Compositor automático de música

A. Nassiff, M. Rolé, G. Tessi
Supervisores: M. Gerard, T. Molas Giménez, L. Vignolo

***Resumen*— Varias características en las redes neuronales las hacen deseables para la composición de música. Estas características incluyen la habilidad de aprender relaciones complejas a partir de ejemplos y que hacen generalizaciones razonables en situaciones para las que no fueron entrenadas. El entrenamiento consiste en exponer la red a ejemplos de melodías y ajustar su representación interna para que concuerde con los ejemplos de entrada. El método usado para presentar las piezas musicales a la red va a afectar en gran medida las características que la red va a extraer. Este artículo examina una estrategia que involucra crear música usando redes neuronales dinámicas.**

***Palabras clave*— generación, música, composición, predicción, multicapa**

I. Introducción

Los estudiantes de música a menudo aprenden a componer mediante ejemplos de obras clásicas o contemporáneas. Esta estrategia o enfoque tiene gran similitud con las técnicas de entrenamiento para redes neuronales artificiales, lo que conlleva a que las mismas pueden ser utilizadas para predecir con cierto grado de exactitud datos futuros a partir de datos pasados.

La música puede ser caracterizada por diferentes aspectos ya sea ritmo, melodía, género o estilo musical, rango y tono de las diferentes notas, entre otras. Los archivos MIDI tienen una estructura similar a la de una partitura como se ilustra en la figura 1, lo cual permite extraer todas las características antes mencionadas en forma sencilla. El presente trabajo se enfoca en desarrollar una metodología que permita la composición de música en forma automática, utilizando como base piezas musicales compuestas con el nivel de creatividad de los seres humanos.

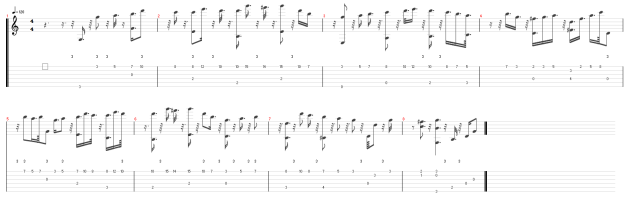

Fig. 1. Partitura del archivo MIDI “jesu.mid”.

Las redes neuronales dinámicas tienen en cuenta el comportamiento dependiente del tiempo, lo cual permite el modelado de la no linealidad y dinamismo de los componentes de un sistema. Son muy útiles en el análisis y modelado de series temporales, como la música [1]. Se utilizará una red neuronal con una estructura similar a la de un *auto-encoder*. Los *auto-encoders* no son redes neuronales dinámicas por naturaleza, sino compresores o codificadores de información con un comportamiento estático [2] [3]. De todas maneras, con pequeñas modificaciones, la red se puede adaptar para que sea dinámica recibiendo una porción de la salida actual como entrada de la iteración siguiente [4].

Sistemas basados en entrenamiento supervisado son aplicados para aprender las regularidades estructurales y estadísticas de la música. Las redes neuronales pueden entrenarse con dos o más melodías para combinar sus ideas. Esto se considera como una combinación a nivel de red [5].

Estudios anteriores han intentado diferentes métodos y técnicas para la composición de música, con resultados prometedores [1] [4] [5].

A fin de poder evaluar los resultados finales y obtener conclusiones, se realizarán encuestas al público en general sobre algunos de los resultados obtenidos por la metodología propuesta. Existen diferentes clases de medidas las cuales cuantifican las características de las melodías compuestas por la red, pero no serán consideradas debido a que no existen escalas con límites bien establecidos para ellas, lo cual convierte a las medidas en subjetivas [1] [6] [7].

II. Materiales

Para la realización del presente trabajo, se utilizaron piezas musicales de una sola pista en formato MIDI y el toolbox de MatLab para manipulación y creación de MIDIs [8], de forma que pueda ponerse a prueba el método a desarrollar.

Estas grabaciones han sido obtenidas de diferentes bases de datos y fuentes en Internet.

III. Metodología

El método consta de tres etapas, referidas a la obtención de las entradas de la red, que luego serán utilizadas para la generación de las diferentes melodías, finalizando en la producción de una pieza musical a partir de las salidas de la red.

