UNIVERSITAT DE BARCELONA

Trabajo de Final de Grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

LA MÚSICA DE LOS ORDENADORES

Ignasi Rovira

Director: Eloi Puertas
Realizado en: Departamento de
Matemáticas e Informática

Barcelona, 20 de junio del 2021

Índice

1. Resumen y abstract

Resumen

Este trabajo tiene este nombre ya que lo que se pretende es componer música clásica con las herramientas que tenemos hasta ahora en los ordenadores modernos. El planteamiento inicial es: ¿Es posible componer una canción clásica con los ordenadores e inteligencia artificial moderna? Hasta ahora no se ha indagado mucho sobre la generación de obras individuales y originales de música clásica ya que las aproximaciones más buenas han sido o para casos aislados financiados por una gran multinacional como el proyecto de Huawei donde se terminó la sinfonía inacabada del compositor Austríaco Schubert. O el proyecto que más se acerca a lo que comento és DeepBach (red neuronal capaz de generar corales para cuatro voces al estilo del compositor alemán Johan Sebastian Bach). Dicho eso mi objetivo es componer un vals del estilo del compositor Polaco Chopin con tres redes neuronales. La primera la que se encarga de generar la harmonía de la canción (esqueleto de ella), después otra red que generará el acompañamiento que tocará la mano izquierda y finalmente una red neuronal que generará la melodía de la canción que tocará la mano derecha. Por la cantidad de tiempo necesario que se necesita para obtener música de buena calidad en formato electrónico he tenido que hacer el entrenamiento solo con un total de 25 valses, pero ha podido generar satisfactoriamente un acompañamiento para la mano izquierda a partir de una harmonía, pero no he podido llegar a implementar la red neuronal para la mano derecha. Si se quieren resultados fiables y decentes se necesita mucho tiempo para recopilar todas las partituras en formato electrónico (MIDI) por eso mismo si se quiere hacer un proyecto de esta envergadura dedicar todo el máximo tiempo a la recolección y procesado de las canciones para la red neuronal.

Abstract

This final degree project has this specific name because my goal is to compose classical music with modern computers. My question is: Is it possible to compose a classical song with the actual computers and artificial intelligence? Since these last few years, the research in music generation in the classical field has been very limited. For specific cases and financed by large companies like the project of Huawei where they ended the unfinished Schubert symphony, or the DeepBach project where the final goal is to generate Bach corals (simple songs played by 4 different voices). My objective is to compose a Chopin waltz with three different neural networks. The first network will generate the harmony of the song, the second neural network will generate the accompaniment for the left hand and finally the last neural network will generate the melody of the song for the right hand. For the amount of time, it takes to get good quality music in electronic format (MIDI) I only could use 25 waltzes to train the neural networks, but it could generate accurate enough an accompaniment for the left hand from a harmony, but I have no been able to get to implement the neural network for the right hand. If you want reliable and decent results you need a lot of time and effort to collect all the scores in electronic format, so if you want to do a project of this size, dedicate all the maximum time to the collection and processing of the songs for the neural network.

2. Introducción y motivación

Desde el inicio de los tiempos el ser humano ha necesitado de la música para expresar no solo sus emociones más íntimas sino poder contar o acompañar la narración de historias épicas. La música no solo transmite un mensaje, sino que desata la capacidad imaginativa del escuchante. Pero siempre ha necesitado tanto del avance de la técnica de los lutieres que hacen los instrumentos como el desarrollo de la técnica que utilizan los virtuosos al tocar ese instrumento.

Por eso mismo desde la aparición de las primeras máquinas capaces de aprender por ellas mismas dado un conjunto de datos, aparecieron los primeros soñadores planteando la gran pregunta que aún ahora no podemos responder: ¿Puede una máquina componer como un humano?

Para perseguir ese ideal tanto matemáticos, físicos y músicos se sientan codo con codo para intentar descifrar los misterios de la música.

Para hacer entender a una máquina algo tan abstracto como la música, la ciencia decidió racionalizarla, simplificando esas hondas de sonido a frecuencias fijadas para cada nota. De tal manera que, si antes gravábamos un piano como una sola onda de sonido, ahora podíamos gravar cada nota individual tocada junto a la duración y el momento que fue tocada. Esa recopilación de notas ahora será reproducida por un instrumento virtual (que en este caso no será más que todas las notas de piano gravadas en la mejor calidad posible y cada nota gravada varias veces tocada de manera diferente para mayor fidelidad en la reproducción).

