente: Elaborada por el autor.

El conjunto de datos está formado por 18 variables, 17 de ellas independientes (variables predictoras) y la décimo octava es la que se busca pronosticar, es decir, la variable dependiente donde se identifica si un individuo sufre o no de enfermedades del corazón.

En la Tabla 1 se puede apreciar las variables que integran este conjunto de datos, incluida la variable *HeartDisease* que es la que se desea predecir.

Tabla 1. Nombres de las variables que intervienen en el estudio.

No.	Nombre de la variable	Descripción
1	*BMI:*	Índice de masa corporal
2	*Smoking:*	Fumador
3	*AlcoholDrinking:*	Bebe alcohol
4	*Stroke:*	Accidente cerebrovascular
5	*PhysicalHealth:*	Días al mes con malestar en su salud física
6	*MentalHealth:*	Días al mes con malestar en su salud mental
7	*DiffWalking:*	Dificultad para subir escaleras
8	*Sex:*	Genero
9	*AgeCategory:*	Categoría de la edad
10	*Race:*	Color de piel
11	*Diabetic:*	Padece diabetes
12	*PhysicalActivity:*	Actividad física
13	*GenHealth:*	Bienestar

14	*SleepTime:*	Número de horas de sueño
15	*Asthma:*	Padece asma
16	*KidneyDisease*	Afección renal
17	*SkinCancer:*	Cáncer de piel
18	*HeartDisease:*	Enfermedad del corazón

Fuente: Elaborada por los autores.

Para el análisis de los datos y el desarrollo del modelo de clasificación se empleó el lenguaje de programación Python versión 3.8.19, utilizando las bibliotecas *Pytorch* y *TensorFlow* en las versiones 1.10+c113 y 2.10.0 respectivamente, se empleó la interfaz de desarrollo Visual Studio Code versión 1.92.2 y la tarjeta NVIDIA GeForce RTX 4050 para el uso del GPU.

2.1 Método de clasificación y desbalance de datos

Se emplearon Redes Neuronales para generar los modelos de pronóstico, los cuales están basados en las redes neuronales biológicas. Estas están formadas por un conjunto de capas, las cuales se dividen en capas de entrada, capas ocultas y capas de salida, todas ellas formadas por nodos o neuronas a través de las cuales se realizan una serie de cálculos que conducen a la generación del modelo de pronóstico que tienen una gran cantidad de aplicaciones, como pueden ser: clasificación, regresión, reconocimientos de imágenes, procesamiento del lenguaje natural, etc. Para profundizar en este tema se puede consultar [7] y [8].

Pytorch es una biblioteca o *framework* del lenguaje de programación Python, la cual proporciona una serie de herramientas que permiten la creación y entrenamiento de redes neuronales. Una de sus características principales es el uso de tensores que son estructuras de datos parecidas a los arreglos (unidimensionales, bidimensionales, etc.) los cuales se emplean para realizar cálculos matriciales y operaciones numéricas empleando tanto GPU como CPU [9], en el caso de la biblioteca de *TensorFlow,* esta tiene características similares a *Pytorch,* pero es su lugar, es una biblioteca de aprendizaje automático que soporta múltiples lenguajes de programación, su implementación principal y más utilizada es en Python [10].

Se emplearon las técnicas *DownSamplig* y *UpSamplig* [4] para tratar el desbalance de datos; para la primera se seleccionaron dos muestras aleatorias con una cantidad igual de individuos a la *clase Si,* la cual se corresponde con los pacientes enfermos, mientras que para el segunda se tomaron dos muestras aleatorias de un tamaño tres veces la cantidad de pacientes enfermos.

Se corrieron todas las muestras aplicando el algoritmo de Redes Neuronales con las bibliotecas *Pytorch* y *TensorFlow* por separado y empleado en cada una de ellas la estructura siguiente: la capa de entrada compuesta por 17 nodos que se corresponden con las variables predictoras, luego dos capas ocultas, la primera de ellas con 21 nodos y la segunda con 7, finalmente la capa de salida con dos nodos. Para todas las muestras se generaron los modelos con 10, 50, 100, 150 y 200 épocas. Se empleó la función de activación *ReLu* (*Rectified Linear Unit*) y el método de optimización de *Adam* (*Adaptive Moment Estimation*).

Los conjuntos de datos de entrenamiento estuvieron compuestos en todos los casos por el 80% y el 20% para el conjunto de validación. Se hizo uso de la GPU con soporte para CUDA la cual puede realizar cálculos paralelos masivos, lo que las hace útiles para aplicaciones de aprendizaje automático.

3. Resultados y discusión

Se generó un modelo de pronóstico empleando todo el conjunto de datos sin aplicar ninguna técnica que permitieran balancearlos, para tomar como referencia su comportamiento. En la Tabla 2 se muestra la matriz de confusión óptima, que es la que el modelo debería pronosticar. Como se puede observar los datos de validación del modelo están conformados por 58,574 pacientes que no presentan enfermedad del corazón (*clase No*) y por 5,385 pacientes que si presentan padecimiento del corazón (*clase Si*).

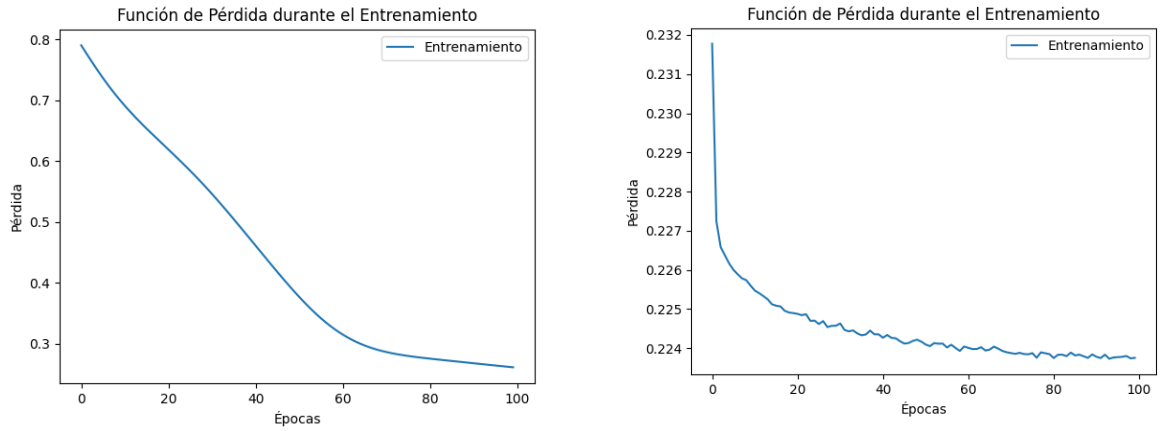
Tabla 2. Matriz de confusión óptima.

