Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

SISTEMA PARA LA DETECCIÓN DE ARRITMIAS CARDÍACAS BASADO EN REDES NEURONALES LSTM

Israel Rivera Zárate
Instituto Politécnico Nacional-CIDETEC
irivera@ipn.mx

Miguel Hernández Bolaños
Instituto Politécnico Nacional-CIDETEC
mbolanos@ipn.mx

Patricia Pérez Romero
Instituto Politécnico Nacional-CIDETEC
promero@ipn.mx

Abstract

Long Short-Term Memory (LSTM) is a variation of recurrent network (RNN) able to analyze variable length data sequences. It's a powerful class of computational model that correlates information in the short or long-term of chronologically dependent data. LSTMs can use their internal state (memory) and several internal gates to be able to know what to forget and what to remember. In the present work it is shown how LSTMs can perform classification that allows, based on training, the identification of arrhythmias with respect to a healthy patient.

Palabras clave: Red LSTM, Procesamiento de Secuencias, Lenguaje Python.

Un ciclo es el conjunto de fenómenos cardíacos que se producen desde el comienzo de un latido del corazón hasta el comienzo del latido siguiente. En un ciclo normal las dos aurículas se contraen mientras los dos ventrículos se relajan; así mismo, mientras los dos ventrículos se contraen se relajan las dos aurículas según indica W. Hassan et al. (2017).

El término sístole se refiere a la fase de contracción en tanto que el término diástole alude a la fase de relajación. En un ciclo cardíaco consta de una sístole y una diástole de ambas aurículas y una sístole seguida de una diástole de ambos ventrículos, como se observa en la Figura 1.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

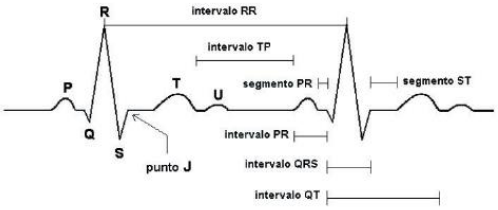
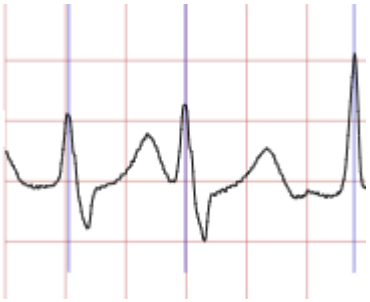


Figura 1. Rivera, Hernández y Pérez (2020). Ondas, segmentos e intervalos del ECG.

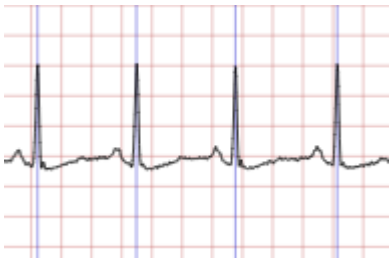
Por otra parte, de acuerdo con Sodi D. (2010). Una arritmia cardíaca es toda irregularidad en el ritmo natural del corazón. Las alteraciones en la función cardíaca se producen no como consecuencia de un músculo anormal, sino por un ritmo cardiaco anormal. Las causas de las arritmias son una de las siguientes alteraciones del sistema de ritmicidad-conducción del corazón o una combinación de ambas:

1. Ritmos sinusales anormales.
2. Desplazamiento del marcapaso desde el nodo sinusal.
3. Bloqueos en diferentes puntos de la propagación del impulso a través del corazón.
4. Vías anormales de transmisión del impulso a través del corazón.
5. Generación espontánea de impulso anormal en casi cualquier parte del corazón.

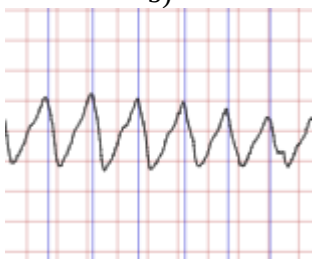
Ejemplos de estas señales se presentan en la figura 2.



