Clasificación de ECG usando Redes Neuronales

J. Agustín Barrachina
Miembro Estudiantil IEEE
Instituto Tecnológico de Buenos Aires

***Abstract*—Se clasificaron latidos de un segmento de señal de un electrocardiograma según el tipo de arritmia. El algoritmo utilizado fue un perceptrón multicapa usando backpropagation como clasificador. Se utilizaron PCA y SOM para reducir la dimensionalidad de los datos.**

I. Introducción

El trabajo consistió en clasificar los latidos de un segmento de una señal de electrocardiograma de dos canales de 21 horas de duración según el tipo de arritmia.

El objetivo fue crear y entrenar un sistema que pueda reconocer y clasificar los distintos tipos de latidos.

II. Data Set

Se utilizaron las grabaciones de "MIT-BIH Long Term Database" [1] disponible en el repositorio PhysioNet [2].

La grabación incluye anotaciones que identifican la posición y tipo de cada uno de los latidos presentes en la misma.

La grabación se identificará por el número 14172, y el identificador de la base de datos es "ltdb" (Long Term DataBase). La misma presenta principalmente 4 tipos de latidos:

- **Normales**. Identificados por la letra 'N'.
- **Ventriculares prematuros** Identificados por la letra 'V'.
- **Supraventriculares prematuros** Identificados por la letra 'S'.
- **Nodales prematuros** Identificados por la letra 'J'.

Para el tratamiento de los datos se utlizó la librería de python "wfdb" [3].

A continuación se muestran la cantidad de latidos de cada tipo que poseen los datos.

```
('Numero de latidos: ', 66409)
set(['a', 'F', 'J', 'N', 'S', 'V', '~'])
a :  1
F :  1
J :  148
N :  58323
S :  1003
V :  6530
~ :  407
```

A. Reducción de Dimensionalidad

Se usaron dos técnicas distintas para reducir la dimensión de los datos y así acelerar y facilitar las operaciones de la red neuronal.

La primera técnica fue la utilización de *Principal Componen Analysis* (PCA). La teoría del mismo no se desarrollará en éste informe.

Otra técnica utilizada fue la implementación de una red *Self Organized Map* (SOM) que tampoco se desarrollará en este informe.

Finalmente se decidió por utilizar la implementación del PCA por un motivo puramente de tiempo de ejecución, ya que la red SOM requería mucho tiempo de cómputo y no obtenía resultados perceptiblemente superiores a los de la red PCA. Se truncó las dimensiones de PCA reduciendo los datos de cada latido de 250 a 32. En la figura 1 se muestra un ejemplo de los datos originales contra los datos "filtrados".

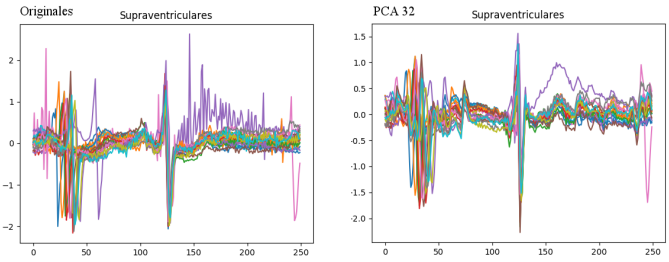

Fig. 1. Ejemplo de reducción de los datos utilizando PCA

III. Diseño de la Red Neuronal

Según Rezaul Begg et al. [4], la técnica más comúnmente utilizada de Redes Neuronales hasta el día de la fecha en procesamiento de ECG es el perceptrón multicapa (MLP) entrenada con backpropagation (Sección 2 Capítulo 3), por dicho motivo se utilizará éste mismo como el algoritmo para utilizar en éste trabajo.