	No	**Si**
No	58574	0
Si	0	5385

Fuente: Elaborada por los autores.

En la Figura 2 se muestran las funciones de pérdida de las dos bibliotecas empleadas, los cuales indican la diferencia entre las predicciones del modelo y los valores reales [11] y [12] empleando 100 épocas en cada una de ellas.

En el caso de la biblioteca *Pytorch* comienza desde valores cercanos a 0.8 y disminuyen hasta llegar por debajo de 0.3 lo que indica que el modelo puede estar aprendiendo bien, ya que reduce la pérdida de manera significativa durante el entrenamiento; para la biblioteca de *TensorFlow* comienza cercano a 0.232 y disminuye hasta 0.224, esto nos indica que el modelo se encuentra en una zona de estabilidad, lo que podría hacer pensar que ha encontrado un mínimo local. Las pequeñas fluctuaciones que presenta pueden deberse a que el modelo no está aprendiendo de manera significativa para la cantidad de épocas empleadas.



(a) (b)

Figura 2. Comportamiento de la función de pérdida para las bibliotecas (a) *Pytorch* y (b) *TensorFlow*, con 100 épocas en cada caso.

Fuente: Elaborada por los autores.

Se generaron dos modelos de pronóstico empleando las bibliotecas, *Pytorch* y *TensorFlow,* y se obtuvieron las dos matrices de confusión del conjunto de datos de validación. Estos resultados se pueden apreciar en la Tabla 3.

Tabla 3. Matriz de confusión resultante al aplicar las dos bibliotecas al conjunto de datos de validación.

	Pytorch		*TensorFlow*	
Épocas	**100**		**100**	
	No	**Si**	**No**	**Si**
No	58574	0	58131	443
Si	5385	0	4861	524

Fuente: Elaborada por los autores.

En el caso de *Pytorch* pronostica de manera excelente los valores para los pacientes que no están enfermos (*clase No*) mientras que para los pacientes con padecimiento cardiaco es nulo el pronóstico (*clase Si*), es decir, todos los pacientes fueron pronosticados incorrectamente. Para el caso de *TensorFlow* mejoró el pronóstico de los pacientes enfermos, pero de todos modos se quedan lejos del resultado al que debería llegar, disminuye el pronóstico para los pacientes sanos en un 0.76% y aumenta el pronóstico para los pacientes enfermos en un 9.73%.

En la Tabla 4 se muestran las medidas resultantes de la matriz de confusión al validar los modelos, se toma la c*lase Si* como clase positiva. Se puede apreciar que los valores de Exactitud de los modelos son altos, en ambos casos con poco más de 91%.

En el caso de *Pytorch*, la sensibilidad [13] y [14], que es el porciento de valores positivos que son clasificados como positivos y los valores de predicción positivos (VP+) que indican la probabilidad de que un valor sea positivo si resultó positivo en la predicción son cero, es decir, se pronostican erróneamente. Los valores de especificidad, que es el porciento de negativos que son clasificados como negativos y los valores de predicción negativos (VP-) que indican la probabilidad de que un valor sea negativo si resultó negativo en la predicción son altos, confirmando los resultados de la matriz de confusión resultante.

En el caso de *TensorFlow*, el comportamiento de los resultados de la matriz de confusión es muy parecida a la obtenida con *Pytorch,* aunque se puede apreciar que mejora un poco en los resultados del pronóstico resultante [15] y [16].

Es importante mencionar que existió una diferencia marcada en el tiempo de cálculo entre ambas bibliotecas para la misma cantidad de épocas, siendo *TensorFlow* la que consumió mayor tiempo.

Tabla 4. Medidas de la Matriz de Confusión de ambas técnicas.

	Pytorch	*TensorFlow*
Épocas:	**100**	**100**
Exactitud (%):	91.58	91.71
Sensibilidad (%):	0.00	9.73
Especificidad (%):	100.00	99.24
VP+ (%):	0.00	54.19
VP- (%):	91.58	92.28
Tiempo (seg.):	4.12	1919.80

Fuente: Elaborada por los autores.

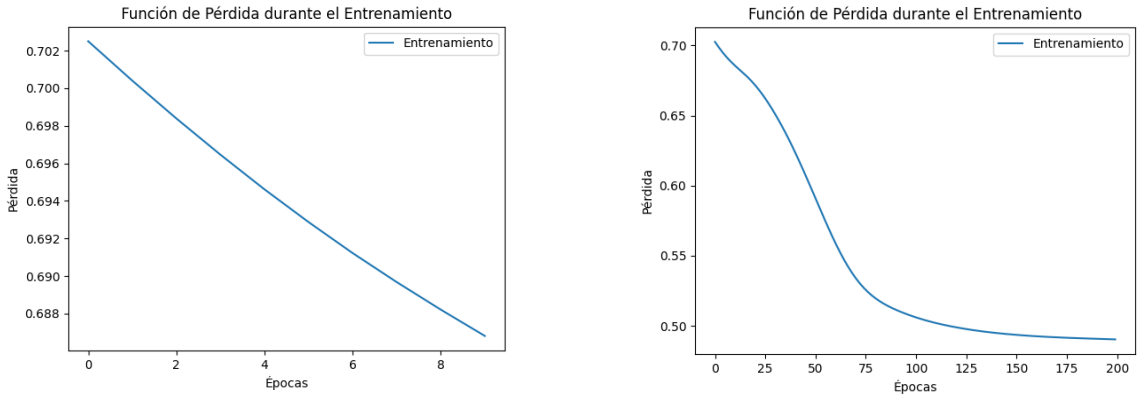
Estos resultados indican que se deben aplicar técnicas para mitigar el desbalance de datos, es este caso se emplearon *DonwSampling* y *UpSampling*.

3.1. Redes Neuronales empleando la biblioteca Pytorch

3.1.1 DownSampling

Una vez seleccionado de forma aleatoria las dos muestras de los pacientes sanos (*clase No)*, cada una de ellas se combinó con los datos de los pacientes enfermos (*clase Si)* de menara independiente.

