Trabajo Práctico Inteligencia Artificial

Generar melodías con una RNN-LSTM

Asignatura

Inteligencia Artificial

Jefe de cátedra

Gustavo Javier Meschino

JTP

Matías Yerro

Integrantes

Ávalos, Wenceslao

Lapiana, Santiago Nicolás

Sosa, Santiago Fabián

Click para ver repositorio

Índice

INTRODUCCIÓN Y MARCO TEÓRICO..... 2

METODOLOGÍA..... 6

RESULTADOS..... 14

CONCLUSIONES..... 14

INTRODUCCIÓN Y MARCO TEÓRICO

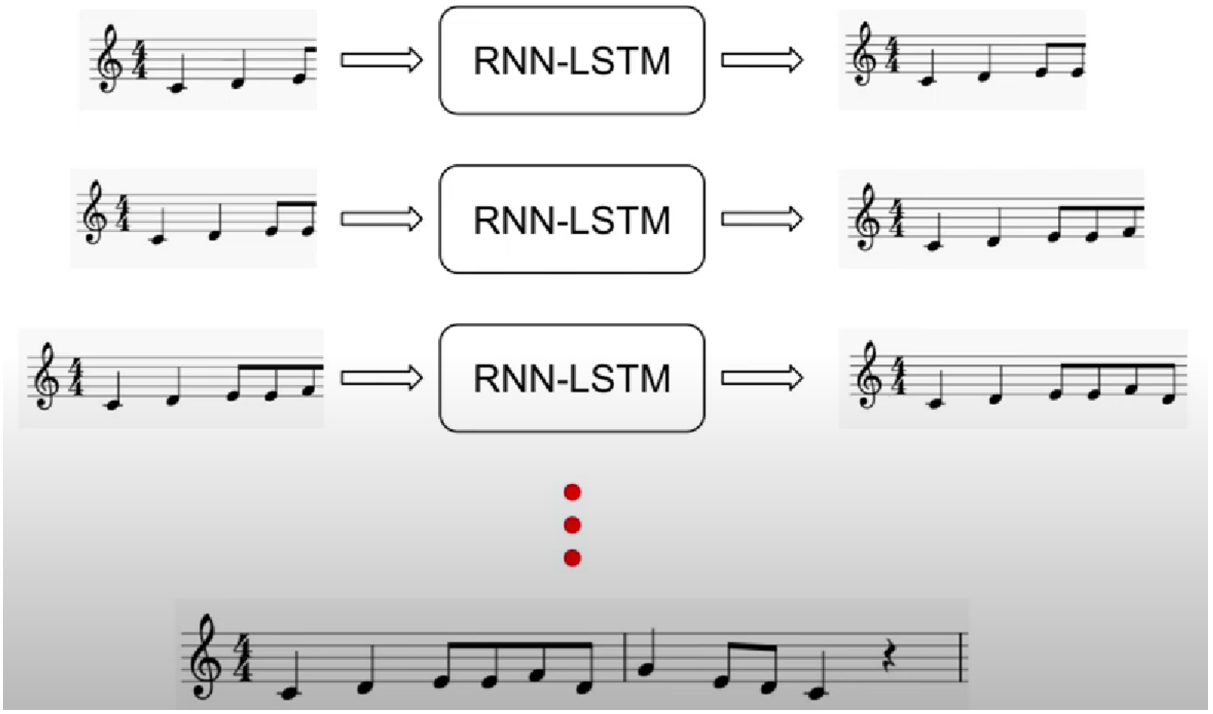
El presente informe se centrará en el entrenamiento de una Red Neuronal Recurrente (RNN) que prediga una secuencia de notas y su duración a partir de una semilla de guía, que puede variar para generar diferentes melodías. La idea principal es que la melodía resultante sea realista en cuanto a la tonalidad en la que está, los intervalos de las notas y los silencios.