El mismo trabajo posee una tabla (figura 2) que presenta diferentes aplicaciones de redes neuronales en el ámbito de cardiología. Lamentablemente ninguno de los casos es aplicable a nuestro caso ya que suelen ser diferenciación entre si y no sobre la probabilidad de tener AMI. Sin embargo, la mayoría de los casos usan una configuración de MLP de una sola capa oculta con lo cual en este trabajo se utilizará esa misma configuración. Normalmente además se recomienda utilizar solo una capa ya que casi con seguridad no será necesario más capas.

Table 1. Applications of ANNs in cardiology

Authors	Objective	Input	Output	Type of ANN	Results/Remarks
Azuaje et al. (1999)	Coronary disease risk	Poincare plots Binary coded Analog coded	Two, three, and five levels of risk	MLP (144,576,1024/ 70,200,500/30,100, 200/2,3,5)	Even binary coded Poincare plots produce useful results
Baxt (1991); Baxt et al. (1996)	Acute Myocardial infarction (AMI)	20 Items including symptoms, past history, and physical and laboratory findings, e.g., palpitations, Angina, Rales or T-wave inversion	AMI yes/no	MLP (20/10/10/1)	MLP had higher sensitivity and specificity in diagnosing a myocardial infarction than physicians
Dorffner and Porenta (1994)	Coronary artery disease	45 Parameters of the Stress redistribution scintigram (5 segments times 3 views times 3 phases – stress, rest, washout)	Normal/ pathological	1) MLP 2) RBFN 3) Conic section function networks	MLP can be among the poorest methods Importance of initialization in MLP and CSFN
Kennedy et al. (1997)	AMI	39 items, e.g., Blood and ECG parameters derived to 53 binary inputs	AMI yes/no	MLP (53/18/1)	Trained ANN at least as accurate as medical doctors
Kukar et al. (1997)	Ischaemic heart disease (IHD)	77 parameters; IHD data for different diagnostic levels (e.g. signs, symptoms, exercise ECG, and myocardial scintigraphy, angiography)	Classification	1) (semi-)naive Bayesian classifier 2) Backpropagation learning of neural networks 3) Two algorithms for induction of decision trees (Assistant-I and -R) 4) k-nearest neighbors	Improvements in the predictive power of the diagnostic process. Analyzing and improving the diagnosis of ischaemic heart disease with machine learning
Mobley et al. (2004)	Coronary stenosis	11 personal parameters	Stenosis yes/no	MLP (11/36/1)	Specificity 26% Sensitivity 100%
Mobley et al. (2000)	Coronary artery stenosis	14 angiographic variables	Stenosis yes/no	MLP (14/26/1)	ANN could identify patients who did not need coronary antiography

Fig. 2. Redes Neuronales Aplicadas en Cardiología según [4]

Si bien existen varios trabajos sobre la elección de la cantidad de elementos re la red oculta como [5] o [6], no existe una receta única e infalible sobre cuántos nodos darle a la capa oculta. Para elegirlos se utilizaron las siguientes reglas *RoT* (Rule of Thumb) que se describirán a continuación.

1) *Errar por más y no por menos:* Unos nodos extra no es probable que causen daño a la convergencia de la red, se los puede pensar como un exceso de capacidad. Mientras que usar pocos nodos pueden afectar a la convergencia.
2) *Basado en la entrada y la salida:* Utilizar una cantidad de nodos que se encuentren entre la cantidad de nodos de la entrada y la cantidad de nodos de la salida. (Jeff Heaton autor de *"Introduction to Neural Networks for Java"*)

Utilizando estas dos reglas se eligió utilizar la expresión $nodos_{hidden} = (nodos_{entrada} + nodos_{salida})/1.5$.

Para el algoritmo de aprendizaje se utilizó una librería [7] que implementa redes neurnales feed-forward (Redes Multicapa) permitiendo una variedad de algorimos de aprendizajes de backpropagation.