Pista	Canal	Notas	Volumen	Tiempo Inicio	Tiempo Fin
1	0	99	43	0,9779	1,6112
1	0	68	67	1,2005	1,3737
1	0	75	69	1,4115	1,5299
1	0	88	71	1,625	1,8268

Fig. 2. Matriz de características del archivo MIDI “jesu.mid”.

La primer etapa consiste en procesar los archivos MIDI, mediante las funciones y herramientas del toolbox, para obtener sus características en forma de matriz como se muestra en la figura 2. Dichas características se utilizarán para crear vectores de notas utilizando ventaneo temporal sobre la música, que consisten en las notas ejecutadas en dicho intervalo. Para poder manipular las piezas musicales, se optó por utilizar la estructura mostrada en la figura 3, que toma cada vector de notas como una fila de la matriz.

La segunda etapa consiste en el entrenamiento de la red neuronal con los vectores de notas obtenidos en la etapa anterior, y la generación de nuevas melodías a partir de un proceso con retroalimentación de las entradas.

	Nota 1	Nota 2	Nota 3	...	Nota N
Ventana 1	28	36	89	...	99
Ventana 2	89	71	75	...	0
Ventana 3	66	80	73	...	0
...	...				
Ventana K					

Fig. 3. Matriz de vectores de notas como filas.

La tercer etapa codifica los vectores de notas generados por la red en formato MIDI.

Estas etapas se detallan a continuación.

A. *Etapa 1 - Preproceso: Obtención de las entradas*

Las notas musicales son la característica principal, junto a su tiempo de inicio y fin. Para simplificar la metodología, no se tomaron en cuenta el tempo, la métrica ni el compás musical de los archivos MIDI a procesar, entre otras propiedades. En un principio, se descompone la música utilizando un método de ventaneo temporal con solapamiento (producido entre el final de la ventana anterior y el principio de la siguiente), el cual determina las notas que fueron ejecutadas en un determinado intervalo, generando así un vector de notas. La dimensión de este vector se define según la máxima cantidad de notas ejecutadas de entre todas las ventanas. Finalmente, se obtiene una matriz con vectores de notas como filas, en donde cada fila representa un intervalo de tiempo definido por el tamaño de la ventana y por el solapamiento; y cada columna es la nota ejecutada en dicho intervalo. Nótese que para los intervalos de tiempo en donde no se ejecuten el número máximo de notas detectado anteriormente, el vector de notas es completado con ceros, que es el equivalente musical a la ausencia de notas en la partitura.

B. *Etapa 2 - Proceso: Red neuronal dinámica*

Este proceso toma la matriz generada en la etapa anterior, concatenando una cierta cantidad de filas de la misma y formando el patrón de entrenamiento. La estructura de la red neuronal es similar a la de un *auto-encoder* [2], la cual tiene como objetivo aprender una representación comprimida (codificación) de un conjunto de datos, reduciendo la dimensionalidad del mismo y tomando sus características más importantes. Básicamente la estructura de este tipo de red viene dada por una capa de entrada de dimensión N, un conjunto de capas con un número de neuronas necesariamente menor a N, y una capa de salida de igual dimensión que la entrada, representada en la figura 4.

Se adaptó la estructura de este tipo de red, de manera que el tipo de entrenamiento sea supervisado utilizando la regla delta con propagación hacia atrás del error (*backpropagation*), la cual va ajustando los pesos en función del error cuadrático instantáneo entre la salida de la red y el mismo patrón de entrada de la misma iteración [5]. El criterio de corte del entrenamiento es por cantidad de épocas realizadas, elegido así por simplicidad ya que adoptar otros criterios de corte, como tolerancia del error, pueden no ser representativos o adecuados.

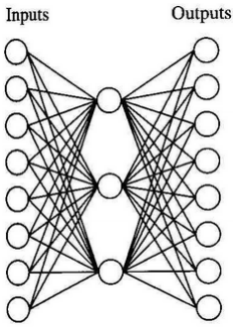

Fig. 4. Estructura del *auto-encoder*.

Una vez finalizado el entrenamiento, se exita a la red con una determinada cantidad de vectores de notas concatenados, que fueron utilizados durante el entrenamiento. Luego se toma una porción de la salida generada, equivalente a la dimensión del vector de notas, que será utilizada para la retroalimentación de las entradas, insertándose en la misma de manera similar a una cola y guardándose en una matriz [4]. Este procedimiento puede visualizarse en la figura 5.