A continuación, se muestran los resultados obtenidos en la muestra uno. En la Figura 3 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas. Se aprecia una tendencia decreciente de la función de pérdida de *Pytorch* la cual varía de 0.702 hasta 0.688 en 10 épocas (a), indicando que le modelo está mejorando durante el proceso de entrenamiento, pero aún puede mejorar más. En el caso de la función de pérdida de 200 épocas (b) la pérdida disminuye significativamente entre 0.7 y 0.5 indicando que en la fase inicial el modelo está aprendiendo muy rápido; a partir de la época 50 la curva comienza a aplanarse disminuyendo su ritmo de aprendizaje lo que indica que el modelo ha capturado la mayor parte de los patrones en los datos; después de 100 épocas sigue disminuyendo pero de manera más lenta hasta llegar a la época 200 donde alcanza un valor en la función de pérdida cercano a 0.48.



(a) (b)

Figura 3. Comportamiento de la función de pérdida usando la biblioteca *Pytorch* en la muestra uno. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

En la Tabla 5 se presenta la matriz de confusión óptima a replicar para esta primera muestra, mientras en la Tabla 6 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 7 las medidas resultantes de estas matrices de confusión. Se aprecia que el valor de exactitud en el modelo para 10 épocas es bajo al igual que la especificidad mientras la sensibilidad es mucho más alta, aquí a pesar de que la exactitud del modelo no es buena la identificación de enfermos es bastante alta (98.28%), no así en el caso de los pacientes sanos (8%). Para el resto de las épocas la identificación de los pacientes, tanto enfermos como sanos, mejoran significativamente al igual que los valores de exactitud, sensibilidad y especificidad del modelo, alcanzando un valor de exactitud de 76.16% para 200 épocas. Los tiempos de cálculo varían muy poco, estando entre 2.57 y 3.24 segundos.

Tabla 5. Matriz de confusión óptima para la muestra uno.

	No	**Si**
No	5435	0
Si	0	5515

Fuente: Elaborada por los autores.

Tabla 6. Matrices de confusión de la muestra uno para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	435	5000	4164	1271	4123	1312	4056	1379	4028	1407
Si	95	5420	1570	3845	1370	4145	1248	4267	1204	4311

Fuente: Elaborada por los autores.

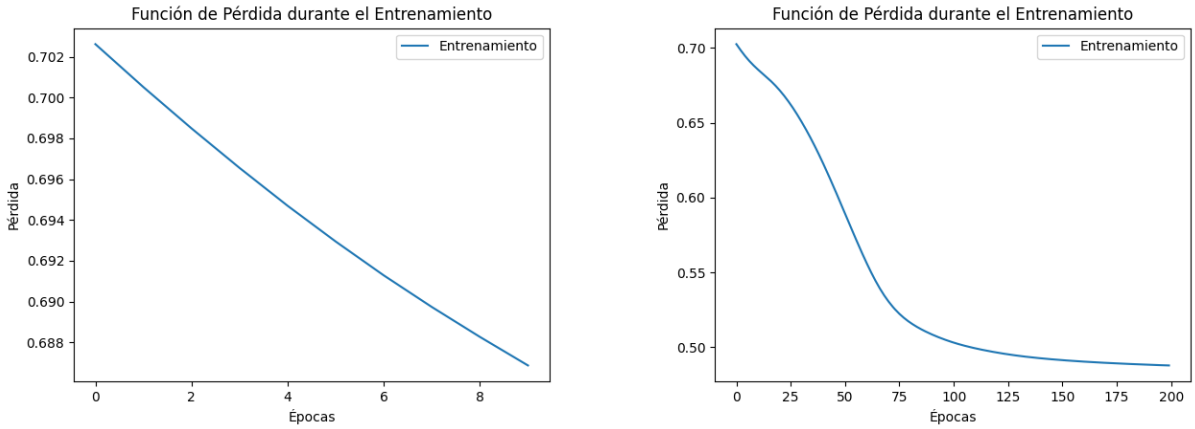
Tabla 7. Medidas de las Matrices de Confusión de la muestra uno para las diferentes épocas.

Épocas:	**10**	**50**	**100**	**150**	**200**
Exactitud (%):	53.47	74.05	75.51	76.01	76.16
Sensibilidad (%):	98.28	71.53	75.16	77.37	78.17
Especificidad (%):	8.00	76.61	75.86	74.63	74.11
VP+ (%):	52.02	75.63	75.96	75.58	75.39
VP- (%):	82.08	72.62	75.06	76.47	76.99
Tiempo (seg.):	2.57	2.63	2.90	2.94	3.24

Fuente: Elaborada por los autores.

En la Figura 4 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas de la segunda muestra, en la Tabla 8 se presenta la matriz de confusión óptima a replicar para esta muestra, mientras en la Tabla 9 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 10 las medidas resultantes de estas matrices de confusión.

Se puede apreciar que los resultados obtenidos en esta segunda muestra coinciden con los obtenidos en la primera; en este caso la mejor exactitud se alcanza en el modelo de 150 épocas con un valor de 75.84%, ligeramente menor al que se obtuvo en la muestra uno.



(a) (b)

Figura 4. Comportamiento de la función de pérdida usando la biblioteca *Pytorch* en la muestra dos. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

Tabla 8. Matriz de confusión óptima para la muestra dos.

	No	**Si**
No	5438	0
Si	0	5512

Fuente: Elaborada por los autores.

Tabla 9. Matrices de confusión de la muestra dos para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	417	5021	4130	1308	4108	1330	4046	1392	3959	1479
Si	98	5414	1517	3995	1373	4139	1254	4258	1179	4333

Fuente: Elaborada por los autores.

Tabla 10. Medidas de las Matrices de Confusión de la muestra dos para las diferentes épocas.

Épocas:	**10**	**50**	**100**	**150**	**200**
Exactitud (%):	53.25	74.2	75.32	75.84	75.73
Sensibilidad (%):	98.22	72.48	75.09	77.25	78.61
Especificidad (%):	7.67	75.95	75.54	74.4	72.8
VP+ (%):	51.88	75.33	75.68	75.36	74.55
VP- (%):	80.97	73.14	74.95	76.34	77.05
Tiempo (seg.):	2.41	2.59	2.62	3.04	3.29