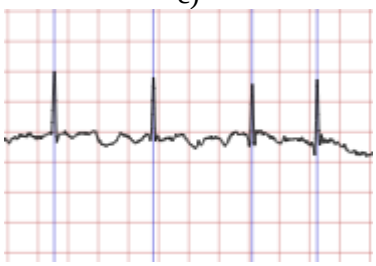
a)



b)



c)



d)

Figura 2. Rivera et al. (2020). a) Taquicardia supra ventricular, b) Taquicardia ventricular, c) Aleteo ventricular (flutter) y d) Fibrilación auricular.

Principales arritmias:

A. Taquicardia supra ventricular paroxística (TSVP). Son contracciones rápidas causadas por una señal eléctrica que comienza por encima de los ventrículos y se manifiesta como una serie de pulsos rápidos y rítmicos con latidos fuertes y enérgicos. El paroxismo significa que aparece de vez en cuando. La frecuencia cardiaca puede ir de 140-220 latidos/min.

B. Taquicardia ventricular. Se trata de una taquicardia ventricular que tiene al menos cuatro latidos ectópicos consecutivos que persisten menos de 30 segundos.

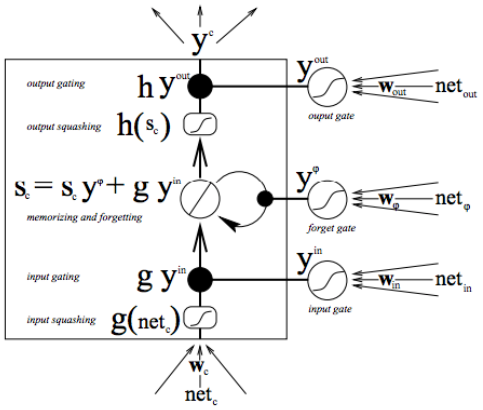
Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

C. Aleteo ventricular (Flutter). Es una taquicardia de origen ectópico auricular. Generalmente se origina mediante un circuito de reentrada en el ventrículo derecho. Produce latidos auriculares de 250-300 latidos/min y latidos ventriculares 75-100 latidos/min.

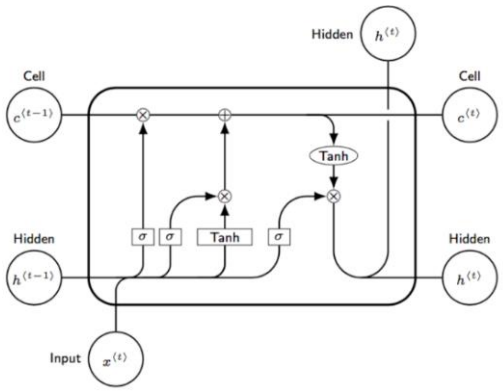
D. Fibrilación auricular. Se caracteriza por una actividad auricular desorganizada y muy rápida (350-600 latidos/min). Esto conduce a una respuesta irregular del ventrículo (150-200 latidos/min) y a un pulso irregular.

I. Red LSTM

Las redes LSTM (*long short-term memory*: memoria a corto y largo plazo) funcionan de forma similar a como el cerebro analiza las secuencias. Por ejemplo, si se analiza un texto; el cerebro se enfoca solamente en las palabras relevantes y se desecha el resto de la información. Como lo indica Bengio Y. et al (1994). Las redes LSTM son capaces de añadir o desechar la información que se considere relevante para el procesamiento de la secuencia. Comparada con una celda de red recurrente básica; la celda LSTM tiene una entrada y salida adicional, este elemento adicional se conoce como celda de estado: "*cell state*". Esta celda de estado es la clave del funcionamiento de las redes LSTM. La celda de estado es como una banda transportadora a la que se pueden añadir o donde se pueden remover datos que no se desean preservar en la memoria de la red. Para añadir o remover datos se utilizan varias compuertas como compuerta de olvido ("*forget gate*"): que permite eliminar elementos de la memoria; compuerta de entrada ("*update gate*"): que permite añadir nuevos elementos de la memoria; compuerta de salida ("*output gate*"): que permite crear el estado oculto actualizado. Estas compuertas son redes neuronales que funcionan como válvulas donde totalmente abiertas permiten el paso de la información y totalmente cerradas lo bloquean por completo. Ver figura. 3.



a)



b)

Figura 3. a) Operaciones básicas en la red recurrente LSTM, b) Bloques de la red recurrente LSTM. http:// colah.github.io/posts/2015-08-Understanding-LSTMs.