A. Set de Entrenamiento

En un primer momento se entrenaba la red con los primeros casos de latidos. Para dicho set de entrenamiento, el error aumentaba a medida que los datos estaban más alejados temporalmente de los mismos, es decir, si se tomaban los primeros 10 casos del set de datos para el entrenamiento, el conjunto de datos a partir del onceavo hasta el veinteavo mostraba grandes resultados, mientras que los datos rondando el número mil poseía resultados muy poco satisfactorios.

Para contrarrestar este problema se tomaron los datos de entrenamiento de forma aleatoria. Ésto logró mejorar mucho el desempeño pero a coste de tener mayor varianza en los resultados. Además el caso de seleccionar dos veces los mismos datos puede ocurrir con el código actual.

IV. Overfitting

Se utilizó un test set para ir verificando la evolución de la red. Como no se deseaba darle prioridad a ningún latido sobre los demás se utilizó la misma cantidad para cada categoría. Siendo el máximo numero de latidos de la categoría con menor cantidad de latidos 148. Se tomó 100 datos de cada categoría como test set.

La implementación de la red presentó overfitting. Como se puede ver en los resultados listados a continuación, el error del set de datos del test set comienza a incrementar.

```
Error: 0.290525167175 Epoch: 1000
Error: 0.252432576731 Epoch: 2000
Error: 0.233333077085 Epoch: 3000
Error: 0.226507241997 Epoch: 4000
Error: 0.227502496085 Epoch: 5000
Error: 0.230580316278 Epoch: 6000
Error: 0.233771065888 Epoch: 7000
Error: 0.236764878811 Epoch: 8000
Error: 0.239612203297 Epoch: 9000
Error: 0.242385089641 Epoch: 10000
```

La librería utilizada [7] permite el uso de dropout para poder reducir el overfitting. El resultado a continuación muestra el caso para un dropout del 50%.

```
Error: 0.330238968525 Epoch: 1000
Error: 0.254160001896 Epoch: 2000
Error: 0.214160846981 Epoch: 3000
Error: 0.193992225022 Epoch: 4000
Error: 0.185100235993 Epoch: 5000
Error: 0.180218777466 Epoch: 6000
Error: 0.176489777988 Epoch: 7000
Error: 0.173452841416 Epoch: 8000
Error: 0.171893196978 Epoch: 9000
Error: 0.172405423062 Epoch: 10000
```

Se puede observar que el dropout mejoró notoriamente el overfitting haciendo que el mismo sea de pasados las 9000 iteraciones. Se realizó entonces una simulación de montecarlo para distintos valores de dropout a fin de verificar el valor óptimo del mismo y comparar su impacto en el resultado. El resultado se muestra en la tabla I.

TABLE I
MSE para distintos valores de dropout

Dropout	MSE	Epoch
0%	0.2233 ± 0.0070	5011 ± 443
50%	0.1973 ± 0.0045	7141 ± 393
60%	0.1946 ± 0.0029	7662 ± 324
70%	0.1979 ± 0.0038	6639 ± 1428
80%	0.1872 ± 0.0033	7310 ± 483
90%	0.1953 ± 0.0040	6725 ± 491

Se puede ver como a medida que aumenta el dropout, el error cuadrático medio se va reduciendo hasta llegar al 80%, tras lo cual comienza a aumentar de vuelta. Se elige entonces un dropout de 80% para este trabajo.

A. *Modificación de la Librería: Early Stop*

Sin embargo el overfitting no se eliminó por completo y existe un número óptimo para el cual conviene no seguir entrenando la red. Por dicho motivo, se modificó la librería para poder realizar *early stop*. La implementación permite elegir la cantidad de iteraciones con un aumento del error antes de abandonar el entrenamiento (que se denominará *tolerancia del early stop*).

En un principio se hizo que esas iteraciones fueran consecutivas exclusivamente, es decir, que el error aumente sin interrupción durante esa cantidad de iteraciones. Ésto generó que con valores muy altos de tolerancia, jamás ocurriera el early stop. Ésto luego se cambió para que la diferencia entre los últimos cambios haya sido superior a la tolerancia.