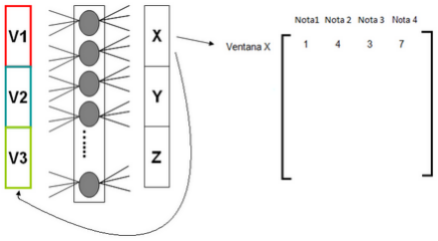

Fig. 5. Esquema de retroalimentación.

C. *Etapa 3 - Postproceso: Composición del MIDI*

Una vez generada la matriz con los vectores de notas en la etapa anterior, se los procesa elemento a elemento, generando otra matriz en donde cada nota será una fila, con los siguientes datos como columnas:

$$\begin{bmatrix} \texttt{numero de nota} \\ \texttt{tiempo inicial} \\ \texttt{tiempo final} \end{bmatrix}$$

El tiempo inicial se calcula según la posición del vector de notas en donde se encuentra dicha nota, y el tiempo final es la suma entre el tamaño de la ventana temporal y el tiempo inicial calculado anteriormente.

La composición del archivo MIDI es un proceso sencillo de realizar, ya que solo consta de armar una matriz similar a la de la figura 3 a partir de la matriz generada por la red. Un detalle a tener en cuenta es que las notas se ejecutan tantas veces como aparezcan en la matriz, y en consecuencia se pierde la capacidad de distinguir si la nota es ejecutada múltiples veces o si la ejecución es única y prolongada. Por simplicidad se supone que las notas que aparecen en intervalos de tiempo consecutivos, son ejecutadas una sola vez con una duración definida por el tiempo inicial de la primera y por el tiempo final de la última en esa consecución. Para detectar cuándo una nota está en intervalos consecutivos y poder establecer su duración, se utiliza el algoritmo mostrado en la figura 6.

A partir de este proceso, solo resta completar la matriz para que tenga la estructura de la figura 3 y poder componer

1. Ordenar la matriz según las notas y su tiempo inicial en forma ascendente.
2. Recorrer la matriz por filas.
   2.1. Verificar que la nota de la fila actual sea igual a la nota de la fila siguiente y que el tiempo inicial de la siguiente sea menor al tiempo final de la actual.
      2.1.1. Actualizar el tiempo final de la nota actual con el tiempo final de la nota siguiente.
      2.1.2. Eliminar la fila siguiente de la matriz.
   2.2. De lo contrario, avanzar a la siguiente fila.
3. Ordenar la matriz según el tiempo inicial.

Fig. 6. Algoritmo para la detección de notas consecutivas.

el archivo MIDI para su ejecución con los métodos provistos por el toolbox [8].

IV. Análisis de Resultados

La configuración de la red neuronal consta de una capa oculta con 30 neuronas, con funciones de activación sigmoideas. La capa de salida tendrá tantas neuronas como elementos tenga el patrón de entradas y funciones de activación lineales. Se establecen 300 épocas de entrenamiento. Todas las piezas musicales utilizadas en las pruebas corresponden a un mismo género musical y son ejecutadas por un mismo instrumento, en este caso el piano.

Para los casos más generales se realizó una evaluación de los tiempos de proceso del método. El sistema de prueba esta constituído por un procesador x86 Intel Pentium Dual Core P6200 a 2,13 GHz, con memoria RAM DDR3 de 2 GB a 1066 MHz, y el programa está compilado con MinGW para el lenguaje C++. Los resultados obtenidos se presentan en la Tabla I.

TABLA I
Resultados de los tiempos de cómputo para diferentes tamaños de ventana y solapamientos.

Música	Tiempo de CPU (seg.)	Tamaño de ventana (seg.)	Solapamiento (seg.)
jesu.mid	7,51	0,3	0,0
jesu.mid	11,70	0,3	0,1
jesu.mid	68,05	0,3	0,25
jesu.mid	5,92	0,5	0,0
jesu.mid	17,92	0,5	0,33
jesu.mid+testout.mid	20,10	0,3	0,1

Se puede apreciar que el solapamiento y el ancho de la ventana temporal son muy relevantes para el tiempo de ejecución del método.

En la figura 7 se muestra la partitura de la melodía generada por la red, a partir de las piezas musicales representadas en las figuras 1 y 8.