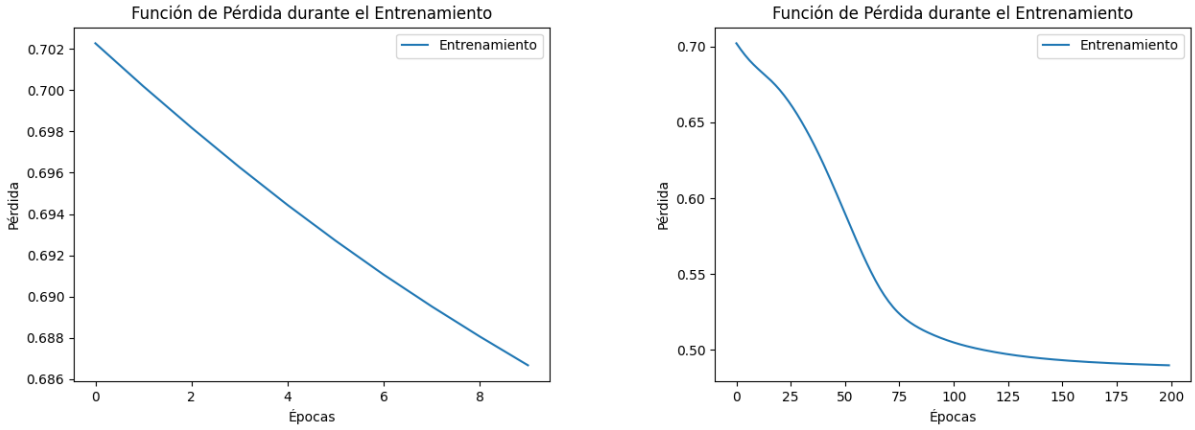
Fuente: Elaborada por los autores.

3.1.2 UpSampling

Para el caso de esta técnica se aumentó la *clase Si* tres veces su tamaño inicial. Se tomaron dos muestras aleatorias del conjunto de datos de la *clase No,* las cuales contienen también tres veces la cantidad de valores de la *clase Si* original, cada una de ellas se combinó con los datos de la *clase Si* de menara independiente.

En la Figura 5 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas, en la Tabla 11 se presenta la matriz de confusión óptima de esta primera muestra, mientras en la Tabla 12 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 13 las medidas resultantes de estas matrices de confusión.

Los resultados obtenidos con esta técnica de desbalance de datos empleando *Pytorch* no se diferencian prácticamente de los obtenidos cuando se aplicó la técnica de *DownSampling*. Las curvas de función de pérdida se comportan de manera similar y la exactitud de modelo llega a 76.38% en el modelo de 200 épocas. Para el caso del modelo de 10 épocas también presenta dificultad para predecir los pacientes sanos alcanzando solo el 6.70% de ellos. El resto de las medidas de las matrices de confusión obtenidas se comportan de manera similar en todos los casos.



(a) (b)

Figura 5. Comportamiento de la función de pérdida usando la biblioteca *Pytorch* en la muestra uno. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

Tabla 11. Matriz de confusión óptima para la muestra uno.

	No	**Si**
No	16452	0
Si	0	16396

Fuente: Elaborada por los autores.

Tabla 12. Matrices de confusión de la muestra uno para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	1103	15349	12526	3923	12511	3841	12343	4109	12157	4295
Si	263	16133	4593	11803	4044	12352	3662	12734	3465	12931

Fuente: Elaborada por los autores.

Tabla 13. Medidas de las Matrices de Confusión de la muestra uno para las diferentes épocas.

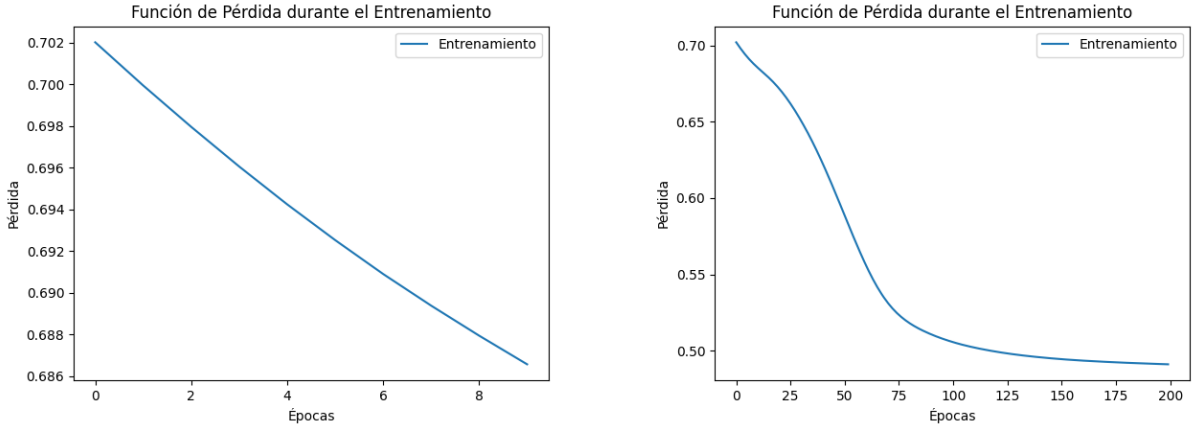
Épocas:	**10**	**50**	**100**	**150**	**200**
Exactitud (%):	52.47	74.07	75.69	76.34	76.38
Sensibilidad (%):	98.40	71.99	75.34	77.67	78.87

Especificidad (%):	6.70	76.15	76.05	75.02	73.89
VP+ (%):	51.25	75.05	75.81	75.6	7.075
VP- (%):	80.75	73.17	75.57	77.12	77.82
Tiempo (seg.):	2.61	2.87	3.24	3.70	3.96

Fuente: Elaborada por los autores.

Se muestran los resultados obtenidos en la segunda muestra. En la Figura 6 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas, en la Tabla 14 se presenta la matriz de confusión óptima a replicar para esta segunda muestra, mientras en la Tabla 15 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 16 las medidas resultantes de estas matrices de confusión.

Los resultados obtenidos en esta segunda muestra coinciden con los obtenidos en la primera; la mejor exactitud se alcanza en el modelo de 200 épocas con un valor de 76.67%, ligeramente mayor al que se obtuvo en la muestra uno. Los resultados en el modelo de 10 épocas siguen estando muy por debajo de los obtenidos en los modelos de 50 a 200 épocas.



(a) (b)

Figura 6. Comportamiento de la función de pérdida usando la biblioteca *Pytorch* en la muestra dos. (a) 10 épocas y (b) 200 épocas.
Fuente: Elaborada por los autores.

Tabla 14. Matriz de confusión óptima para la muestra dos.

	No	**Si**
No	16535	0
Si	0	16313

Fuente: Elaborada por los autores.

Tabla 15. Matrices de confusión de la muestra dos para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	1053	15482	15271	3964	12614	3921	12417	4118	12262	4273
Si	246	16067	4357	11956	3954	12359	3558	12755	3391	12922

Fuente: Elaborada por los autores.

Tabla 16. Medidas de las Matrices de Confusión de la muestra dos para las diferentes épocas.