Las aproximaciones más simples fueron redes generativas usadas para improvisaciones de música moderna como el Jazz (sin tener un principio y un fin definido para la pieza). En el pie de página 1, se detalla los pasos para hacer una de estas redes. En el pie de página 2 hay una canción generada por ella (muy moderna para mi gusto).
Por otra parte, en el ámbito de la música clásica, Gaëtan Hadjeres, François Pachet y Frank Nielsen escribieron el famoso paper DeepBach: a Steerable Model for Bach Chorales Generation (pie de página 3) donde se detalla la creación de una red neuronal capaz de generar corales de Johann Sebastian Bach (Compositor Alemán que compuso a finales del

1 Generación música jazz: https://towardsdatascience.com/jazz-music-generation-using-gpt-a7ed52b0f468
2 Canción jazz generada: https://soundcloud.com/trung-viet-pham/untitled01_mugen-gpt-v3
3 DeepBach paper: https://arxiv.org/abs/1612.01010

siglo 17 hasta mitades del 18, famoso por sus composiciones para piano y música religiosa para orquestra) alimentada por estas recopilaciones de música explicada anteriormente. En el pie de página 4 se pone un ejemplo de una coral generada por esta red neuronal (tocada por un órgano virtual) y el pie de página 5 es una coral compuesta realmente por Bach para poder hacer la comparativa.

En lugar de entrar voces con sus respectivas letras entran las notas con sus respectivas frecuencias. Su fantástico método de entreno para estas redes neuronales es el que me ha influenciado más en mi trabajo.

Finalmente, no puedo acabar esta introducción sin mencionar a Huawei y su acercamiento a este problema. Donde el ambicioso proyecto consistía en terminar la sinfonía número 8 de Schubert (a la cual Schubert no le pudo componer el cuarto movimiento). En el enlace de a continuación pongo el vídeo a esta la sinfonía entera donde he puesto que empiece directamente en el cuarto movimiento compuesto por la inteligencia artificial (Recomendable escuchar-la entera para contextualizar la globalidad de la pieza):

Aunque cabe decir que hay mucho trabajo humano para acabar de pulir la pieza y se pueden identificar algunas de las canciones usadas a parte de los primeros tres movimientos de la sinfonía al entrenar la red.
Después de ver todos los avances en el sector de la música i la informática pude observar que siendo una persona que hizo la carrera de piano y está acabando el grado de informática podía aportar mi granito de arena y que mejor manera de hacer-lo hay, que no sea con mi instrumento favorito: el Piano. Siendo un instrumento muy versátil ya que passar de una canción de piano a una canción de orquestra u otros instrumentos és muy fácil. Por eso decidí hacer lo que había echado en falta en los proyectos mencionados anteriormente: la capacidad de componer piezas de un grado elevado de complejidad y de manera audiblemente agradable a la persona que escucha, pero siempre manteniendo la independencia de cada pieza entre ellas. De igual manera que sean piezas con un principio y un fin. Por eso el reto que me puse fue componer un vals de Frederic Chopin: compositor

5 Canción hecha por DeepBach: https://www.youtube.com/watch?v=QiBM7-5hA6o
6 Coral de Bach: https://www.youtube.com/watch?v=Khn9jLIYE4A
7 Sinfonía Schubert: http://www.youtube.com/watch?v=_6OUGRsslJY&t=35m51s

romántico polaco con una vida muy corta pero fructífera a nivel musical (solo vivió 39 años) y és considerado uno de los compositores más importantes de la historia y uno de mis compositores favoritos capaz de componer piezas que capaces de transportar al que las escucha. Publicando en vida 10 valses y 9 más que fueron publicados después de morir. En el pie de página 8 hay uno de los valses que más me gustan tocado por uno de mis pianistas favoritos.

Con todo dicho espero que el lector pueda disfrutar tanto como yo lo hecho en mi viaje por el mundo de la música y la informática.

8 Chopin "Valse brillante" Op. 34 No. 2: https://www.youtube.com/watch?v=PGdpRmL2XUc

3. Objetivos

Para este trabajo el objetivo principal es hacer una red neuronal capaz de componer un vals de Chopin pero este objetivo tan ambicioso lo he tenido que dividir en objetivos más pequeños.

1. Ser capaz de extraer la armonía de un vals.

2. Hacer una red neuronal capaz de componer la harmonía para un vals de Chopin. La harmonía son los acordes sobre los que se hace la canción.

3. Hacer otra red neuronal que sea capaz con una harmonía como entrada hacer la música para la mano izquierda. Como mi trabajo se basa en el piano como único instrumento primero hacemos el acompañamiento de la canción.

4. Hacer una última red neuronal que sea capaz con las notas de la mano izquierda como entrada predecir las notas para la mano derecha.

4. Planificación

Este trabajo se dividirá en 7 tareas principales.

1) Recopilación de datos donde se conseguirán y corregirán los MIDI para todos los valses con los que entrenar la red neuronal. Se ha puesto un período especialmente largo ya que los MIDI los tendré que gravar o hacer yo mismo.