El descenso del gradiente es el algoritmo de optimización utilizado para el entrenamiento; donde se busca minimizar la función de error de los valores de salida obtenidos con relación a los esperados respecto a los parámetros de la red (producto de derivadas parciales empleando la regla de la cadena). Según indica Fei-Fei et al. (2017). El resultado total del error es la suma de las derivadas parciales calculadas en cada

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

intervalo time-step. El proceso termina cuando se logra ajustar los pesos de las neuronas respecto a la función de coste o pérdida, de ahí que se llama “backpropagation through time” (BTT).

II. Metodología

La metodología adoptada en el presente proyecto consiste en 3 pasos esenciales:

[Paso 1:] Preprocesamiento.

[Paso 2:] Segmentación.

[Paso 3:] Aprendizaje, detección y clasificación.

Preprocesamiento

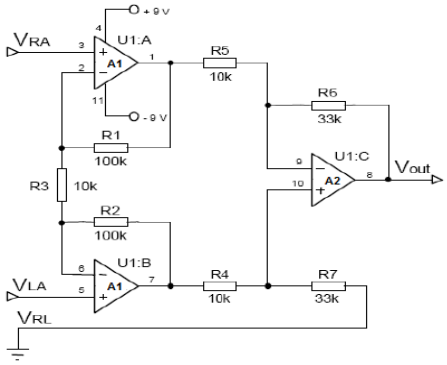
Se empleó el integrado AD8232 que es un conjunto de amplificadores y filtros diseñado para extraer, amplificar y filtrar pequeñas señales biopotenciales, como se muestra en figura 8 a). Así mismo se realizó un programa en Android para la adquisición de los valores digitales de la señal, lográndose visualizar en pantalla y almacenar en archivos binarios. Ver figura 4 b).

a)

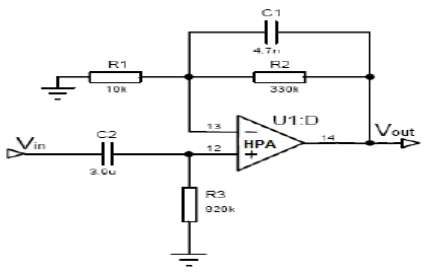
b)

Figura 4. a) Elementos del sistema: electrodos de Ag/AgCl, cables, tarjeta sparkfun, tarjeta Arduino UNO y b) Imagen obtenida en Smartphone bajo Android. Tomado de: www.sparkfun.com.

El AD8232 está dispuesto en la tarjeta Sparkfun de Analog devices donde se configuró el amplificador de instrumentación para brindar alta impedancia de entrada, baja ganancia en modo común y aislamiento eléctrico al paciente. Como lo indican Álvarez J., Herrera J. y Pérez P. (2010). Se utilizaron filtros pasa banda y rechaza banda para eliminar las frecuencias por encima de los 100 Hz al no ser éstas de tipo cardiológico, así como las inferiores a 0.5 Hz debidas al potencial formado entre los electrodos y la piel (300 mV). Asimismo, se eliminaron las frecuencias cercanas a los 60 Hz para evitar el ruido inducido a través de la red eléctrica como se indica en la figura 5.

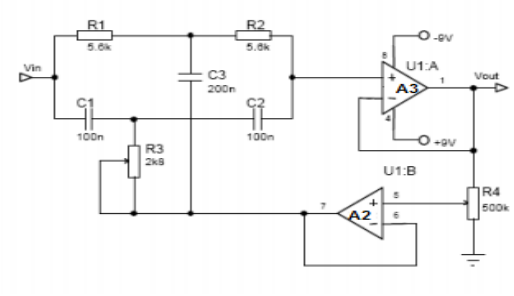


a)



b)

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957



c)

Figura 5. Rivera et al. (2020). Bloques configurados en el integrado AD8232: a) Amplificador de instrumentación, b) Filtro pasa banda y c) Filtro rechaza banda.

