Generar melodías con LSTM a partir de archivos MIDI

Eduardo Díaz Del Castillo, Astrid Hurley, y Abdiel Alveo

Abstract-- Genetic algorithms are a class of evolutionary algorithms, somewhat like computer models based on natural biological evolution that perfectly complement evolutionary programming, evolutionary strategies, and genetic programming.

LSTM networks are a special kind of recurrent networks. Their main characteristic is that data can persist by entering loops in the network diagram. This makes it possible to remember previous states and use that information to decide what the next data would be.

I. INTRODUCCIÓN

Los algoritmos genéticos son una clase de algoritmos evolutivos, algo así como modelos de computación basados en la evolución biológica natural que se complementan perfectamente con la programación evolutiva, la evolución de estrategias y la programación genética.

Las redes LSTM son un tipo de especial de redes recurrentes. Su principal característica es que la data puede persistir ingresando bucles en el diagrama de la red. Esto hace pueda recordar estados previos y utilizar esa información para decidir cual sería el siguiente dato.

II. REDES LSTM

A. *Definición*

[1]Las redes LSTM o por sus siglas en inglés, Long short term memory; son un tipo de redes neuronales recurrentes que consiste en múltiples puertas y son capaces de aprender a largo plazo debido a su estructura tienen la capacidad de recordar información por periodos largos de tiempo, olvidad información innecesaria y exponer la información dada.

Este tipo de redes depende mucho de su resultado actual para tomar en cuenta el resultado posterior. Si el resultado actual es menos que el posterior, entonces el posterior no es tomado en cuenta y el programa sigue buscando un resultado actual óptimo. Si no encuentra ninguno, simplemente se queda con el mejor.

En una neurona LSTM hay tres puertas: puerta de entrada, puerta del olvido y puerta de salida. Estas puertas determinan si se permite o no una nueva entrada. Ya que elimina la información que no es importante o se deja que afecta a la salida en el paso de tiempo actual. [2]

III. PLANTEAMIENTO DEL PROBLEMA

Los estudiantes de música necesitan un programa que les ayude a inspirarse, para la confección de nuevas melodías. Actualmente solo cuentan con una base de datos, con los sonidos de las teclas de un piano y quieren usar eso como la base del sistema. Por lo que necesitamos una solución apropiada que se pueda aprovechar las notas adquiridas para que pueda complementar el estudio de la música generando automáticamente melodías.

IV. EXPERIMENTACIÓN

[3]El experimento fue basado en un blog de towards data science.

Para la experimentación del laboratorio se utilizo la herramienta de Google, Colab. La cual fue aumentada a Colab Pro para la disminución del entrenamiento.

A. *Recursos del laboratorio*

El repositorio en classroom donde nos proporcionaban los recursos del proyecto, no fueron utilizados en su totalidad. Ya que muchos de los archivos que nos proporcionaban están en el formato WAV.

Nuestro grupo opto por utilizar archivos con extensión MIDI, ya que estos archivos pueden medir fácilmente los tiempos, las notas, vibratos, etc. Mientras que los demás son solo audios.

Para el entrenamiento de la data utilizamos canciones de chopin en formato MIDI. Se pueden utilizar cualquier tipo de música siempre y cuando este en formato MIDI.

B. *Primer entrenamiento*

[4]Dentro de lo explicado en el video del laboratorio 1, el primer entrenamiento fue de 75 clases y cada una de ella tenían una duración de 1h 30 min cada una aproximadamente (a veces menos, a veces más). Al parecer el Colab no puede estar mas de 12h seguidas ejecutándose sin ningún