2) Procesamiento de los datos. Donde se harán todas las funciones necesarias para convertir esas canciones MIDI a vectores que puedan usarse para entrenar la red.

3) Hacer la red neuronal que compondrá la harmonía del nuevo vals.

4) Hacer la red neuronal que compondrá la mano izquierda del vals. Lo que tocará la mano izquierda del pianista al tocar el vals.

5) Hacer la red neuronal para componer la mano derecha del vals (melodía de la canción).

6) Pruebas y posibles correcciones a las tres redes neuronales anteriores.

7) Redacción de esta memoria.

En la tabla 1 se pone el diagrama de Gantt con la planificación para estas tareas a lo largo de la duración del semestre.

Tabla 1

Actividad / Mes	Febrero	Marzo	Abril	Mayo	Junio
Recopilación de datos					
Procesamiento de los datos					
Red neuronal harmonía					
Red neuronal mano izquierda					
Red neuronal mano derecha					
Pruebas					
Redacción de la memoria					

5. Análisis

Para poder entender el trabajo en profundidad antes hay que hacer una pequeña introducción musical que servirá para comprender todos los pasos seguidos y a aprender las palabras relacionadas con la música que se usan.

Un vals se caracteriza por ser una pieza bailable (recuerda a los bailes de salón de la clase alta del siglo 19) dividida en compases de tres tiempos cada uno. Aplicando-lo al piano la mano izquierda del pianista toca un acompañamiento y la mano derecha la melodía de la canción.

Hasta no hace mucho (finales del siglo 20) entendíamos la música como un conjunto de ondas de sonidos con frecuencias variables que se podían o escuchar o grabar. Pero una vez grabadas era muy difícil poder modificar las notas de esa canción. Para poder reproducir música sin tener que grabar-la de un músico real se inventó el formato MIDI. Ahora el ordenador asignaba un nombre de nota a cada frecuencia, una duración y el momento en el tiempo que tiene que sonar la nota. De esta manera si más de una nota tienen el mismo valor de tiempo se tocan a la vez y forman a lo que llamamos un acorde. A parte una de las grandes ventajas del formato MIDI es el poquísimo espacio comparado con las grabaciones que se habían hecho hasta ahora.

Toda canción clásica está tocada en lo que llamamos en una escala que puede ser mayor o menor. Las canciones tocadas en escalas menores suenan más tristes o melancólicas y las mayores alegres y vivas. En la imagen 1 que viene continuación dejo un ejemplo de escala de Sol Mayor que contiene las notas sol, la, si, do, re, mi, fa sostenido y sol:

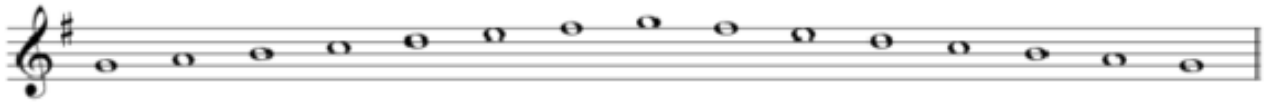

Imagen 1

Por eso mismo podemos separa los valses en dos tipos: los valses tocados en escalas mayores y los valses tocados en escalas menores.

En un vals por cada compás, formado por tres tiempos donde cada tiempo tiene una duración especificada al inicio de la pieza, generalmente se pueden agrupar las notas de este i formar un acorde que puede ser mayor o menor como las escalas mencionadas

anteriormente. Por ejemplo, el acorde de Do Mayor tendría las notas Do, Mi y Sol que pueden estar o no repetidas (para simplificar un poco la explicación he supuesto que en cada compás siempre solo se toca un acorde). Cogiendo cada acorde formado por cada compás se genera lo que llamamos la harmonía de la canción o el esqueleto de esta. Para entender-lo de manera más simple la harmonía son los acordes que tocaría un guitarrista al acompañar la melodía de la canción.

Para poder componer una canción clásica hay unas reglas que tenemos de tener en cuenta para que la canción sea audiblemente agradable para el que la escucha. Primero se escoge en que escala se quiere componer el vals y si se quiere hacer un vals más triste o melancólico y hacer esta escala menor o si por el contrario se quiere un vals alegre, dinámico y vivo hacer la escala mayor. Al haber escogido la escala la harmonía que haremos para esta canción tendrá que seguir unas secuencias de acordes concretas acordes a la tonalidad escogida (mayor o menor). Por eso mismo si entendemos la canción como una secuencia de acordes con patrones matemáticamente lógicos (limitados por las propias reglas de la música) podemos hacer una red capaz de captar estos patrones y replicarlos para poder generar nuevas secuencias. Meter en la red neuronal estas secuencias (representadas por vectores) hay que leer la canción en formato MIDI y procesar-la. Para esto usaremos una librería que te da muchísimas herramientas para trabajar con canciones en este formato que se llama music21.