Segmentación

Se realizó un programa en Android que permite generar archivos binarios donde se almacenan las representaciones digitales de los complejos ECG obtenidos por el convertidor DAC de 10 bits de la tarjeta Arduino de la fase previa. Los valores obtenidos son accedidos mediante su número de muestra y son el dato de entrada de la etapa de clasificación basada en red neuronal. La estructura definida para la red neuronal recurrente de la siguiente etapa se requiere de 300 muestras por complejo PQRS por evaluar estableciendo especial atención al segmento ST que es donde ocurren las principales variantes que distinguen a las señales con presencia de arritmia.

Aprendizaje, detección y clasificación

Desde 1975 los laboratorios del Beth Israel Hospital en Boston y el MIT (Instituto Tecnológico de Massachusetts) han llevado a cabo una investigación sobre el análisis de arritmias. Uno de los principales productos importantes de ese esfuerzo es la Base de datos de arritmias MIT-BIH que ha sido distribuida desde 1980. La Base de datos fue el primer material de prueba que se volvió un estándar generalmente disponible para la evaluación de detectores de arritmias y se ha utilizado con ese propósito, así como para la investigación básica de la dinámica cardiaca en más de 500 sitios en todo el mundo. Las señales del ECG han sido reunidas en 48 archivos que contienen 30 minutos cada uno de extractos de información de complejos PQRS obtenidos a partir de grabaciones de equipos Holter aplicados a cientos de pacientes. Los archivos están agrupados en dos grupos principales: la serie 100 y la serie 200.

La implementación de estas redes profundas se ha desarrollado en Python (https://www.python.org, versión 3.6.1) debido a que es un lenguaje de programación de alto nivel que consiste en una sintaxis sencilla de comprender e implementar, además es multiplataforma lo cual permite hacer ejecutable su código fuente entre varios sistema operativos; con la ayuda de las librerías gratuitas Keras (https://keras.io, versión 2.0.6) y Tensorflow (https://www.tensorflow.org, versión 1.2.1), ampliamente aceptadas por la comunidad y con un gran crecimiento en fiabilidad y potencial. El entorno de desarrollo empleado ha sido el *Notebook* de Jupyter (http://jupyter.org, versión 5.0.0).

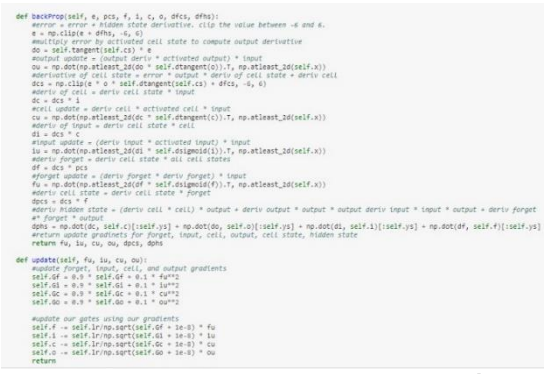
El modelo consta de dos capas: una capa LSTM de 256 neuronas para la entrada que entrega las secuencias y otra capa de 256 neuronas de salida a una función *softmax* que considera las probabilidades e indicará el carácter más probable a generar como predicción. La red utiliza *Dropout* con probabilidad del 20%, como se muestra en el fragmento de código (implementación) de la Figura 6.