Épocas:	**10**	**50**	**100**	**150**	**200**

Exactitud (%):	52.12	74.67	76.03	76.63	76.67
Sensibilidad (%):	98.49	73.29	75.76	78.19	79.21
Especificidad (%):	6.37	76.03	76.29	75.1	74.16
VP+ (%):	50.93	75.1	75.92	75.59	75.15
VP- (%):	81.06	74.26	76.13	77.73	78.34
Tiempo (seg.):	2.62	2.91	3.12	3.57	3.92

Fuente: Elaborada por los autores.

Es importante mencionar que en todos los resultados con la *biblioteca* de *Pytorch* los tiempos de cálculo fueron menores a 4 segundos.

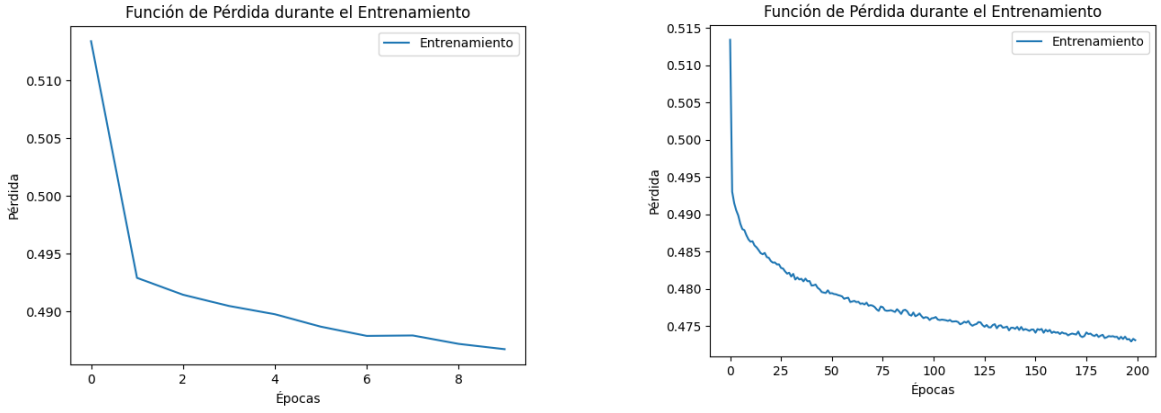
3.2. Redes Neuronales empleando la biblioteca TensorFlow

3.2.1 DownSampling

Una vez seleccionado de forma aleatoria el conjunto de datos de la *clase No* para las dos muestras, cada una de ellas se combinó con los datos de la *clase Si* de menara independiente. Cada una de las muestras tiene la misma cantidad de elementos que la *clase Si*.

En la Figura 7 se presentan los gráficos de las funciones de pérdida para los modelos de 10 y 200 épocas empleando la biblioteca *TensorFlow*. En el caso del primero, hay una disminución fuerte para la primera época, variando de poco más de 0.51 hasta 0.493 aproximadamente, indicando que el modelo está aprendiendo muy rápido cuando está iniciando el entrenamiento, situación que no sucede en ninguno de los casos obtenidos con la biblioteca *Pytorch*, tanto en *DownSampling* como *UpSampling*; después de la primera época continúa disminuyendo pero ya más lento y en la época 6 comienza a estabilizarse hasta alcanzar aproximadamente el valor de 0.486, este valor nunca fue alcanzado con *Pytorch* que llegó solo a 0.687.

En el caso del gráfico de la función de pérdida para modelo de 200 épocas, tiene una caída pronunciada hasta la época 75 y a partir de ahí comienza a descender de manera más suave hasta legar al valor de 0.473 aproximadamente en la época 200; presenta pequeñas fluctuaciones las cuales pueden deberse a que el modelo no está aprendiendo de manera significativa para esa cantidad de épocas, para esto se generó un modelo con 5000 épocas, el resultado se puede ver la Figura 8, donde la función de pérdida no disminuye de manera significativa, por lo que solo con generar el modelo con 200 épocas sería suficiente.



(a) (b)

Figura 7. Comportamiento de la función de pérdida usando la biblioteca *TensorFlow* en la muestra uno. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

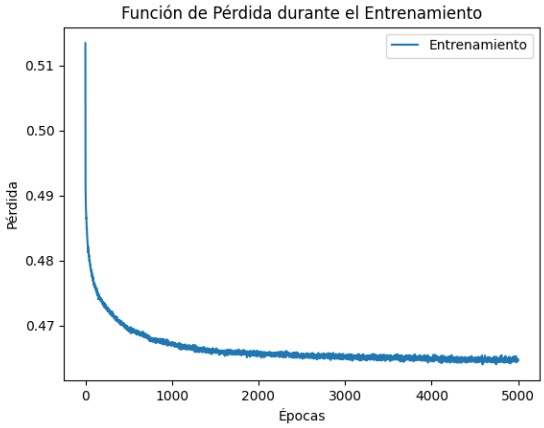


Figura 8. Comportamiento de la función de pérdida usando la biblioteca *TensorFlow* para 5000 épocas.
Fuente: Elaborada por los autores.

En la Tabla 17 se presenta la matriz de confusión óptima a replicar para esta primera muestra, mientras en la Tabla 18 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 19 las medidas resultantes de estas matrices de confusión.

Los resultados que se obtienen en las medidas de las matrices de confusión son buenos desde la época 10, situación que no sucede cuando se emplea *Pytorch*, los valores de exactitud de todos los modelos están alrededor del 76%, siendo el mayor en el caso del modelo de 10 épocas con 76.32%. El comportamiento de las matrices de confusión se corresponde con los resultados de las medidas obtenidas en cada una de ellas, siendo muy parecidos los valores de sensibilidad y especificidad lo que nos indica que en todos los modelos la predicción de los pacientes sanos y enfermos fue buena.

Tabla 17. Matriz de confusión óptima para la muestra uno.

	No	**Si**
No	5435	0
Si	0	5515

Fuente: Elaborada por los autores.

Tabla 18. Matrices de confusión de la muestra uno para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	4018	1417	3950	1485	4023	1412	3883	1552	3826	1609
Si	1176	4339	1127	4388	1217	4298	1061	4454	1031	4484

Fuente: Elaborada por los autores.

Tabla 19. Medidas de las Matrices de Confusión de la muestra uno para las diferentes épocas.