movimiento en el documento. Por consiguiente, el primer entrenamiento fue bajado a 20 clases, las cuales solo 16 lograron salvarse.

```
[ ]  1 filepath = "/content/drive/My Drive/{epoch:02d}-{loss:.4f}.h5"
     2
     3 checkpoint = ModelCheckpoint(filepath, monitor='loss',verbose=0,
     4                              save_best_only=True,mode='min')
     5
     6 callbacks_list = [checkpoint]
     7 model.fit(network_input, network_output, epochs=20, batch_size=64,
     8           callbacks=callbacks_list)
```

```
Epoch 1/20
868/868 [==============================] - 5150s 6s/step - loss: 4.9609
Epoch 2/20
868/868 [==============================] - 5136s 6s/step - loss: 4.9538
Epoch 3/20
868/868 [==============================] - 5189s 6s/step - loss: 4.9524
Epoch 4/20
868/868 [==============================] - 5146s 6s/step - loss: 4.9511
Epoch 5/20
868/868 [==============================] - 5210s 6s/step - loss: 4.9494
Epoch 6/20
868/868 [==============================] - 5269s 6s/step - loss: 4.9495
Epoch 7/20
868/868 [==============================] - 5241s 6s/step - loss: 4.9493
Epoch 8/20
526/868 [=================>............] - ETA: 33:55 - loss: 4.9457
```

Ilustración 1 - Ajuste del primer entrenamiento. Fuente: propia

Por lo que este entrenamiento fue descartado. Ya que no cumplió con las 20 clases.

Las primeras partituras de este entrenamiento lucen de esta manera:

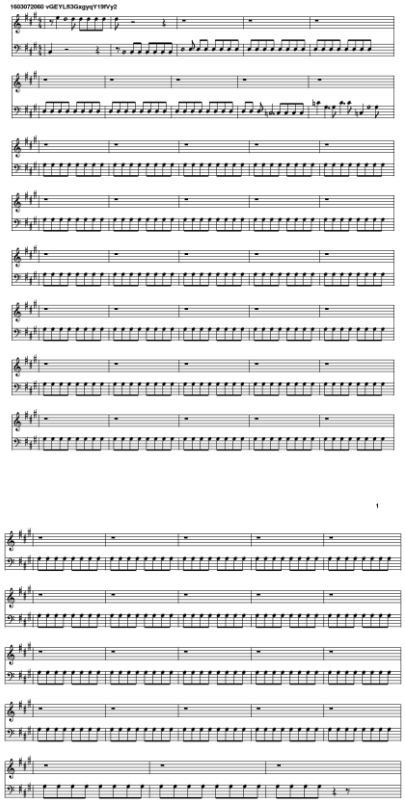


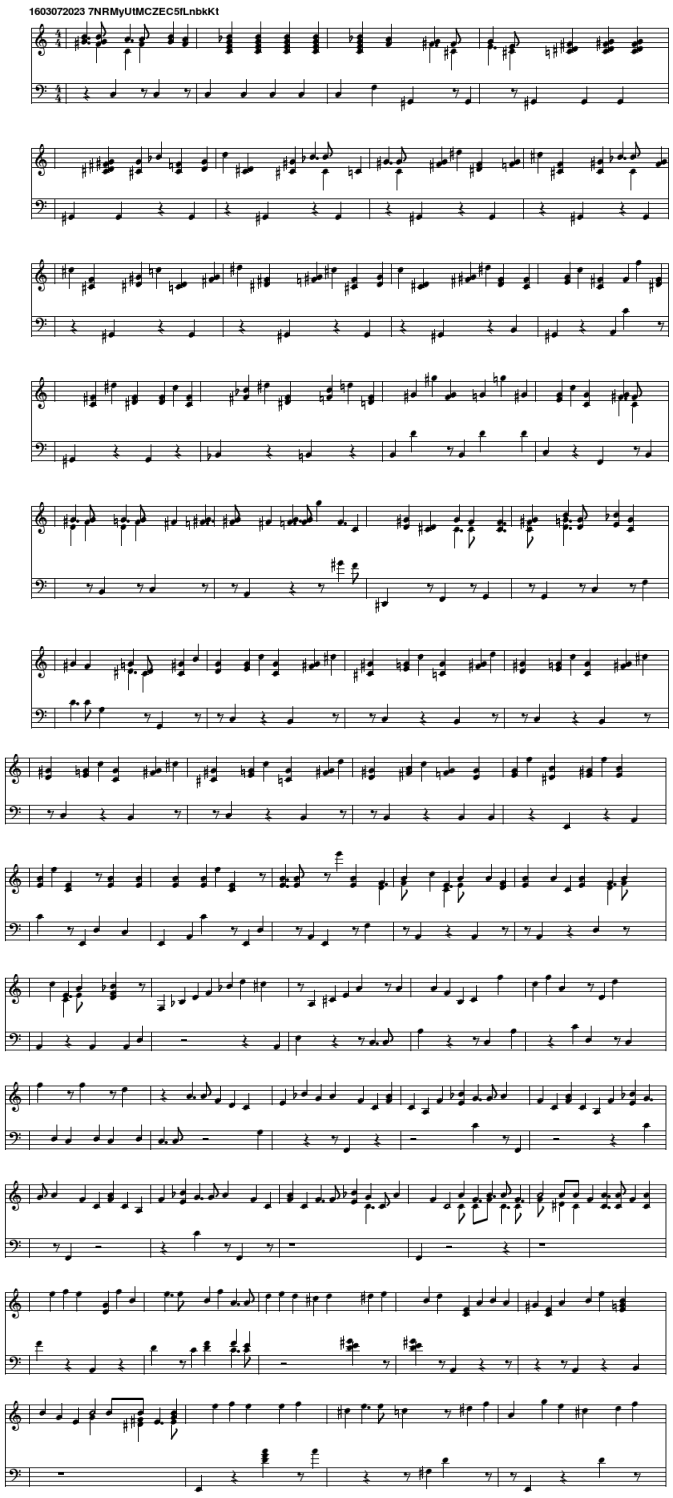
Ilustración 2 - Salida del primer entrenamiento - Fuente: propia