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

```
In [ ]: class LSTM:
    # LSTM cell (input, output, amount of recurrence, learning rate)
    def __init__ (self, xs, ys, rl, lr):
        #input is word length x word length
        self.x = np.zeros(xs+ys)
        #input size is word length + word length
        self.xs = xs + ys
        #output
        self.y = np.zeros(ys)
        #output size
        self.ys = ys
        #cell state intialized as size of prediction
        self.cs = np.zeros(ys)
        #how often to perform recurrence
        self.rl = rl
        #balance the rate of training (learning rate)
        self.lr = lr
        #init weight matrices for our gates
        #forget gate
        self.f = np.random.random((ys, xs+ys))
        #input gate
        self.i = np.random.random((ys, xs+ys))
        #cell state
        self.c = np.random.random((ys, xs+ys))
        #output gate
        self.o = np.random.random((ys, xs+ys))
        #forget gate gradient
        self.Gf = np.zeros_like(self.f)
        #input gate gradient
        self.Gi = np.zeros_like(self.i)
        #cell state gradient
        self.Gc = np.zeros_like(self.c)
        #output gate gradient
        self.Go = np.zeros_like(self.o)

    #activation function to activate our forward prop, just like in any type of neural network
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    #derivative of sigmoid to help computes gradients
    def dsigmoid(self, x):
        return self.sigmoid(x) * (1 - self.sigmoid(x))

    #tanh! another activation function, often used in LSTM cells
    #having stronger gradients: since data is centered around 0,
    #the derivatives are higher. To see this, calculate the derivative
    #of the tanh function and notice that input values are in the range [0,1].
    def tangent(self, x):
        return np.tanh(x)

    #derivative for computing gradients
    def dtangent(self, x):
        return 1 - np.tanh(x)**2

    #lets compute a series of matrix multiplications to convert our input into our output
    def forwardProp(self):
        f = self.sigmoid(np.dot(self.f, self.x))
        self.cs *= f
        i = self.sigmoid(np.dot(self.i, self.x))
        c = self.tangent(np.dot(self.c, self.x))
        self.cs += i * c
        o = self.sigmoid(np.dot(self.o, self.x))
        self.y = o * self.tangent(self.cs)
        return self.cs, self.y, f, i, c, o
```

Figura 6. Rivera et al. (2020). Fragmento de código: Implementación.

Se utilizó el optimizador gradiente descendente calculando las derivadas parciales y aplicando la regla de la cadena. Ver figura 7. Adicionalmente se actualizan los pesos y los gradientes de cada compuerta (forget, input, output y cell state), Observar que el algoritmo considera la acumulación total del error luego de calcular el error local en cada elemento de la celda LSTM y de cada time-step.



```
def backProp(self, e, pcs, f, i, c, o, dfcs, dfhs):
    #error = error + hidden state derivative. clip the value between -6 and 6.
    e = np.clip(e + dfhs, -6, 6)
    #multiply error by activated cell state to compute output derivative
    do = self.tangent(self.cs) * e
    #output update = (output deriv * activated output) * input
    ou = np.dot(np.atleast_2d(do * self.dtangent(o)).T, np.atleast_2d(self.x))
    #derivative of cell state = error * output * deriv of cell state + deriv cell
    dcs = np.clip(e * o * self.dtangent(self.cs) + dfcs, -6, 6)
    #deriv of cell = deriv cell state * input
    dc = dcs * i
    #cell update = deriv cell * activated cell * input
    cu = np.dot(np.atleast_2d(dc * self.dtangent(c)).T, np.atleast_2d(self.x))
    #deriv of input = deriv cell state * cell
    di = dcs * c
    #input update = (deriv input * activated input) * input
    iu = np.dot(np.atleast_2d(di * self.dsigmoid(i)).T, np.atleast_2d(self.x))
    #deriv forget = deriv cell state * all cell states
    df = dcs * pcs
    #forget update = (deriv forget * deriv forget) * input
    fu = np.dot(np.atleast_2d(df * self.dsigmoid(f)).T, np.atleast_2d(self.x))
    #deriv cell state = deriv cell state * forget
    dpcs = dcs * f
    #deriv hidden state = (deriv cell * cell) * output + deriv output * output * output deriv input * input * output + deriv forget
    #* forget * output
    dphs = np.dot(dc, self.c)[:self.ys] + np.dot(do, self.o)[:self.ys] + np.dot(di, self.i)[:self.ys] + np.dot(df, self.f)[:self.ys]
    #return update gradients for forget, input, cell, output, cell state, hidden state
    return fu, iu, cu, ou, dpcs, dphs

def update(self, fu, iu, cu, ou):
    #update forget, input, cell, and output gradients
    self.Gf = 0.9 * self.Gf + 0.1 * fu**2
    self.Gi = 0.9 * self.Gi + 0.1 * iu**2
    self.Gc = 0.9 * self.Gc + 0.1 * cu**2
    self.Go = 0.9 * self.Go + 0.1 * ou**2

    #update our gates using our gradients
    self.f -= self.lr/np.sqrt(self.Gf + 1e-8) * fu
    self.i -= self.lr/np.sqrt(self.Gi + 1e-8) * iu
    self.c -= self.lr/np.sqrt(self.Gc + 1e-8) * cu
    self.o -= self.lr/np.sqrt(self.Go + 1e-8) * ou
    return
```

Figura 7. Rivera et al. (2020). Fragmento de código: Entrenamiento.