Se realizó una simulación para obtener el valor que mejor consiga adecuarse al valor real del mínimo error.

TABLE II
TOLERANCIA DE EARLY STOP

EARLY STOP TOLERANCE	MSE ERROR	EPOCH ERROR
1	0.032274 ± 0.002175	6898 ± 867
5	0.000329 ± 0.000140	4575 ± 876
10	0.000178 ± 0.000153	2795 ± 1361
15	0.000002 ± 0.000001	386 ± 168
20	0.000001 ± 4e-07	561 ± 308
24	0.000001 ± 5.4e-07	-46 ± 218
28	4.5e-07 ± 1.5e-07	-253 ± 144
30	2.6e-07 ± 1e-07	-223 ± 102
32	6.5e-07 ± 2e-07	-389 ± 107
34	0.000001 ± 4e-07	-477 ± 191
40	7.6e-07 ± 1.4e-07	-306 ± 131

Se dejó el error de los epochs como diferencia bruta (y no su cuadrado) para poder observar como a partir de cierto punto, el algoritmo termina después del mínimo de la función. Ésto demuestra como una tolerancia baja de *early stop* hace que el la red deje de entrenar antes de alcanzado el punto óptimo mientras que colocar una tolerancia muy alta hace que el mismo termine mucho después. Si se desea observar el error cuadrático se puede ver la figura 3.

La figura 4 muestra el MSE error que figura en la tabla II.

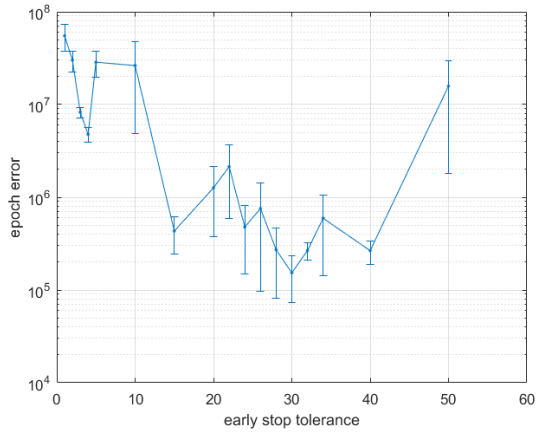

Fig. 3. Error MSE de la cantidad de epochs realizados de más o de menos antes del early stop

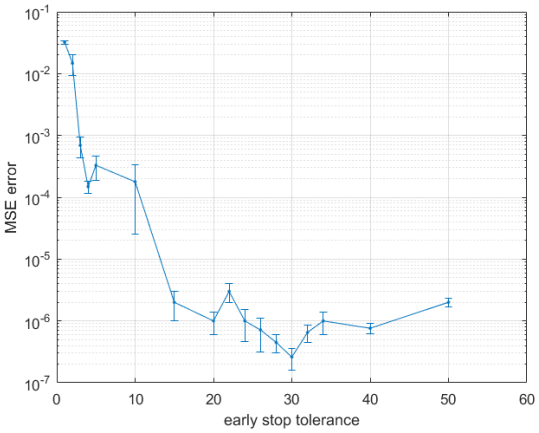

Fig. 4. Error MSE conseguido con early stop contra el óptimo de la iteración

V. IMPLEMENTACIÓN DE LA RED

La siguiente figura (figura 5, levemente modificada de la figura 6 de [4]) muestra el diagrama básico de la implementación del proyecto.