Para empezar, es necesario ver el problema como una serie temporal. Para eso, puede pensarse el pentagrama musical en dos ejes:

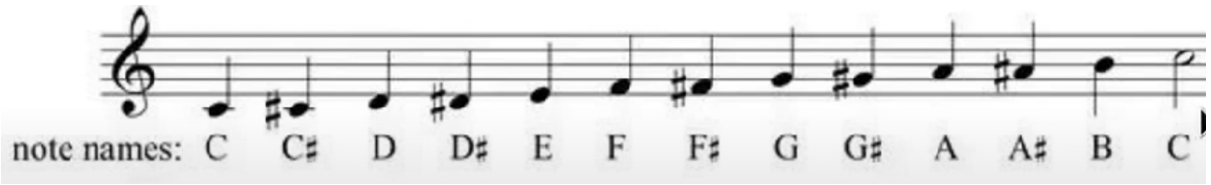
- Eje vertical: identifica a las diferentes notas musicales de acuerdo a su altura (dada por la frecuencia).
- Eje horizontal: es el tiempo que transcurre y que marca la secuencia y duración de las distintas notas musicales.

Se pretende utilizar una RNN dado que, para lograr ese realismo buscado en la predicción de cada nota, importa el orden en el que están los datos de entrenamiento y eso se puede lograr llevando el estado interno de forma que cada paso tendrá la información de los datos actuales y de los datos del paso anterior.

En el siguiente gráfico puede observarse cómo, a partir de una secuencia de tres notas musicales, se predicen las siguientes siete notas (y un silencio):

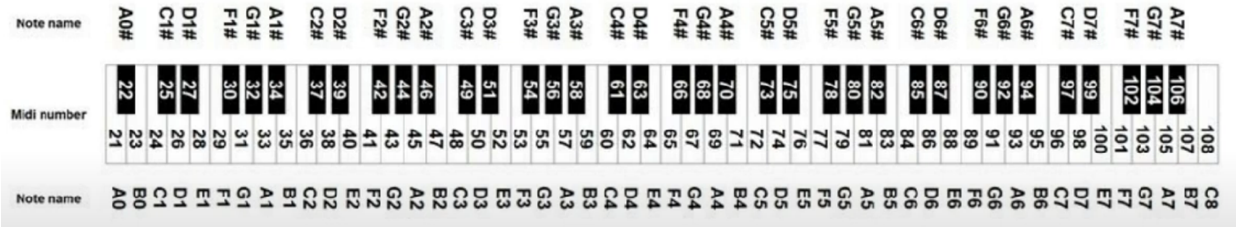


Para la preparación de los datos, es fundamental tener en cuenta algunos conceptos de teoría musical. En primer lugar, se entiende por **melodía** a una secuencia de notas y silencios. Las **notas** tienen un tono y una duración. El **tono** es la frecuencia que caracteriza a sus correspondientes ondas sonoras, y que determina cuán grave o aguda es una nota. Los tonos tienen diferentes nombres de acuerdo a la frecuencia, con la particularidad de que comienza a repetirse su nombre al multiplicar su frecuencia por un entero.

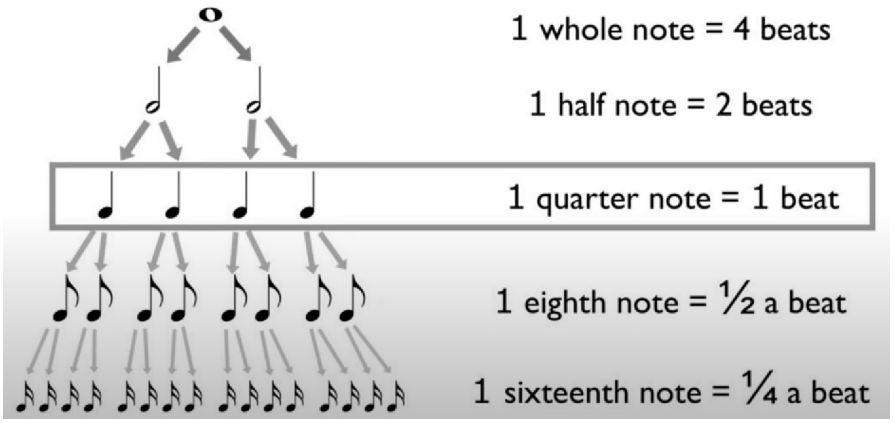


A un mismo tono pero con distinta frecuencia se lo llama octava. Por eso, para diferenciar entre octavas, se identifican los tonos con su nombre y el número correspondiente a la octava. Las teclas del medio de un piano corresponden a la octava número 4 (C4, D4, E4, …, B4).