C. Segundo entrenamiento

Durante este entrenamiento se tomó la decisión de pagar por el Google Colab, ya que nos garantiza una mejor GPU, RAM, y espacio en disco. Los resultados fueron notables, cada clase terminaba en 71 segundos. Por consiguiente, se realizó un entrenamiento de 200 para que la data fuera entrenada correctamente.

Al llegar a la clase 94, notamos que no siguió guardando data. Ya que llego a un estado optimo y todas las clases siguientes eran mayores que la clase 94 y no las tomaba en cuenta.

Las partituras de este entrenamiento lucen de esta manera:



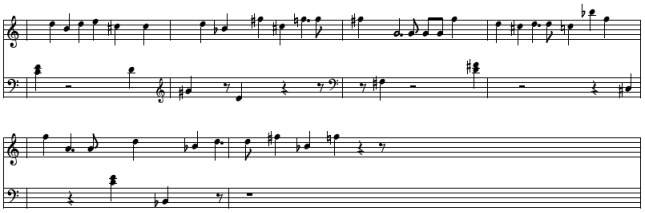


Ilustración 3 - Salida del entrenamiento 2. Fuente: propia

IX. CONCLUSIÓN

El segundo entrenamiento funciono, inclusive podemos notar una amplia diferencia en la variedad de notas que se encuentran en las partituras. Este método es lento, pero sin lugar a duda es el más optimo siempre y cuando se tenga con los recursos para realizarlo.

Esta implementación del LSTM puede entrenar cualquier tipo de melodía. Por ejemplo: las melodías propuestas en el planteamiento del problema pueden pasar a archivos MIDI con algún software de sonido e importar la data para su posterior entrenamiento.

V. REFERENCIAS

[1] May Pech, Fernando;, «Riostecnm,» 2018. [En línea]. Available: http://rios.tecnm.mx/cdistribuido/recursos/DLScr/PLN.html#%28part._.R.N.N_.L.S.T.M%29. [Último acceso: 23 octubre 2020].

[2] Torres, Jordi;, «Torres ai,» 22 septiembre 2019. [En línea]. Available: https://torres.ai/redes-neuronales-recurrentes/#Long-Short_TermMemory. [Último acceso: 24 octubre 2020].

[3] S. Skúli, «towardsdatascience,» 7 diciembre 2017. [En línea]. Available: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-keras-68786834d4c5. [Último acceso: 24 octubre 2020].

[4] E. Díaz del Castillo, A. Hurley y A. Alveo, «Youtube,» 23 octubre 2020. [En línea]. Available: https://www.youtube.com/watch?v=JrvRdA4jwyM. [Último acceso: 24 octubre 2020].