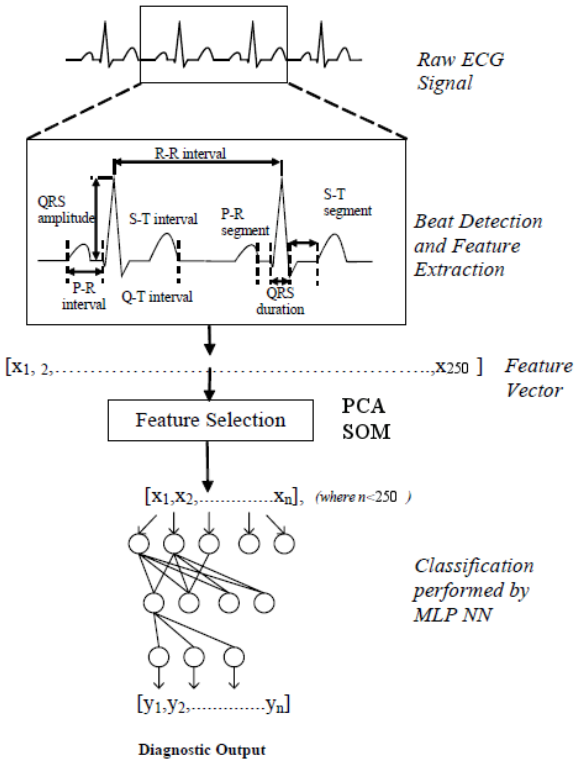

Fig. 5. Diagrama general del proyecto

En primer lugar se extrae y se extraen varios vectores de dimensión 250 de cada caso de latido. Luego se reduce su dimensión utilizando PCA o SOM a una dimensión menor. Finalmente se aplica la red multicapa obteniendo la salida y_i donde cada valor corresponde a una clasificación posible ($0 \leq y_i \leq 1$ que representa la probabilidad de que el latido corresponda a dicha clasificación).

- Reducción de Dimensión usando PCA 32
- Configuración 32 - 24 - 4
- Función de activación: Sigmoidea
- Cantidad de latidos de entrenamiento: 10
- Cantidad de latidos de test set: 100
- Dropout = 80%
- Tolerancia Early Stop 30

VI. Análisis de los Resultados

Para analizar los resultados se usaron dos parámetros que se los denominó *aciertos* y los *hallados*.

El primero cuenta la cantidad de *aciertos* de cada decisión, es decir, si se dijo que cierto numero de latidos corresponden a un categoría, cuántos de ellos realmente eran de esa categoría.

Por ejemplo, si encuentro 100 latidos ' N' y 90 de ellos lo eran, el porcentaje de aciertos será del 90% lo cual parece ser un número muy aceptable. Sin embargo, si se alimentó la red con 900 latidos, solo el 10% fueron hallados, haciendo que el desempeño de la red sea indeseado.

Por dicho motivo, para este segundo caso, se midió la cantidad de elementos de esa categoría encontrados o *hallados* sobre el total real.

Además se colocaron los *falsos positivos* (decidir que el latido posee arritmia cuando no tenía) y *falsos negativos* (decidir que los latidos estaban bien cuando no lo estaban).

También se guardó la cantidad total de *aciertos* y el *porcentaje ponderado de enfermos* el cual contiene el promedio de la cantidad de *hallados* de cada categoría de arritmia (Ventriculares, Supraventriculares y Nodales) dándole igual peso a cada uno (para que pesen igual los 6000 casos de Ventriculares a los 148 casos de los Nodales).

Se le debe dar especial atención a los *falsos negativos* y al *porcentaje ponderado de enfermos* ya que son los casos en los que se estaría cometiendo un error clínico de mayor importancia.

VII. Resultados

A continuación se muestra una iteración utilizando todos los latidos del set de datos.

```
Normales totales 58320
Normales hallados: 34768
Aciertos: 34535.0
Errores: 233.0
Porcentaje correcto: 99.33 %
Porcentaje hallados: 59.22 %
_____________________________

Ventriculares totales 6529
Ventriculares hallados: 7569
Aciertos: 6394.0
Errores: 1175.0
Porcentaje correcto: 84.48 %
Porcentaje hallados: 97.93 %
_____________________________

Supraventriculares totales 1003
Supraventriculares hallados: 14034
Aciertos: 698.0
Errores: 13336.0
Porcentaje correcto: 4.97 %
Porcentaje hallados: 69.59 %
_____________________________

Nodales Prematuros totales 148
Nodales Prematuros hallados: 9629
Aciertos: 118.0
Errores: 9511.0
Porcentaje correcto: 1.23 %
Porcentaje hallados: 79.73 %
_____________________________

Aciertos totales: 41745 (63.00 %)
Porcentaje enfermos ponderado: 82.42 %
Falsos negativos: 233
Falsos positivos: 23785
```