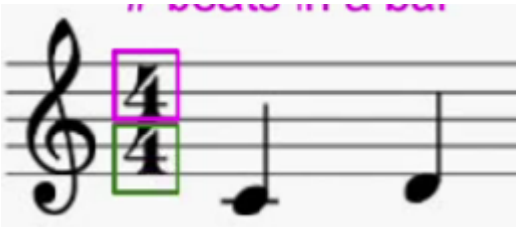
Una vez entendidos estos conceptos, es necesaria una representación para melodías que sea entendida por la computadora. Por eso surgió el formato MIDI, un protocolo en donde se asocia cada tono a un número entero, de la siguiente forma.



Luego, las duraciones de las notas también tienen su representación en el lenguaje musical, que puede observarse en el gráfico de a continuación.



En español, los nombres desde arriba hacia abajo son: redonda, blanca, negra, corchea y semicorchea. Se puede pensar que todas las duraciones son relativas a una negra, en donde una blanca dura el doble y una redonda el cuádruple, mientras que una corchea la mitad y una semicorchea un cuarto. Sin embargo, de acuerdo al pentagrama, puede variar cuál es la nota de referencia para las duraciones.



El 4 inferior indica que la nota de referencia a tomar es "1 quarter", es decir, una negra, y el 4 superior indica que por compás (una forma de separar internamente las partes de una melodía) entrarán cuatro negras.

Una **tonalidad** es un grupo de notas musicales que componen el eje de una pieza musical. Se componen de una **tónica** que se corresponde con un tono sobre el cual girará toda la pieza, y un **modo** que puede ser mayor o menor, y que tiene que ver con la forma de combinar un conjunto de tonos de forma ascendente o descendente. Como existen 12 tonos distintos y 2 modos, en total se tienen 24 tonalidades distintas. Como entrenar a una RNN para generalizar a esta cantidad de tonalidades por cada melodía sería un proceso altamente costoso, en este proyecto se decidió optar por las tonalidades de C y Am (Do mayor y La menor) dado que cuentan con el mismo conjunto de notas. Esto requiere que el conjunto de datos sea transpuesto, ya que no todas las melodías se encontraban en dichas tonalidades. Esto quiere decir que el conjunto de notas de cada melodía sea elevado o disminuido en un intervalo determinado de forma que coincida con la tonalidad buscada.

Para alimentar la RNN, se opta por una forma de representación de las melodías basada en series temporales. El enfoque se pone en los eventos de transición de una nota/silencio a otra, de forma de poder **muestrearla** con determinada precisión. Para este proyecto, se adoptó un muestreo correspondiente a una **semicorchea** (un cuarto de negra). Entonces, cada muestra de la serie se corresponde con 3 posibles tipos de valores:

- **Valor MIDI** de la nota que comienza.
- **Símbolo "_"** indicando que sigue transcurriendo la última nota/silencio.

- **Símbolo "r"** indicando el comienzo de un silencio.

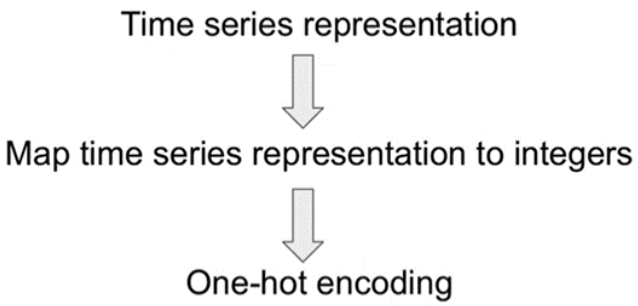
Por lo tanto, para pentagramas en 4/4, se tienen 16 muestras por compás en donde cada negra se muestrea en 4 valores (ya que una semicorchea es un cuarto de negra)

METODOLOGÍA

Para entrenar el modelo se utilizó un grupo de melodías clásicas alemanas, del sitio web *Essen Asociative Code and Folksong Database*. Este dataset provee un conjunto de canciones en formato .krn.