La manera que tenemos los músicos de leer o modificar las canciones son las partituras. La herramienta que usaré para poder modificar partituras o poder visualizar lo que music21 ha leído o lo que la red ha generado usaré MuseScore que si se instala permite ver la partitura directamente en el júpiter notebook como se ve en el código entregado del trabajo.

Ahora hablemos de una de las problemáticas más grandes que encontramos en el momento de querer meter una canción en una red neuronal. Para poder hacer-lo tenemos que convertir las notas o acordes tocados en vectores. Donde a cada vector le asignaré una duración fija. Por ejemplo, si asigno a los vectores una duración de medio tiempo una nota que dure 1 tiempo y medio ahora corresponderá a 3 vectores iguales pero una nota que dure un cuarto de tiempo se ampliará su duración a medio tiempo. Pero el problema viene con las notas con duraciones fraccionales, por ejemplo, una nota que dura 1/3 de un tiempo o 1/5 o 1/7 que no es múltiplo del valor asignado al vector. Para simplificar estos casos se hará que por ejemplo 5 notas seguidas que duran 1/5 y que se tocan seguidas, pasaran a agruparse en dos vectores donde en uno se representarán 3 notas i en el otro 2.

Los vectores tendrán 88 posiciones (correspondientes a las 88 teclas del piano) donde si la tecla es tocada habrá un 1 y un 0 en caso contrario.

Casi todos los valses de Chopin están tocados en tonos diferentes (escalas diferentes) y no solo eso sino que Chopin los compuso para ser tocados en ese tono (para que sea cómodo para el pianista), pero el problema con las redes neuronales es que si tenemos una canción es Sol Mayor (esta tocado en la escala que contiene las notas sol, la, si, do, re, mi, fa sostenido y sol) y ahora esa misma canción la bajamos a Do Mayor (tocado en la escala con las notas do, re, mi, fa, sol, la, si y do) para una red neuronal que recibe estos vectores descritos anteriormente es una canción completamente distinta. Por eso para solucionar este problema primero se tendrá que pasar todas las canciones a la escala de do Mayor si son Mayores o a la escala de Do menor si son menores aun que audiblemente hablando puede quedar demasiado agudo o grave.

Para usar vectores en Python usaré la librería Numpy, librería perfecta para trabajar con grandes cantidades de datos de manera mucho más eficiente (al estar implementada en C++) y haré las redes neuronales con Keras (Tensorflow). Librería hecha por google que da una solución potente, simple, eficiente y fácil de trabajar con redes neuronales.

Si se quiere indagar mas en profundidad en alguna de las herramientas mencionadas en este trabajo dejo los vínculos en los anexos al final de la memoria.

6. Recopilación y procesamiento de los datos

Este es el apartado más grande, crítico y largo de todo el trabajo ya que sin buenas canciones en formato MIDI y un buen procesamiento de esos datos para convertirlos en vectores no se podrá obtener ningún resultado decente en el trabajo.

6.1. Recopilación de los datos

Al indagar por internet me encontré con la desagradable sorpresa que no había ninguna manera de conseguir los valses de Chopin en formato MIDI lo mas fieles posible a la partitura, no solo eso, sino que los MIDIs encontrados eran de calidades deplorables. Y me propuse dos alternativas. La primera era coger todas las partituras en formato papel y introducirlas a mano en MuseScore y de allí extraer el MIDI. La segunda era gravar con el piano todos los valses usando MuseScore y después exportarlos en formato MIDI. Ya que introducir cada vals a mano era muy tedioso acabé haciendo un poco de las dos opciones (algunos valses los grabé yo mismo y otros los pasé de la partitura en formato PDF a MuseSore a mano) pero de esta manera me aseguraba que la fidelidad a la partitura original era perfecta y la calidad del MIDI era buena.

Cuando finalmente pude conseguir todos los valses en formato MIDI empezó la segunda fase. Como cada vals estaba escrito en una escala diferente usé MuseScore para transponer todos los valses a la escala de Do (mayor o menor dependiendo del vals).

Finalmente, lo último que faltaba corregir eran las notas con duraciones irregulares (que no era múltiplo de la duración que había escogido para mis vectores). Para esto usé GarageBand para forzar a la canción MIDI a que el tiempo más pequeño tocado en la canción sea igual al que había escogido para mis vectores.

Ya que la gran mayoría de notas tocadas en la mano izquierda no eran más pequeñas de duración que medio tiempo escogí ese valor para la duración de estos vectores. Compases de 3 tiempos: 6 vectores por compás.

Por el contrario, para la mano derecha escogí una duración de un cuarto de tiempo, compases de tres tiempos: 12 vectores por compás.

Finalmente separé las notas de la mano derecha en un MIDI con la duración mínima de la nota a un cuarto de tiempo y las de la mano izquierda en un MIDI con la duración mínima de medio tiempo y los guardé en dos carpetas diferentes una llamada MD para la mano derecha y otra llamada MI para la mano izquierda.