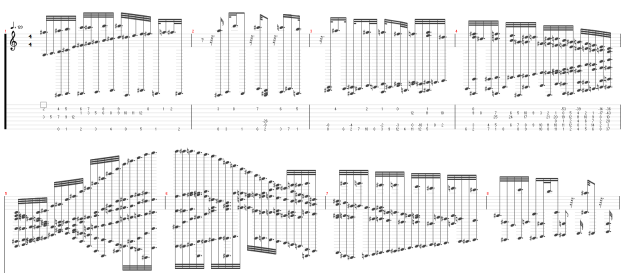

Fig. 7. Partitura del archivo MIDI generado.

Para poder determinar las bondades del método utilizado se realizó una encuesta a 74 personas sobre 4 melodías con

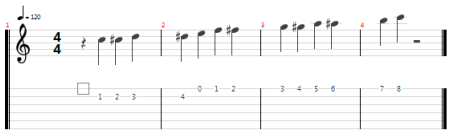

Fig. 8. Partitura del archivo MIDI “testout.mid”.

diferentes características: la primera está compuesta solo de una melodía; la segunda y la tercera están compuestas por las mismas piezas musicales de entrada pero se varió el tamaño de la ventana temporal; y la cuarta es una concatenación de tres melodías diferentes. Los resultados se muestran en la Tabla II. Como un conjunto de notas no puede significar nada para un oyente pero, sin embargo, otro se puede mostrar alerta ante dicho conjunto, se determinaron escalas propias de valores para obtener una medida de las bondades de la metodología anteriormente presentada. Las escalas de valores de calificación se establecieron como: Malo, Bueno, Muy Bueno, Excelente; con el siguiente grado de severidad: 1, 3, 5 y 7, respectivamente.

TABLA II
Resultados de la encuesta realizada.

	Música 1	Música 2	Música 3	Música 4
Malo	14	12	34	14
Bueno	35	34	18	23
Muy Bueno	14	16	10	17
Excelente	5	6	6	14
Puntaje (%)	24,72	26,05	19,87	29,36

A continuación se presentan diferentes pruebas realizadas.

A. *Normalización de vectores de notas a la octava dominante*

Una octava es el intervalo que separa dos sonidos cuyas frecuencias fundamentales tienen una relación de dos a uno, y contiene doce semitonos. Se hicieron pruebas llevando todas las notas de cada vector de notas a la octava dominante del mismo, esto es identificar la octava dominante a partir de la mayor cantidad de notas que pertenezcan a alguna octava en dicho vector. La normalización fue llevada a cabo solamente en los vectores de entrada de la red neuronal, luego solamente en los vectores de salida de la red, y por último en ambos a la vez. Los resultados obtenidos no fueron los esperados, ya que la melodía generada se ubicaba en una octava o en un rango acotado de dos octavas o menos, perdiendo capacidad de progresión y variación musical. La misma también tendía a progresar las escalas hacia la octava siguiente pero en lugar de ello, empezaba nuevamente en la primer nota de la octava actual. Esta prueba fue realizada a partir del archivo “jesu.mid” y la partitura del resultado se presenta en la figura 9.

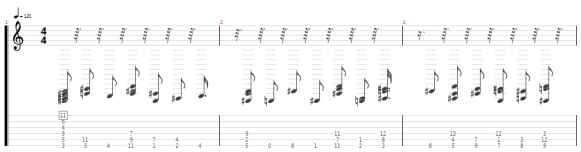

Fig. 9. Fragmento de la partitura normalizada.

B. Variación del tamaño y solapamiento de ventanas temporales

Al variar el tamaño y las posiciones de las ventanas temporales, se puede definir lo que se va a exponer a la red, ya sea una porción temporal particular, un compás o bien la pieza musical completa. Cuando las ventanas tenían un tamaño muy grande, más de 4 segundos, la red solo podía construir ventanas con múltiples notas, todas siendo ejecutadas a la vez, y no tenía capacidad de progresión de las notas ni escalas musicales. A medida que se reduce el tamaño de la ventana, la red obtiene gradualmente y en proporción capacidad de variación, por lo que empiezan a aparecer escalas y progresiones muy variadas. Adoptar una ventana muy pequeña, menor al orden de los 0,1 segundos, genera melodías sin una estructura predecible y totalmente diferente a las piezas musicales utilizada durante el entrenamiento. El solapamiento de las ventanas no es un factor determinante, ya que más o menos solapamiento produjo resultados muy similares, siempre manteniendo fijos el tamaño de las ventanas y utilizando las mismas melodías para cada una de las experiencias realizadas.