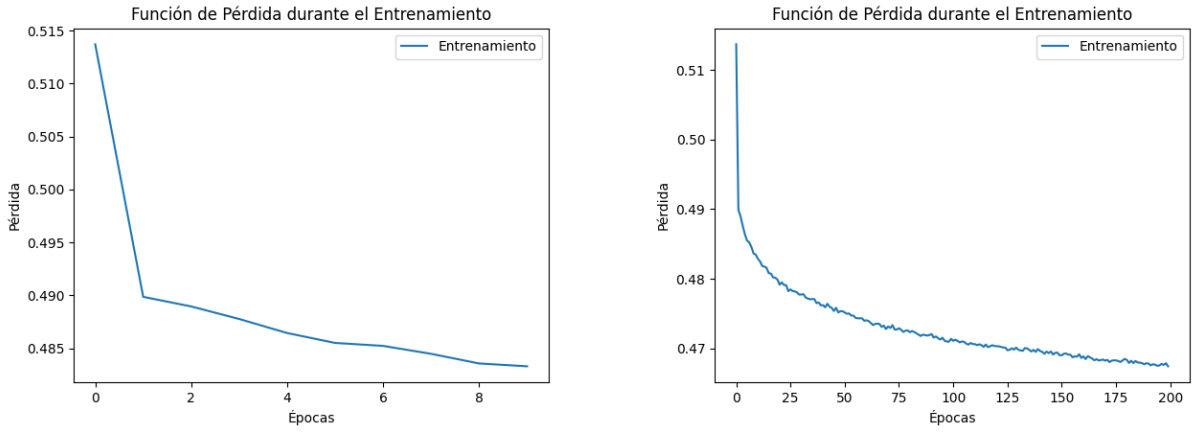
Épocas:	**10**	**50**	**100**	**150**	**200**
Exactitud (%):	76.32	76.15	75.99	76.14	75.89
Sensibilidad (%):	78.68	79.56	77.93	80.76	81.31
Especificidad (%):	73.93	72.68	74.02	71.44	70.4
VP+ (%):	75.38	74.71	75.27	74.16	73.59
VP- (%):	77.36	77.80	76.77	78-54	78.77
Tiempo (seg.):	35.17	172.28	385.29	533.37	691.79

Fuente: Elaborada por los autores.

En cuanto a los tiempos de cálculo obtenidos son significativamente mayores a los obtenidos cuando se emplea *Pytorch*, en ese caso no pasaban de 4 segundos mientras que con *TensorFlow* el modelo de menos épocas se demora poco más de 35 segundos, llegando a 691 en el modelo de 200 épocas.

A continuación, se presentan los resultados obtenidos en la muestra dos empleando *TensorFlow* con la técnica *DownSamplig*. En la Figura 9 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas, en la Tabla 20 se presenta la matriz de confusión óptima a replicar para esta segunda muestra, mientras en la Tabla 21 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 22 las medidas resultantes de estas matrices de confusión.

Los resultados obtenidos en esta segunda muestra coinciden con los obtenidos en la primera; en este caso la mejor exactitud se alcanza en el modelo de 50 épocas con un valor de 75.55%, quedando ligeramente por debajo al que se obtenido hasta ahora.



(a) (b)

Figura 9. Comportamiento de la función de pérdida usando la biblioteca *TensorFlow* en la muestra dos. (a) 10 épocas y (b) 200 épocas.
Fuente: Elaborada por los autores.

Tabla 20. Matriz de confusión óptima para la muestra dos.

	No	**Si**
No	5438	0
Si	0	5512

Fuente: Elaborada por los autores.

Tabla 21. Matrices de confusión de la muestra dos para las diferentes épocas.

Épocas	**10**		**50**		**100**		**150**		**200**	
	No	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**	**No**	**Si**
No	3929	1509	3909	1529	4030	1408	4032	1406	3840	1598
Si	1176	4336	1148	4364	1277	4235	1312	4200	1125	4387

Fuente: Elaborada por los autores.

Tabla 22. Medidas de las Matrices de Confusión de la muestra dos para las diferentes épocas.

Épocas:	**10**	**50**	**100**	**150**	**200**
Exactitud (%):	75.48	75.55	75.48	75.18	75.13
Sensibilidad (%):	78.66	79.17	7683	76.2	79.59
Especificidad (%):	72.25	71.88	74.11	74.14	70.61

VP+ (%):	74.18	74.05	75.05	74.92	73.3
VP- (%):	76.96	77.30	75.94	75.45	77.34
Tiempo (seg.):	34.84	172.28	329.95	513.39	674.60

Fuente: Elaborada por los autores.

3.2.2 UpSampling

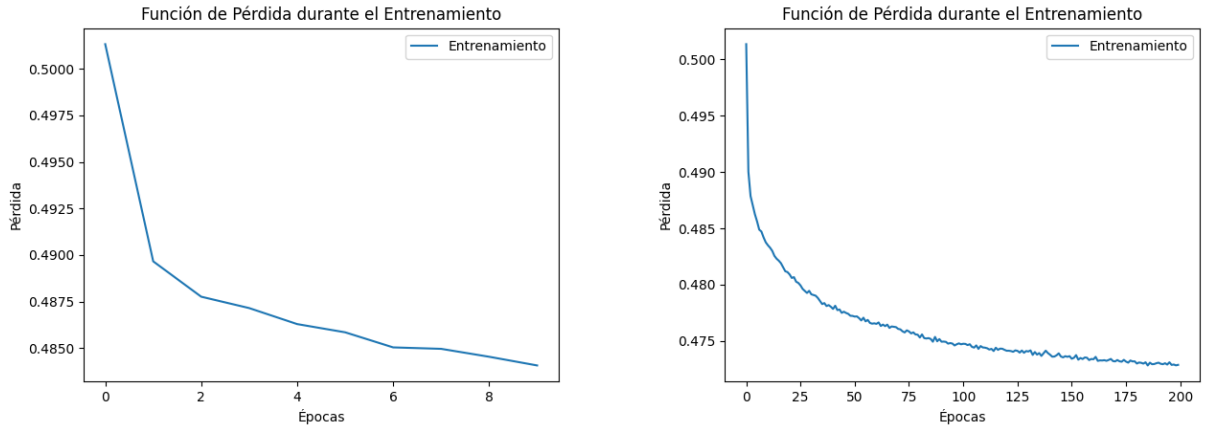
Aquí se aumentó la *clase Si* tres veces su tamaño inicial. Se tomaron dos muestras aleatorias del conjunto de datos de la *clase No,* las cuales contienen tres veces la cantidad de valores de la *clase Si* original, cada una de ellas se combinó con los datos de la *clase Si* de menara independiente.

En la Figura 10 se presentan los gráficos de la función de pérdida, para 10 y 200 épocas de la muestra uno, en la Tabla 23 se presenta la matriz de confusión óptima de esta primera muestra, mientras en la Tabla 24 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 25 las medidas resultantes de estas matrices de confusión.