Se puede observar un resultado muy satisfactorio en cuanto a los *falsos negativos* y el *porcentaje de enfermos ponderados*, lo cual es muy satisfactorio ya que como se dijo en la sección VI, estos dos parámetros serán los de mayor relevancia.

Los falsos negativos corresponden a un 0.35% del total latidos lo cual es un resultado muy satisfactorio.

Sin embargo se destacan dos numeros muy bajos, los mismos corresponden al porcentaje de aciertos de los latidos Supraventriculares y Nodales que son inferiores al 10%. Estos numeros bajos son reducen notablemente los aciertos totales (a un 63%) y la cantidad de falsos positivos ya que corresponde a los casos que se detectó como alguno de esas dos categorías pero el latido correspondía a los normales.

Se probó entonces realizar una simulación con 2000 casos de cada categoría con el objetivo que haya números similares para cada caso.

```
Normales totales 2000
Normales hallados: 1386.0
Aciertos: 1056.0
Errores: 330.0
Porcentaje correcto: 76.19 %
Porcentaje hallados: 52.80 %
-----------------------------------------
Ventriculares totales 2000
Ventriculares hallados: 2002.0
Aciertos: 1960.0
Errores: 42.0
Porcentaje correcto: 97.90 %
Porcentaje hallados: 98.00 %
-----------------------------------------
Supraventriculares totales 1003
Supraventriculares hallados: 1196.0
Aciertos: 582.0
Errores: 614.0
Porcentaje correcto: 48.66 %
Porcentaje hallados: 58.03 %
-----------------------------------------
Nodales Prematuros totales 148
Nodales Prematuros hallados: 567.0
Aciertos: 99.0
Errores: 468.0
Porcentaje correcto: 17.46 %
Porcentaje hallados: 66.89 %
-----------------------------------------
Aciertos totales: 3697 ( 71.00 %)
Porcentaje enfermos ponderado: 74.31 %
Falsos negativos: 330.0
Falsos positivos: 944.0
```

Como se esperaba, los valores de porcentaje correcto de los Supraventriculares aumentó notablemente y mejor fue su aproximación cuanto más se acercaban los datos a 1000 para cada caso. Sin embargo, no se obtuvo un buen resultado para los Nodales Prematuros. Incluso bajando el número de datos a menos de 150 el porcentaje de aciertos apenas si logró superar el 50%.

La cantidad de aciertos totales fue superior al 70% en la mayoría de los casos que se entrenó con 2000 datos (recordar que como se usan sets se entrenamientos aleatorios en casa caso V, el mismo suele tener varianza).

Los latidos que más satisfactoriamente se detectó fueron los Ventriculares ('V') en los cuales se obtuvo bajo varias condiciones, un resultado superior al 97% tanto de *aciertos* como de cantidad total *hallada*.

Para ésto último se le puede dar una explicación gráfica a partir de la red implementada SOM. El resultado de la misma se encuentra en la figura 6. En dicha figura se puede ver que los valores de los ventriculares están claramente definidos por lo cual es de esperar que se obtengan tan buenos resultados. Los normales y ventriculares poseen error pero en líneas generales tiene cierto grado de diferenciación. Los nodales prematuros por otro lado se superponen mucho con los demás resultados haciendo que en varios casos se interprete los demás casos (normales, ventriculares y supraventriculares) como un caso de nodales prematuros.