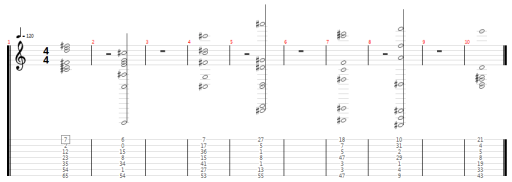

Fig. 10. Partitura del archivo MIDI generado con ventana de 3s.

C. Concatenación de varias piezas musicales

Tomando melodías aisladas, las generadas por la red tendían a imitar sus características, esto es surgían variaciones solo en las octavas de las piezas musicales originales. A partir de concatenar dos o más melodías diferentes, las generadas contenían una gran capacidad de progresión a lo largo del tiempo obteniendo resultados muy diversos. Por ejemplo, si la melodía es tan sencilla como una escala musical, lo que genera la red tiende a adoptar esta forma en conjunción con las otras piezas musicales adicionales. Esto se aprecia en la figura 7, generada a partir de los archivos "jesu.mid" y "testout.mid".

D. Adición de vectores con notas aleatorias

Al agregar entre uno y tres vectores con notas aleatorias, en el rango de valores desde 21 a 108 (rango audible para el ser humano), se aprecian variaciones en la sucesión de notas que mantienen cierto grado de relación con las notas anteriores [1]. La partitura de este resultado se puede apreciar en la figura 11.

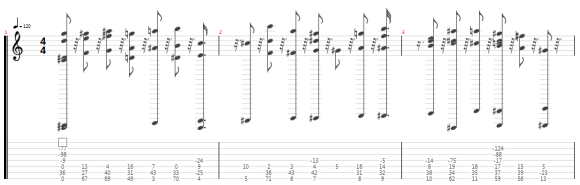

Fig. 11. Fragmento de la partitura generada con notas aleatorias.

V. Conclusiones y Trabajos Futuros

Pese a la subjetividad a la hora de decidir qué música es buena y qué música es mala, la encuesta realizada permite tener un criterio estadístico para poder concluir sobre las bondades del método aquí presentado. De esta manera, se concluye que las melodías generadas han tenido buena aceptación entre el público en general.

La melodía que tuvo mayor puntuación fue generada por una red entrenada con tres piezas musicales diferentes, en comparación a las demás que contaron con dos o una sola pieza durante su entrenamiento. Esto verifica lo analizado anteriormente, que a mayor cantidad de melodías concatenadas, se producen otras estadísticamente mejores.

Como trabajos futuros, se propone estudiar las variaciones musicales producidas por la red a partir de piezas musicales de diferentes géneros y estilos, así como también variar los compases y tiempos de las mismas. También es posible cambiar el tipo de entrenamiento, la estructura o el tipo de red neuronal, con el objetivo de verificar si se producen mejores o peores resultados que los presentados en este artículo o si hay conclusiones diferentes a las aquí expuestas, sin descartar realizar una optimización en función del tiempo de cómputo y espacio de memoria. Por otro lado, sería beneficioso involucrar en el proyecto a algún experto en el arte de la música con el objetivo de validar la metodología y sus resultados.

Referencias

[1] A. Coca y L. Zhao, "Generation of composed musical structures through recurrent neural networks based on chaotic inspiration," *IEEE*, 2011.

[2] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," *Workshop on Unsupervised and Transfer Learning*, 2012.

[3] G. F. Harpur, "Low entropy coding with unsupervised neural networks," Disertacin doctoral, Queens'College - University of Cambridge, Febrero 1997.

[4] D. Corrêa y S. Abib, "Composing music with bptt and lstm networks: Comparing learning and generalization aspects," *IEEE International Conference on Computational Science and Engineering*, 2008.

[5] J. P. Lewis, "Creation by refinement: A creativity paradigm for gradient descent learning networks," *Digital Sound Laboratory - New York Institute of Technology*.

[6] L. Hofmann-Engl, "An evaluation of melodic similarity models," *Chameleon Group Online Publication*, 2005.

[7] T. Eerola y A. C. North, "Expectancy-based model of melodic complexity," *University of Leicester*, 2000.

[8] T. Eerola y P. Toiviainen, "Mir in matlab: The midi toolbox," *University of Jyväskylä, Finland*, 2004.