Debido a la cantidad de valses 19 valses y aparte añadí 8 valses más de otros compositores parecidos a los valses de Chopin para tener más datos.

Debido a la envergadura del proceso este apartado ha sido el que ha cogido más tiempo con diferencia.

6.2. Procesamiento de los datos

Para poder convertir el fichero MIDI que he generado antes al formato de vectores que he explicado anteriormente (vectores de 88 posiciones donde cada posición es una tecla del piano).

6.2.1. Harmonía de la canción

Los primeros pasos son solo para los MIDI de la mano izquierda que he extraído antes ya que de esos MIDI se extrae la harmonía de la canción:

1. Leo el fichero MIDI creando una lista donde se van añadiendo todos los acordes, notas o silencios en el orden que tienen que ser tocados. Cada elemento de la lista si es una nota o un acorde tendrá primero una lista con un string con el nombre de la nota (carácter americano: la = A, si = B, do = C, re = D, mi = E, fa = F, sol = G y si es sostenido se representa como ‘#’ por ejemplo G# y cuando son bemoles se representa como ‘-‘ por ejemplo B-) y la octava que ha sido tocada del piano (el piano tiene 8 octavas más tres notas en la parte izquierda que forman parte de la octava 0, números del 0 hasta el 8). Por ejemplo, G#3. Y el tiempo dentro del compás en el que se tiene que tocar que va del 1 al 3,5 (compases de 3 tiempos, pero cada compás se subdivide en dos que es el valor mas pequeño para los vectores). Después de esta lista pongo el objeto de music21 Note() o Chord() original del que he sacado los datos. Ejemplo para un acorde de Sol mayor tocado en el tiempo 2 del compás: [[‘G3 B3 E4’, 2], <music21.chord.Chord G3 B3 E4>]. En el caso de los silencios la lista

contendrá ‘rest’, el tiempo en del compás que se toca y la duración del silencio y el objecto Rest() de music21. Por ejemplo un silencio tocado en el tiempo 2 y que dura 5 tiempos: [[‘rest’, 2, 5.0], <music21.chord.Rest>].

2. Music21 al leer el MIDI separa, sin ninguna lógica musicalmente hablando, los silencios en silencios más pequeños, para evitar problemas junto todos los silencios que van seguidos en un silencio más grande que tendrá una duración que será la suma de la todas las duraciones.

3. Ahora lo que hago es agrupar en listas por compás, en el caso de que haya un silencio que dure más que los tiempos que le faltan al compás lo parto en silencios más pequeños de un tiempo cada uno. Por ejemplo, si tengo un silencio que dura 7 tiempos y que se toca en el tiempo 2 del compás llenaré el tiempo dos y tres del compás con silencios de un tiempo y empezaré a llenar los siguientes compases hasta llegar a la duración del silencio. De esta manera tendré una lista de n compases donde n corresponderá al número de compases totales en la partitura original de la canción y dentro de cada compás las notas y silencios que se tocan.

4. Ahora todo lo que hay en cada compás lo juntamos en un mismo acorde. Con esto tendremos la secuencia de acordes que forman la harmonía de la canción.

5. Ahora convierto esta lista de acordes a una lista numpy.array de vectores de 88 posiciones.

6. Finalmente agrupo los vectores en ventanas de 5 vectores cada ventana teniendo en cuenta que un acorde puede llegar a estar en 5 ventanas. Para que la red neuronal aprenda donde hay un principio de canción y donde hay un final añado 4 silencios al principio de todo y 5 silencios al final.

Con estos pasos tendré los acordes procesados de la harmonía de la canción preparados para poder-se meter en la red neuronal.

6.2.2. Mano izquierda

Ahora los pasos para procesar los datos de la mano izquierda:

Los tres primeros pasos son exactamente iguales que los explicados en el apartado anterior para procesar los acordes de la harmonía. Así que empezaré a explicar por el paso que viene a continuación:

1. Después de tener la lista con listas correspondientes a cada compás, elimino los silencios erróneos, ya que hay casos en los que una nota que debería durar un tiempo entero music21 lo lee como una nota que vale medio tiempo y un silencio de medio tiempo. Ese tipo de silencios son los que quiero limpiar.

2. Ahora por cada compás en la lista divido las notas, acordes o tiempos que duran más de medio tiempo en partes iguales de medio tiempo cada una. De esta manera quedan n compases con 6 notas, acordes o silencios dentro de ellos.

3. Y ahora ya puedo convertir esta lista de compases en una lista de vectores de 88 posiciones cada uno.

4. Por último, agrupo los vectores en ventanas de 12 elementos, donde un vector puede estar en un máximo de 12 ventanas. Esta será la mupy.array que recibirá la red neuronal.