III. Pruebas y Resultados

Se logró contar con la participación de la alumna Lesley Abigail Rivera Garduño que es estudiante de Tercer semestre de la carrera de Médico Cirujano y Partero de la ESM del IPN; quien nos apoyó en la adquisición de las señales electrocardiográficas de tres individuos que decidieron participar en las pruebas voluntariamente los cuales han sido diagnosticados con patologías cardiacas y que contaban con electrocardiograma reciente (mismos que se muestran en papel milimétrico en la figura 8 respectivamente). Los casos comprenden las situaciones siguientes:

A. Varón de 49 años electrocardiograma rítmico, en ritmo sinusal, con frecuencia cardiaca de 80 latidos/min. Intervalos PR y QT normales, sin alteraciones en el segmento ST.

B. Mujer de 72 años con ECG revelando hipertrofia ventricular izquierda. Q patológica, disminución del segmento ST.

C. Mujer de 70 años con ECG mostrando extrasístole ventricular, onda P positiva en todas las derivaciones menos en AVR, seguidas de QRS estrecho, intervalo PR normal de 0.15 s. QT corregido normal de 400 ms, segmento ST isoeléctrico.

Se conectaron los electrodos del sistema a los tres individuos y se procedió a la obtención de la señal ECG en el dispositivo móvil las cuales se muestran en la parte inferior del electrocardiograma correspondiente, esto se puede observar en la figura 8.

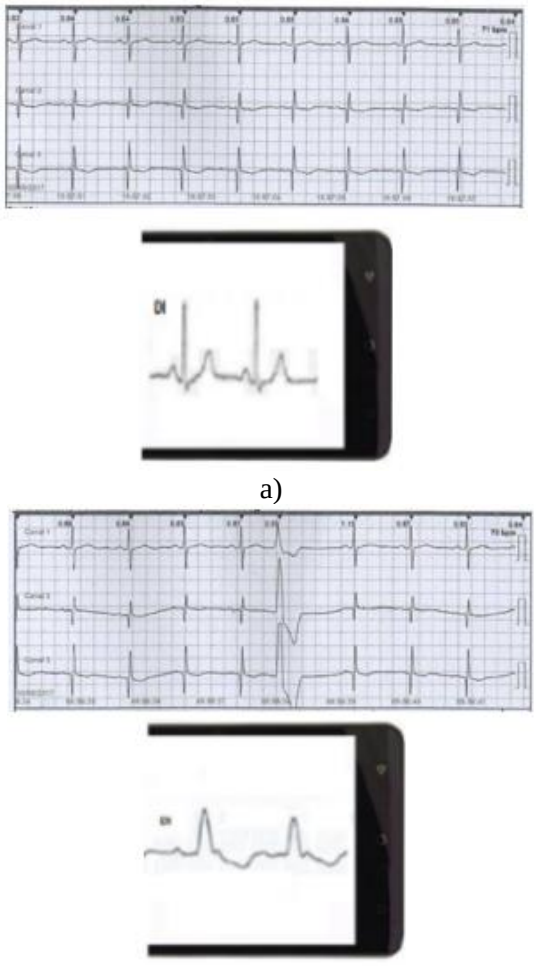
En todos los casos señalados se trabajaron los tres tipos de formato de archivos de datos de entrada, esto es: archivos de 500 ciclos cardiacos con 300 y 1000 muestras de entrada a la red neuronal recurrente, así como

Humanidades, Tecnología y Ciencia, del Instituto Politécnico Nacional
Certificado de Reserva de Derechos al Uso Exclusivo del Título: 04-2010-0326124-14000-203 ISSN: 2007-1957

el caso de archivos de 1000 ciclos con 300 muestras. El resultado obtenido aparece en la tabla 2.