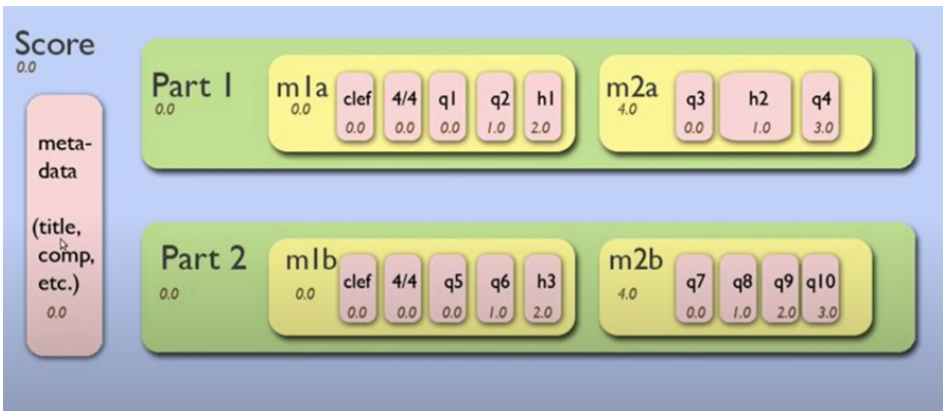
ETAPA DE PREPROCESAMIENTO DE DATOS

El manejo de diferentes formatos de datos (hasta llegar a la codificación deseada) es de suma importancia para comprender cómo se provee de datos a la red. Al tratarse de una RNN LSTM, los datos de entrada deben ser series temporales y valores enteros, por lo que el proceso de conversión de un archivo .krn a una serie temporal entera es fundamental.



a. *Carga de archivos .krn y filtrado de canciones*

Un archivo .krn, correspondiente a una canción, se puede pasar a un objeto Score (utilizando la librería music21), el cual contiene a su vez otros objetos que corresponden a las diferentes partes de la melodía.



Music21 permite ejecutar métodos sobre los objetos Score, para facilitar su manipulación y acceder a las notas que componen al objeto.

Utilizando estas funcionalidades, se filtraron las diferentes canciones de acuerdo a la duración de sus notas y se transponen a la tonalidad de Cmayor y Amenor. La idea es entrenar el modelo solo con 2 de las posibles 24 tonalidades, reduciendo la potencia de generalización, pero pudiendo utilizar menos datos para entrenarlo (en el otro caso, se debería entrenar el modelo con las canciones transpuestas a las 24 tonalidades).

b. Representación de las canciones en series temporales

Una vez filtradas las canciones, se pasan a su correspondiente representación en serie temporal. Cada elemento de la serie temporal corresponde a ¼ beat.

Utilizando nuevamente los métodos sobre los objetos Score, se diferencia si un componente es una nota o un silencio (y su duración). Cada nota queda codificada con su correspondiente valor MIDI y los silencios con "r". Las prolongaciones (tanto del silencio como de las notas) se representan con "_".

Las canciones ya codificadas se guardan cada una en un archivo, para luego unirlas en un único string (separadas por "/"), que se almacena separado.

c. Mapeo de notación de serie temporal a enteros

Como se mencionó previamente, a la red hay que proveer enteros, por lo que es necesario pasar los caracteres de la serie temporal a enteros (ya que entre ellos están, por ejemplo, "r" y "_")

Para esto, se crea un diccionario donde cada carácter de la serie temporal se corresponde a un entero y luego se almacena en el archivo mapping.json.

d. Creación de inputs y targets del modelo

Para crear los inputs y targets se toman subsets del archivo file_dataset (donde están todas las melodías juntas en formato de serie temporal) y, utilizando el diccionario creado, se convierten a enteros y se generan las

secuencias de entrenamiento del modelo. El target, para cada secuencia, es el valor que sigue justo después de cada secuencia.