6.2.3. Mano derecha

Muy a mi pesar este trabajo ha acabado teniendo una envergadura más grande de la que estimábamos. Por eso mismo aun que me hubiera encantado haber podido llegar a hacer la parte de la mano derecha también, no ha sido posible. Pero explicaré igualmente los pasos que tenia previsto seguir para esta mano:

Los tres primeros pasos serán exactamente igual que en el punto 6.2.1. Los pasos siguientes serán como los del apartado de antes (de la mano izquierda) pero con alguna variación, ya que ahora los compases no tendrán un máximo de 6 notas, acordes o silencios,

sino que serán 12, valiendo un cuarto de tiempo cada uno. Además, ahora ya no serán ventanas de 12 elementos, sino que serán ventanas de 24.

7. Implementación de las redes neuronales

Para poder componer el vals entero se necesitarán un total de tres redes neuronales. La primera será la que generará harmonías para una nueva canción. La segunda generará la mano izquierda y por última la tercera generará la mano derecha.

7.1. Red neuronal para la harmonía

Esta será la red encargada de generar la harmonía de la canción.

Para poder hacer esto se montará la red de esta manera:

Los inputs para esta red serán grupos de 5 vectores de 88 elementos. Para hacer esto hay que coger todas las notas, silencios y acordes ya procesados y añadir 4 vectores vacíos delante y cinco detrás ahora como si deslizáramos una ventana de 5 elementos por toda la lista ir cogiendo los elementos de 5 en 5, de tal manera que un elemento puede estar en un máximo de 5 ventanas a la vez.

Los outputs de esta red serán por cada ventana de 5 elementos el vector que vendría justo después. De esa manera predecimos el elemento siguiente a los 5 que ya tenemos en la ventana.

Esta red será estará compuesta por capas LSTM (Long Short-Term Memory en inglés) seguidas por capas densas (capas de neuronas conectadas completamente con la capa anterior). Las capas LSTM son tipos de capa muy buenas en recordar patrones o secuencias en series temporales. Ya que estamos tratando a la canción como una serie temporal que sigue unos patrones y frecuencias concretas este tipo de red neuronal cuadra bastante bien con nuestras necesidades. También pondré un Dropout del 20% justo después de pasar por las capas LSTM. El dropout lo que hace es apagar aleatoria las neuronas para que el entrenamiento sea más robusto y no haya overfit (sobre entrenamiento de la red, la red deja de funcionar bien con datos ajenos a los usados en el entrenamiento de esta). La función de activación de la ultima capa densa de 88 neuronas (para tener una salida de un vector de 88) será la sigmoide ya que mis datos dentro del vector solo pueden ser 0 o 1 (son binarios) y justamente esta función funciona muy bien para valores de 0 a 1. Por último, la función de perdida que usaré será la binary crossentropy, (entropía binaria cruzada logarítmica) és

justamente una función de pérdida que funciona muy bien en problemas de clasificación binaria y con un optimizador Adam.

Para entrenarla lo hago durante 100 epochs y como no hay demasiados datos va rápido en entrenar.

Para generar una nueva harmonía el primer input para poder predecir tiene que ser 4 silencios (vectores con todo 0s) i un primer acorde que hay que poner a mano. En el código empiezo con un acorde de Do mayor y genero un total de 63 acordes a partir de este.

7.2. Red neuronal para la mano izquierda

Para entender la arquitectura de esta red neuronal podéis ir a mirar la imagen 2 que he puesto justo después de esto:

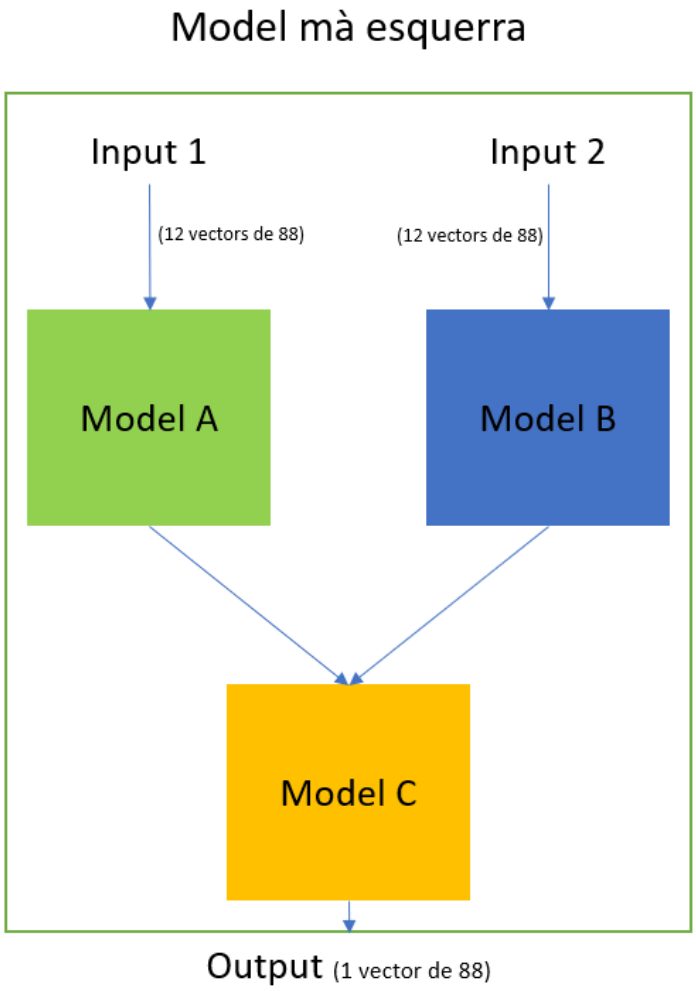

Imagen 2

Esta red neuronal se compone de 3 submodelos más pequeños a parte que en lugar de un input como antes, ahora tiene 2.

El funcionamiento será el siguiente:

Ahora entrarán tanto las 12 notas antes que la que queremos predecir como las 12 de después. De esta manera la red neuronal tiene un contexto más amplio para poder predecir la nueva nota.

De esta manera ahora haremos una lista con bloques de 12 vectores donde al principio i al final de la canción se incluyen 11 i 12 vectores en blanco para mostrar tanto el inicio como el final de la canción a la red. Cada vector puede estar en un máximo de 12 ventanas. El input uno irá des de el elemento más lejano del que queremos repetir entre los elementos anteriores a el más cercano. El input 2 irá también des de el elemento más lejano del que queremos repetir en este caso entre los elementos futuros a el más cercano.

Para poder generar estos inputs tendré que coger los acordes que había encontrado antes en el apartado 6.2.1, donde cada acorde corresponde a un compás de la canción y los repetiré 6 veces cada uno. De esta manera si tengo una canción de un total de 100 pentagramas y he generado una harmonía de 100 acordes ahora tendré una lista de 600 acordes que son la cantidad máxima de notas que puede haver en la canción ya que la medida de tiempo más pequeña es ½ de tiempo y hay 3 tiempos en cada compás.

Como se puede observar al obtener las notas de la mano izquierda obtenemos un total también de 600 notas en este caso. Equivaliendo cada nota a un acorde. Por eso ahora por cada 24 acordes, por ejemplo, de entrada, serían los acordes del 0 al 11 i los acordes del 24 al 13. Y de salida la nota número 12 de la mano izquierda.

Esta red neuronal ahora será bastante más grande que la anterior, con el doble de capas LSTM en cada uno de los modelos A y B que en la red neuronal de la harmonía después una capa que concatena la salida de las LSTM del modelo A con la salida de las LSTM del modelo B, poniendo las del A antes que los del B. Finalmente el Modelo C conta de 2 capas densas teniendo la última una activación sigmoide como antes y 88 neuronas. Igual que antes la función de pérdida será la categorical_crossentropy y un optimizador Adams.

Por desgracia no he podido llegar a hacer la red neuronal para la mano derecha pero la arquitectura sería la misma lo único que cambiaría sería la cantidad de capas en cada modelo y la entrada en lugar de 12 vectores por cada input, serán 24 en cada entrada de la red neuronal.

8. Pruebas, evaluación y resultados

Al ser un trabajo meramente musical, también las pruebas se han enfocado hacía la crítica musical y la evaluación de los resultados en base a mis conocimientos musicales.

8.1. Extracción de los acordes de la harmonía

Los primeros pasos fueron probar la eficiencia con la que mi programa extraía los acordes de una canción dada por parámetro. Para esto comparaba los acordes generados por mi programa con los acordes que sabía que tenían que salir. Los resultados fueron sorprendentes, ya que con bastante facilidad es capaz de extraer con eficiencia los acordes de cada compás. Pero solo funciona en las canciones donde tienen la mayor parte del tiempo el acompañamiento en la mano izquierda (como son los valses) pero si se quisiera generar una canción con la melodía en la mano izquierda se tendría que hacer un enfoque diferente. En la carpeta del código hay un fichero MIDI llamado ‘harmonia.midi’ donde se puede escuchar la precisión con la que extrae los acordes de uno de los valses de Chopin.

8.2. Red para componer la harmonía

Para poder hacer las pruebas de generar una harmonía se ha hecho lo siguiente:

Primero hay que escoger un acorde con el que empezar a componer, a partir de allí llenar una ventana con 4 silencios (vectores con todo cero) y el quinto elemento será el acorde inicial. Se predice el segundo acorde y se vuelve a llenar una ventana ahora con tres silencios y los dos acordes (el inicial y el predicho) en las posiciones 4 i 5 de la ventana. De esta manera si se va haciendo n veces se puede generar un total de n+1 acordes para la harmonía (n acordes predichos más el acorde con el que se decidió empezar).