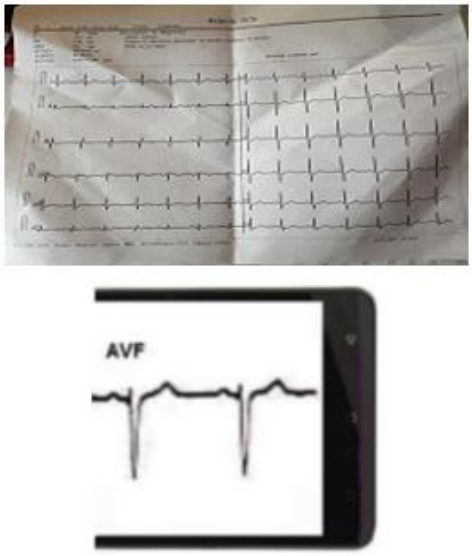
Tabla 2. Comparación de resultados obtenidos por el clasificador por diferente número de muestras y ciclos cardíacos.

Número Ciclos / muestras	**500 / 300 muestras**	**500 / 1000 muestras**	**1000 / 300 muestras**
Señal normal	100%	90%	95%
Señal con arritmias	100%	85%	90%



a)

b)



c)

Figura 8. Rivera et al. (2020). Señal ECG pacientes (Papel milimétrico, dispositivo móvil del sistema implementado): a) Normal, b) Extrasístole ventricular y c) Hipertrofia Ventricular izquierda.

IV. Conclusiones

Con base en los resultados obtenidos se puede mencionar que se logró una precisión superior al 90%. Se demostró que el empleo de las redes neuronales recurrentes permite clasificar satisfactoriamente las señales con características de arritmia respecto de las normales. Se pudo observar que las redes recurrentes operan en mejor condición para un número reducido de puntos de entrada para el caso particular del orden de 300; esto considerado en función de que la frecuencia de muestreo de la base de datos es del orden de 360 Hz y que un ciclo cardíaco comprende alrededor de 300 muestras. Quedo manifiesto que conforme se empleó un mayor número de muestras el sistema exhibió fallas en la clasificación hasta un grado del orden del 85% de aciertos en el caso de arritmias y del 90 % en señal normal. Se considera que el comportamiento observado en el porcentaje de

fallo obedece principalmente al grado de complejidad de la señal de arritmia. Se concluye que las redes recurrentes LSTM cuando analizan series temporales muy largas y que no mantienen un patrón cíclico muy establecido puede ocurrir que el error tiende muy rápido a cero o bien muy rápido a infinito. Las LSTM en sus diversas celdas de memoria estructurada por sus compuertas de olvido, entrada y salida respectivamente lograron ajustar sus coeficientes alrededor de las 200 épocas propuestas logrando contener el error ya que ahora se tienen sumas de errores y no productos cuya derivada no provoca explosión del error.

Con la realización de este sistema a decir de los mismos individuos bajo prueba resulta de gran apoyo para el monitoreo de la actividad cardíaca y más particularmente en los casos de padecer alguna cardiopatía dada su característica portátil de uso y dado que el empleo del *smartphone* es ya de uso cotidiano. Asimismo, se considera al sistema como una propuesta de bajo costo económico considerando que en suma el sistema sin considerar el *smartphone* asciende a un costo aproximado de $1,000.00 pesos.

En comparación con otros métodos existentes para la detección automática de arritmias se probó y verificó el funcionamiento del sistema resultando de comportamiento igual o más competitivo según se observó el desempeño en la evaluación de los individuos de cardiopatías bien conocidas, con electrocardiograma recientemente practicado.

V. Referencias

W. Hassan, S. Saleem, & A. Habib (2017). Classification of normal and arrhythmic ECG using wavelet transform based template-matching technique, JPMA. The Journal of the Pakistan Medical Association, Vol. 67, No. 6, pp. 843.

Sodi D. (2010). Electrocardiografía clínica. Análisis deductivo. Méndez Editores.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. 5, pp. 157–166.

Fei-Fei et al. (2017). Convolutional Neural Networks (CNNs / ConvNets). URL: http://cs231n.github.io/convolutional-networks (visitado 20-02-2020).

Álvarez Cedillo, J. A., Herrera Lozada, J. C., & Pérez Romero, P. (2010). Sistema informático para análisis de cardiopatía Holter. *Revista Polibits*(41), 59-66