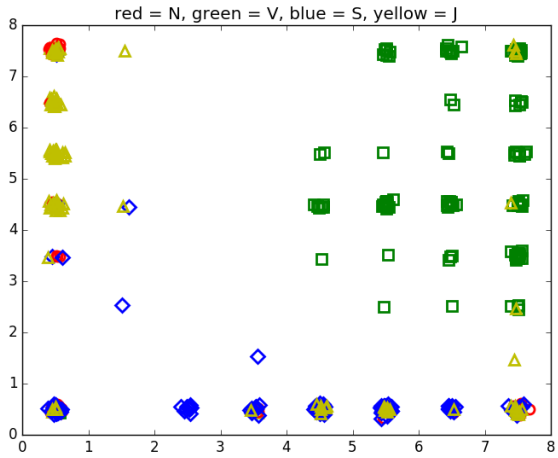

Fig. 6. Resultados de una red SOM sobre algunos datos

VIII. Posibles Mejoras

Se podría implementar el método denominado como *Cascade Correlation* (Fahlman et al. 1990) que permite ajustar dinámicamente la estructura de la red. La misma comienza con una red minimalista (perceptrón simple) y luego va ajustando los nodos de la capa oculta durante el entrenamiento.

El dropout probó ir reduciendo el error cuadrático medio, sin embargo se podría realizar más mediciones para encontrar el valor óptimo del mismo ya que solo se iteró con 5 posibles valores y realizando mayor cantidad de mediciones podría encontrarse un valor que mejor represente los datos. También, se utilizó el test set para verificar los errores siempre igual y con 100 ejemplares de cada categoría. Se podrían variar estos parámetros para verificar la estabilidad de los resultados frente a distintos test sets.

Se debería seleccionar los datos de entrenamiento de una forma que asegure que los mismos no se repitan. Además se podría usar un set de entrenamiento con una cantidad distinta para cada categoría en relación a su cantidad de latidos en el set de datos y verificar como se modifican los resultados con ello (que aspectos se mejoran y cuales no).

La librería permite definir funciones costo propias. Ésto se podría aprovechar para definir una función de costo propia

que le otorgue menor peso a los casos Normales respecto a los demás.

La librería ofrece muchas variantes de backpropagation que no fueron probadas ni investigadas. Sería interesante inspeccionar sobre dichas variantes.

Sería interesante eliminar los Nodales Prematuros y considerar solo los otros tres casos. Los resultados obtenidos con SOM parecen indicar que los mayores problemas derivan por los mismos por lo que quizás mejore mucho la eficiencia de la red si solo se trabajara con 3 categorías.

REFERENCES

[1] The mit-bih long term database. [Online]. Available: https://physionet.org/physiobank/database/ltdb/

[2] G. L. H. J. I. P. M. R. M. J. M. G. P. C.-K. S. H. P. P. Goldberger AL, Amaral LAN and PhysioNet, "Components of a new research resource for complex physiologic signals," Circulation 101(23):e215-e220, jun 2000, [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full].

[3] C. Xie. wfdb-python. [Online]. Available: https://github.com/MIT-LCP/wfdb-python

[4] R. S. Rezaul Begg, Joarder Kamruzzaman, *Neural Network in Healthcare: Potential and Challanges*. Australia: IDEA Group Publishing, 2006.

[5] G. C. Lawrence, S. and A. Tsoi, "What size neural network gives optimal generalization? convergence properties of backpropagation," *Technical Report UMIACS-TR-96-22 and CS-TR-3617*, 1996.

[6] A. Elisseeff and H. Paugam-Moisy, "Size of multilayer networks for exact learning: analytic approach," *Advances in Neural Information Processing Systems 9, Cambridge, MA: The MIT Press*, pp. 162–168, 1997.

[7] J. Grimnes. (2016) Nimblenet. python-neural-network. [Online]. Available: https://github.com/jorgenkg/python-neural-network