Para generar las diferentes secuencias (las cuales son de tamaño fijo), se va moviendo su índice, de forma tal de abarcar otro campo y a su vez se mueve también la posición del target. Por ejemplo:

```
[11, 12, 13, 14]  ->  sec1 = [11, 12]   t1 = [13]
                      sec2 = [12,13]    t2 = [14]
```

Inputs es entonces una matriz tridimensional, donde cada dimensión corresponde a:

- cantidad de secuencias (longitud archivo file_dataset - longitud secuencia)
- longitud de cada secuencia
- cantidad de símbolos en el vocabulario (por el one-hot-encoding)

Por ejemplo, si el vocabulario tiene 18 símbolos diferentes, cada elemento de una secuencia tendrá una profundidad de 18 elementos en la matriz (a causa del one-hot-encoding)

Los targets por su parte tienen una única dimensión, un target para cada secuencia.

CONSTRUCCIÓN Y ENTRENAMIENTO DEL MODELO

Como se ha mencionado previamente la construcción del modelo sera en base a una Red Neuronal Recurrente (RNN) que son una clase de redes neuronales adecuadas para el procesamiento de secuencias, como la música.

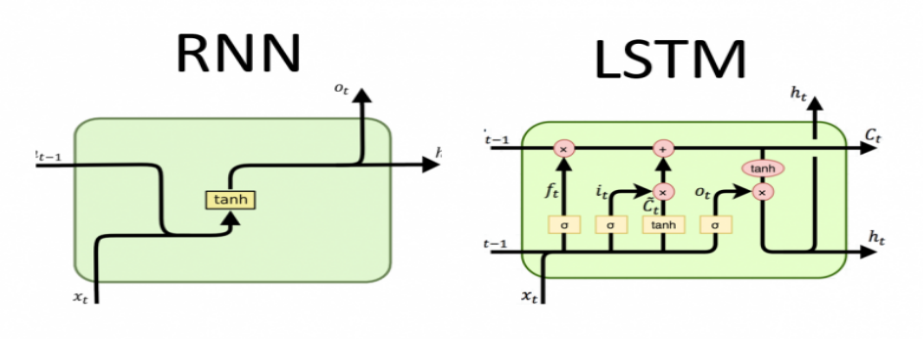
Sin embargo, las RNN puras tienen dificultades para aprender relaciones a largo plazo entre los elementos de una secuencia. Esto se debe a que la información se va perdiendo a medida que la red avanza a través de la secuencia, ocasionado debido a que el gradiente de la función de pérdida decae exponencialmente con el tiempo (llamado el problema del *gradiente que desaparece*).

Las redes neuronales de larga memoria a corto plazo (LSTM) son una variante de las RNN que están diseñadas para superar este problema. Las LSTM

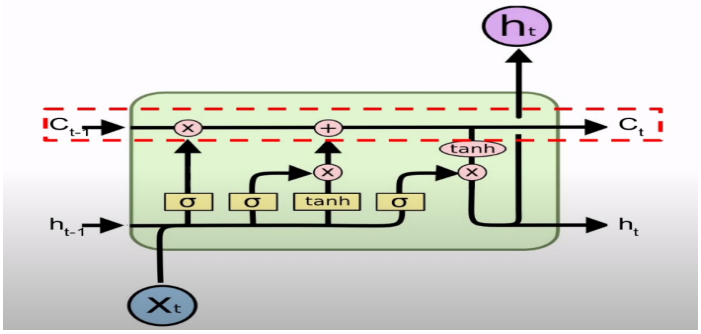
tienen una estructura interna que permite que la información se almacene en la memoria de la celda durante períodos más largos. Esto les permite aprender relaciones a largo plazo, lo que es esencial para la generación de música.

Las neuronas LSTM incluyen una "celda de memoria" que puede mantener información en la memoria durante períodos prolongados. Esta celda de memoria les permite aprender dependencias a largo plazo.

Las LSTM abordan el problema del gradiente que desaparece mediante la introducción de nuevas puertas lógicas, como las compuertas de entrada y "olvido"que funcionan como filtros. Debajo se puede ver una comparación entre una neurona de RNN y otra de LSTM.