En este caso la función de pérdida para 10 épocas bajó hasta poco menos de 0.4850, valor un poco menor al alcanzado con la técnica de *DownSamplig*, la cual solo llegó a 0.487, el resto de la curva se comporta parecida a las obtenidas en las muestras anteriores de *TensorFlow*. En el caso de la función de pérdida para el modelo de 200 épocas se comporta parecido a los obtenidos con anterioridad.

Los valores de exactitud para los modelos de 150 y 200 épocas superan los valores de 77% los cuales son los mejores obtenidos hasta el momento, tanto con *Pytorch* como con *TensorFlow*. Los valores de sensibilidad quedaron por encima de los valores de especificidad entre 7% y 9% en casi todos los casos, indicando que se hizo una mejor predicción para los pacientes enfermos.

Los tiempos de cálculo aumentaron con respecto a los modelos obtenidos con *DownSampling* pues el tamaño de las muestras fue tres veces mayor, variando de 94 segundos en el caso de 10 épocas hasta 1902 segundos para 200 épocas.



(a) (b)

Figura 10. Comportamiento de la función de pérdida usando la biblioteca *TensorFlow* en la muestra uno. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

Tabla 23. Matriz de confusión óptima para la muestra uno.

	No	**Si**
No	16452	0
Si	0	16396

Fuente: Elaborada por los autores.

Tabla 24. Matrices de confusión de la muestra uno para las diferentes épocas.

Épocas	10		50		100		150		200	
	No	Si	No	Si	No	Si	No	Si	No	Si
No	1284	4368	11743	4709	11915	4537	12204	4248	12043	4409
Si	3215	13181	2905	13491	3026	13370	3302	13094	3108	13288

Fuente: Elaborada por los autores.

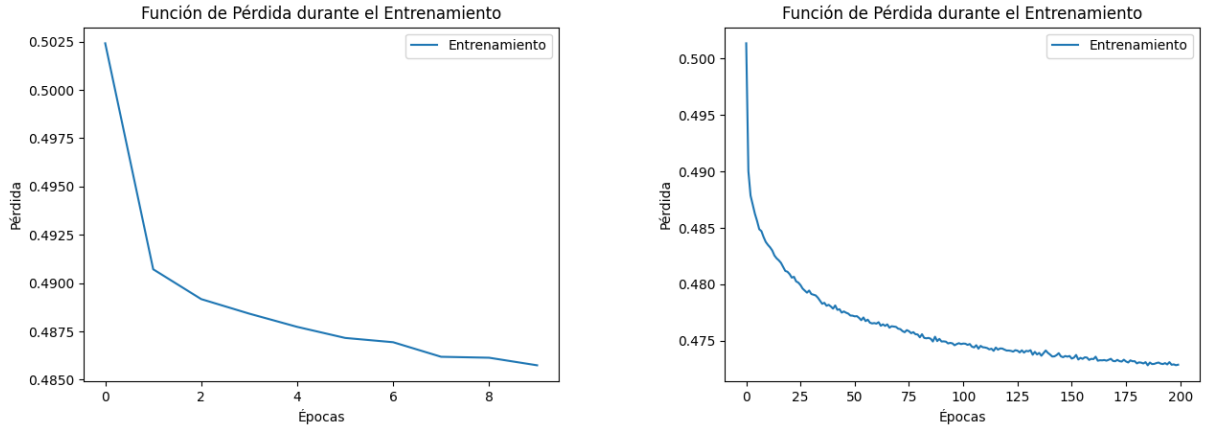
Tabla 25. Medidas de las Matrices de Confusión de la muestra uno para las diferentes épocas.

Épocas:	10	50	100	150	200
Exactitud (%):	75.91	76.82	76.98	77.02	77.12
Sensibilidad (%):	80.39	82.28	81.54	79.86	81.04
Especificidad (%):	73.45	71.38	72.42	74.18	73.2
VP+ (%):	75.11	74.13	74.66	75.5	75.09
VP- (%):	78.99	80.17	79.75	78.71	79.49
Tiempo (seg.):	94.71	483.03	999.11	1446.30	1902.25

Fuente: Elaborada por los autores.

En la Figura 11 se presentan los gráficos de la función de pérdida en los modelos de 10 y 200 épocas para la segunda muestra, en la Tabla 26 se presenta la matriz de confusión óptima, mientras en la Tabla 27 se muestran las matrices de confusión obtenidas para cada una de las épocas y en la Tabla 28 las medidas resultantes de estas matrices de confusión.

Los resultados obtenidos en esta segunda muestra coinciden con los obtenidos en la primera; los valores de exactitud de los modelos 150 y 200 también estuvieron por arriba del 77%.



(a) (b)

Figura 11. Comportamiento de la función de pérdida usando la biblioteca *TensorFlow* en la muestra dos. (a) 10 épocas y (b) 200 épocas.

Fuente: Elaborada por los autores.

Tabla 26. Matriz de confusión óptima para la muestra dos.

	No	Si
No	16535	0
Si	0	16313

Fuente: Elaborada por los autores.

Tabla 27. Matrices de confusión de la muestra dos para las diferentes épocas.

Épocas	10		50		100		150		200	
	No	Si	No	Si	No	Si	No	Si	No	Si
No	12544	3991	11935	4600	12230	4305	12272	4263	11821	4714
Si	3640	12673	3030	13283	3277	13036	3245	13018	2839	13474

Fuente: Elaborada por los autores.

Tabla 28. Medidas de las Matrices de Confusión de la muestra dos para las diferentes épocas.

Épocas:	10	50	100	150	200
Exactitud (%):	76.77	76.77	76.92	77.14	77.01
Sensibilidad (%):	77.69	81.43	79.91	80.11	82.6
Especificidad (%):	75.86	72.18	73.96	74.22	71.49
VP+ (%):	76.05	74.28	75.17	75.4	74.08
VP- (%):	77.51	79.75	78.87	79.09	80.63
Tiempo (seg.):	95.97	488.01	960.60	1422.00	1916.30

Fuente: Elaborada por los autores.

3.3. Análisis de las diferencias en los tiempos de ejecución de los modelos de Pytorch y TensorFlow

En todos los resultados obtenidos, existen marcadas diferencias en los tiempos de ejecución entre las bibliotecas de *Pytorch* y *TensorFlow,* como se había mencionado anteriormente, ambas bibliotecas permiten la construcción y entrenamiento de redes neuronales, pero tienen algunas diferencias que influyen en el tiempo de ejecución de los modelos. *Pytorch,* es más liviano en modelos con operaciones básicas por su interfaz de bajo nivel, permitiendo reducir la sobrecarga de las operaciones en el entrenamiento del modelo, sobre todo en aquellos que tienen una estructura más simple como la que presentan los modelos analizados, mientras que *TensorFlow* tiene un diseño con una mayor cantidad de funcionalidades que pueden aumentar su complejidad y afectar los tiempos de ejecución.