Entonces, analizando un poco más en detalle las redes LSTM, se puede ver que contienen una celda de RNN simple, una sección de memoria a corto plazo, que se encarga básicamente de lo que está pasando en el estado actual y finalmente una "Cell State" que es la marcada abajo.



La Sección "Cell State" se encarga de guardar los patrones de largo plazo, se actualiza solo en 2 puntos en una neurona:

X Determina que olvidar.

\+ Determina qué información de la actual agregar a la cell state

Si bien esta es una explicación breve, es notorio que para el problema resuelto una celda LSTM que es más compleja provee una mayor efectividad y eficiencia para el objetivo propuesto de la generación de música ya que pueden aprender relaciones a largo plazo entre los elementos de una secuencia.

a. Construcción del modelo

Para la construcción del modelo anunciado previamente primero que nada se debe crear la red neuronal.

- **Capa de entrada:** Esta capa es la encargada de recibir los inputs preprocesados previamente.
- **Capa LSTM:** Esta capa es la capa principal de la red neuronal. Es responsable de aprender las relaciones a largo plazo entre las notas musicales. En nuestro modelo, cuenta con 256 neuronas
- **Capa de dropout:** Esta capa se utiliza para evitar el overfitting. El sobreajuste es un problema que ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y no es capaz de generalizar a nuevos datos. La capa de dropout elimina aleatoriamente algunos de los nodos de la red neuronal durante el entrenamiento.
- **Capa de salida:** Esta capa genera las notas musicales para la nueva pieza de música. Esta posee tantas neuronas como símbolos en el diccionario, que tienen que ver con las notas musicales utilizadas en el preprocesamiento. Se adiciona al final una *capa softmax*, la cual asigna una distribución de probabilidad a cada símbolo.

La función de evaluación de costos elegida fue la entropía cruzada, ya que es una red de clasificación.

En cuanto al **learning rate**, que es una constante que controla la velocidad de aprendizaje del modelo, utilizamos un valor de 0.001. Un learning rate demasiado alto puede hacer que el modelo se adapte demasiado a los datos de entrenamiento y que no sea generalizable a nuevos datos. El valor de 0.001 es un valor de learning rate relativamente bajo. Este valor es adecuado para problemas de aprendizaje automático complejos, como la generación de música.

b. Entrenamiento del modelo

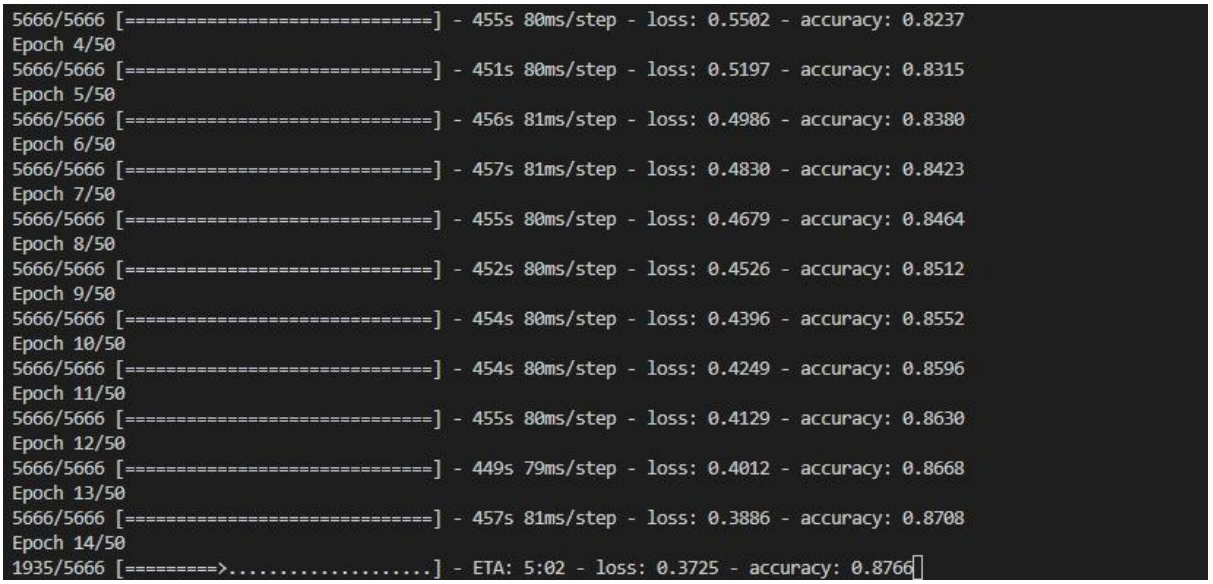
Una vez que se preprocesaron los datos, se prepararon para el entrenamiento del modelo. Para ello, se utilizaron los siguientes hiperparámetros:

- **Épocas**: 50.
- **Batch size**: 64.

El entrenamiento se realizó durante 50 épocas. Una época es un ciclo completo de entrenamiento sobre todos los datos de entrenamiento.

En cuanto al batch size este es un hiper parámetro de las redes neuronales que controla la cantidad de datos que la red neuronal procesa a la vez. Es importante notar que un batch size más grande puede ayudar al modelo a aprender mejor los datos, pero también puede requerir más memoria.

El tiempo de entrenamiento va a variar según la cantidad de épocas, el batch size, la cantidad de datos de entrenamiento y el hardware que se utiliza para el mismo. Utilizando un cpu AMD Ryzen 5 3600 de 6 núcleos, cada época tardo aproximadamente 7 minutos en entrenarse por completo. Debajo se puede ver como se muestra este proceso.



Una vez que el modelo se entrenó correctamente, se guardó para su uso posterior.

Es importante recalcar que no utilizamos datos de test debido a la naturaleza musical del proyecto. Al entrenar un modelo que genera música, consideramos primordial como forma de medir el resultado el oír la melodía generada y no las características no visibles e intrínsecas numéricas, de todas formas es algo que en un futuro podría reverse.

Generación de melodías

Una vez entrenado el modelo, se puede utilizar para generar melodías. El proceso de generación requiere de los siguientes valores:

Semilla: Se proporciona una semilla, que es una pieza de melodía que la red utilizará como punto de partida. La semilla se codifica en la representación musical basada en series temporales.
El tamaño de la semilla suele ser de entre 5 y 15 símbolos.

Número de pasos: Se especifica el número de iteraciones que se realizarán en la serie temporal que representa la música.

Longitud máxima de la secuencia: Se especifica cuántos pasos se consideran en la semilla para la red. Este parámetro ayuda a evitar que la red genere melodías demasiado largas o repetitivas.
Temperatura: La temperatura es un valor importante que afecta al estilo de la melodía generada. Un valor de temperatura alto hace que la red sea más creativa y exploratoria, mientras que un valor de temperatura bajo hace que la red sea más conservadora y siga la distribución de probabilidad de los datos de entrenamiento.

Temperatura $\rightarrow \infty$: La distribución de probabilidad se remodela y todos los valores se vuelven homogéneos. Esto significa que todos los símbolos de salida

tienen la misma probabilidad de ser elegidos. En este caso, la red *genera melodías aleatorias*.

Temperatura → 0: La distribución de probabilidad se remodela tal que el valor original que tenía mayor probabilidad se vuelve 1. Esto significa que la red siempre elige el símbolo con mayor probabilidad. En este caso, la red *genera melodías que son muy similares a las melodías de los datos de entrenamiento*.

Temperatura → 1: Se mantiene la distribución de probabilidad devuelta por la red. En este caso, la red *genera melodías que siguen la distribución de probabilidad de los datos de entrenamiento.*

El proceso de generación se basa en los siguientes pasos:

1) Inicialización: Se proporciona una semilla, que es una pieza de melodía que la red utilizará como punto de partida. La semilla se codifica en la representación musical basada en series temporales.