Pytorch hace un manejo más simple y eficiente en el uso de las GPU, la cual permite una transferencia de los datos entre la CPU y la GPU mucho más fluida, permitiéndole a esta biblioteca gestionar de manera más dinámica la memoria del GPU facilitando una disminución de la ejecución del modelo; en el caso de *TensorFlow* tiene más dificultades para optimizar la memoria del GPU, trayendo como consecuencia que se pueda ralentizar el entrenamiento del modelo, sobre todo en la transferencia de datos al GPU [6], [7] y [17].

4. Conclusiones

Se muestran las funciones de pérdida, estas son fundamentales en el entrenamiento de modelos de aprendizaje automático, sobre todo, en la utilización de redes neuronales; en este caso, se presentan los gráficos para 10 y 200 épocas para todos los modelos generados, mostrando que la más representativas son aquellas que tienen las mayores épocas, pues se puede apreciar de mejor manera cómo se comporta el proceso de entrenamiento; indicando en este caso, que los modelos que varían de 100 a 150 épocas la curva se estabiliza, disminuyendo de forma mucho más lenta hasta alcanzar el valor más bajo en las 200 épocas, indicando que el modelo podrían tener un mejor rendimiento.

Los tiempos de ejecución de los modelos fueron marcadamente más rápidos en *Pytorch* que en *TensorFlow*, pasando de pocos segundos en cualquiera de los modelos de *Pytorch* a más de 10 minutos en los modelos de 200 épocas cuando se emplea técnica *DownSampling* y de 30 minutos en el modelo de 200 épocas para *UpSampling* con la biblioteca de *TensorFlow*; sin embargo, los resultados en cuanto a exactitud, sensibilidad y especificidad de los modelos de *TensorFlow* fueron mejores en casi todos los casos, resultado que en el modelo de 10 épocas se alcanza ya mejores valores en las medidas mencionadas, mientras que con *Pytorch* se necesitan más épocas para lograr lo mismo.

Los resultados alcanzados con los datos para validar los modelos coincidieron con los resultados obtenidos en las matrices de confusión y con las diferentes medidas resultantes, tales como: la exactitud del modelo, su sensibilidad, la especificidad y los valores VP+ y VP-. Basados en estas medidas se puede concluir que los valores de exactitud mejores

se alcanzan a partir de los modelos con 150 épocas, en casi todos los casos, tal y como se muestran en los gráficos de las funciones de pérdida.

Los modelos de redes neuronales empleando las bibliotecas *PyTorch* y *TensorFlow* son útiles en el pronóstico de enfermedades cardiovasculares, por su capacidad de detectar patrones complejos que pueden no ser evidentes mediante métodos tradicionales. Estos modelos están compuestos por una gran cantidad de datos y variables predictivas que permiten establecer relaciones entre los factores de riesgo y los resultados de salud. Por otro lado, brindan la posibilidad del uso de las GPU que facilitan la disminución del tiempo de cálculo en la ejecución de los modelos.

5. Agradecimientos

Los autores agradecen el apoyo del Laboratorio de Automatización del DISCA-IIMAS y a la DGAPA-UNAM Proyecto PAPIIT IG101322, por la infraestructura y el financiamiento para el desarrollo y elaboración de este trabajo.

6. Referencias

[1] Litjens, G., Kooi, T., Ehteshami Bejnordi, B., Adiyoso Setio, A. A., Ciompi, F., Ghafoorian, M., van der Laak, J., van Ginneken, B., Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, *42*, 60-88. https://doi.org/10.1016/j.media.2017.07.005

[2] Poplin, R., Varadarajan, A. V., Blumer, K., Liu, Y., McConnell, M. V., Corrado, G. S., Peng, L., Webster, D. R. (2018). Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, *2* (3), 158-164. https://doi.org/10.1038/s41551-018-0195-0

[3] Cood., E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13* (6), 377-387. https://doi.org/10.1145/362384.362685

[4] del Castillo Collazo, N., Contreras Arvizu, J. A., Durán Ortega, A. J. (2024). Uso de las técnicas *DownSampling* y *UpSampling* para abordar el desequilibrio de datos en la predicción de personas propensas a sufrir accidentes cerebrovasculares. *Revista de Investigación en Tecnología de la Información (RITI),* 12 (25), 66-78. https://doi.org/10.36825/RITI.12.25.007

[5] Novac, O. C., Chirodea, M. C., Novac, C. M., Bizon, N., Oproescu, M., Stan, O. P., Gordan, C. E. (2022). Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network. *Sensors*, 22, 1-23. https://doi.org/10.3390/s22228872

[6] Pytlak, K. (2022). *Indicators of Heart Disease*. https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease

[7] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.

[8] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang. L., Bai, J., Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 33rd International Conference on Neural Information Processing Systems, Vancouver BC Canada. https://dl.acm.org/doi/10.5555/3454287.3455008

[10] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning*. 12th USENIX conference on Operating Systems Design and Implementation, Savannah, GA, USA. https://dl.acm.org/doi/10.5555/3026877.3026899

[11] Nielsen, M. (2015). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/

[12] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep Learning. *Nature, 521*, 436-444. https://doi.org/10.1038/nature14539

[13] Aldás, J., Uriel, E. (2017). *Análisis Multivariante aplicado con R* (2da Ed.). Ediciones Paraninfo.

[14] del Castillo Collazo, N. (2020). Predicción en el diagnóstico de tumores de cáncer de mama empleando métodos de clasificación. *Revista de Investigación en Tecnología de la Información (RITI), 8* (15), 96-104. https://doi.org/10.36825/RITI.08.15.009

[15] Ashraf, M., Ahmad, S. M., Ganai, N. A., Shah, R. A., Zaman, M., Khan., S. A., Shah. A. A. (2021). *Prediction of Cardiovascular Disease Through Cutting-Edge Deep Learning Technologies: An Empirical Study Based on TENSORFLOW, PYTORCH and KERAS*. En D. Gupta, A. Khanna, S. Bhattacharyya, A. E. Hassanien, S. Anand, A. Jaiswal (Eds.) International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing (pp. 239-255). Springer. https://doi.org/10.1007/978-981-15-5113-0_18

[16] Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, *25*, 65-69. https://doi.org/10.1038/s41591-018-0268-3

[17] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., Lungren, M. P., Ng, A. Y. (2017). CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv*. https://doi.org/10.48550/arXiv.1711.05225