2) Mapeo: La semilla se mapea al vocabulario utilizado para ingresar a la red. Esto se realiza utilizando el diccionario creado en el preprocesamiento

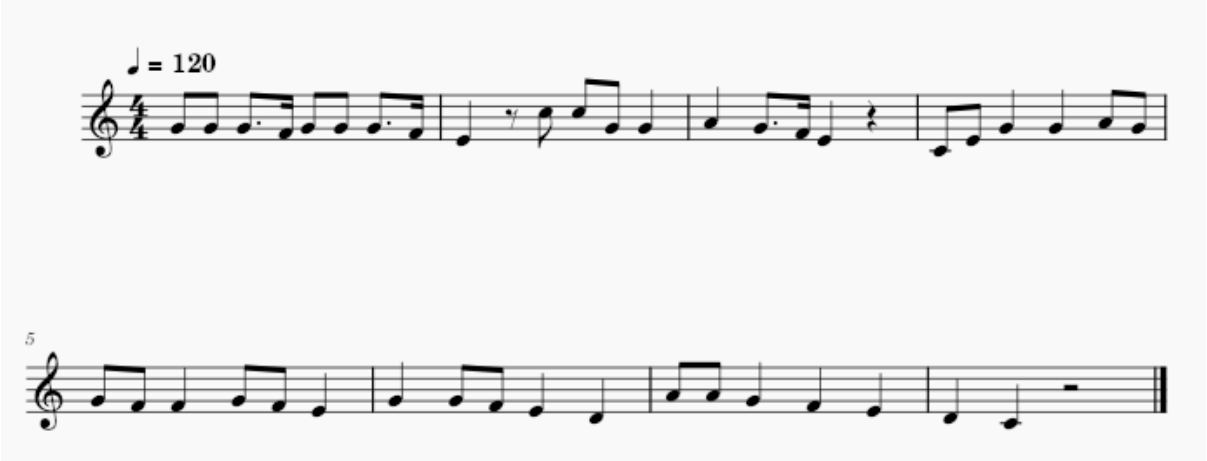
3) Codificación: La semilla se codifica en one hot encoding.

4) Predicción: El modelo predice el siguiente símbolo de la melodía. La predicción se realiza utilizando la función de salida del modelo, que es una función softmax.

5) Mapeo: El símbolo de salida se mapea a un símbolo de la melodía. Esto se realiza utilizando el diccionario nuevamente pero a la inversa.

6) Conversión a MIDI: Una vez generada la melodía, se debe convertir a un formato que se pueda reproducir en algún software y ser oída por humanos.

Tras la generación de la melodía, y la conversión a MIDI podemos ver que un software musical representará la melodía generada por la RNN-LSTM de la siguiente forma:



RESULTADOS

Los resultados de este proyecto son lo suficientemente coherentes como para dejar su interpretación a la subjetividad del oyente. Se obtuvieron distintas melodías no sólo condicionadas por el entrenamiento sino también por la semilla inicial para la generación y la llamada "temperatura".

En general, al predecir notas en las melodías se respeta la idea de "reposo" o "descanso" en la nota tónica (Do, en la mayoría de los casos) y la generación de tensión a través del 5to grado o también llamado dominante (Sol, en la mayoría de los casos) previo a resolver en la tónica. Sin embargo, no ocurre el 100% de los casos, y puede deberse a la falta de pruebas variando hiperparámetros, dado que el entrenamiento es muy costoso computacionalmente demandando intervalos de tiempo bastante grandes.

Se adjuntan externamente algunos MIDIs resultantes para ser escuchados por los lectores.

CONCLUSIONES

Para concluir, los resultados obtenidos sobrepasaron las expectativas de este proyecto, dejando las puertas abiertas a futuros desarrollos más elaborados. Por ejemplo, podría tenerse en cuenta una mayor generalización

para más tonalidades, aunque el entrenamiento sea mucho más costoso. También podrían tenerse más partes de una obra en cuenta, de forma de poder armonizar otras piezas con las piezas ya creadas, pero primero se debería analizar la potencia de una red LSTM para manejar esta propuesta, o si otra implementación ayudaría más.

La idea de que los resultados de las recursivas predicciones puedan ser analizados directamente por las personas sin la necesidad de implementar una medición de errores de entrenamiento y de test motiva bastante, ya que son más tangibles. De todas maneras, como ya se dijo, la evaluación de los resultados en cierto punto se vuelve una tarea de la subjetividad de las personas en vez de cálculos numéricos, por lo que alcanzar la perfección del modelo es prácticamente imposible.