Finalmente, solo hay que convertir estos vectores a acordes o notas de music21 y generar el fichero MIDI. En la carpeta ‘codi.zip’ hay un fichero llamado ‘valse_harmony.mid’ como ejemplo para ver una secuencia de acordes generados por la red.

A nivel harmónico (las notas que hay en cada acorde y la secuencia de acordes que hace) es sorprendente lo que hace la red. Teniendo en cuenta que un total de 25 valses és muy poco para tener resultados prometedores. Pero en lo que no acaba de funcionar bien es en lo grave o agudo que pone las notas ya que a veces hay saltos super grandes que nunca se

pondrían en una canción para piano. Los dos problemas que tiene esta red generativa son dos:

1) A veces se queda en un bucle de acordes infinitamente (pasa cuando son acordes muy simples y la red siempre devuelve lo mismo en bucle). Para solventar esto habría estado bien disponer de mas datos y aumentar el tamaño de la ventana de 5 a 7 como mínimo.

2) Cuesta mucho llegar a un final concluyente. Ya que tienes que parar de generar tu mismo en el momento que crees que los acordes generados pueden funcionar como final. Para solventar este problema se tendría que haber hecho una red más que solo compusiera finales a partir de una entrada de 10 acordes (por ejemplo). Entonces generar n acordes i en el momento que quieras un final llamar a la red que compone finales.

8.3. Red para componer la mano izquierda

Ya que los resultados generados por la red de la harmonía no eran lo suficiente consistente como para generar una mano izquierda con ellos para probar esta red correctamente lo que he hecho es coger una Mazurca de Chopin (bailes tradicionales poloneses que usó Chopin para componer sus 57 mazurcas) de la que he extraído con mi programa la harmonía y la he usado como entrada para testear la red. La mazurca usada se puede escuchar en el pie de página número 9 tocada por uno de los mejores pianistas de todos los tiempos.

Ahora la red tendrá de entrada todas las ventanas de 12 elementos de antes y los 12 de después de cada elemento a predecir, que serán los acordes, repetidos 6 veces cada uno una vez por el menor tiempo que queremos tener en la mano izquierda (que es ½ de tiempo y 3 tiempos por compás), de la harmonía de la canción extraída de la mazurca. Con esto generamos 666 vectores (La mazurca tiene 111 compases y tres tiempos en cada compás).

Y agrupamos los vectores de dos en dos para formar notas o acordes. De esta manera acabo teniendo 333 notas (3 notas por cada compás). Un ejemplo de una mano izquierda generada por esta red está en la carpeta del código y se llama ‘mazurca_ma_esquerra.mid’.

De los resultados de esta red hay que comentar bastantes cosas:

9 Mazurca opus 67 número 4: https://www.youtube.com/watch?v=AcxZRI6aews

Muy sorprendentemente para mi la red es capaz de hacer acordes i notas tocables para el pianista. Nunca habrá un acorde con dos notas tan lejos que sería imposible de tocar.

El programa és capaz de darle una estructura de vals al acompañamiento que estoy generando para la mano izquierda. Encontrando una nota tocada sola (la nota fuerte o bajo del vals) y dos acordes tocados seguidamente en cada compás, aunque a veces hay excepciones. No solo esto, sino que es capaz de incluir notas que no forman parte del acorde, pero quedan bonitas (notas de paso).

La red es capaz de detectar las repeticiones de esta manera tengo partes del acompañamiento comunas en la pieza. Eso da un sentido más musical y natural a la pieza. Ya que casi siempre en un vals se repiten partes de la obra para enfatizar un tema en concreto o finalizar la pieza recordando el principio de esta.

La pena es que la red al haber estado entrenada con pocos valses aun no es capaz de escoger lo agudo y grave que tienen que sonar algunos acordes o notas. También se puede apreciar algún silencio poco natural.

Lo mas sorprendente de todo es que la red es capaz de hacer cadencias con bastante eficiencia. Con cadencias me refiero a esos momentos en la música donde se tocan una serie de acordes que avisa al que escucha que o se llega al final o se pasa a un tema diferente.

Para poder escuchar la red sin preocuparse por el problema de las notas agudas o graves he creado un programa que lo que hace es centrar todas las notas y acordes en el centro del piano. De esta manera acabo teniendo una mano izquierda como la que está en la carpeta del código y se llama ‘mazurca_ma_esquerra_centraliced.mid’.

Finalmente, con MuseScore edito un poco esta salida para que las notas que tienen que funcionar como bajo del vals sean más graves que los acordes tocados a continuación y acabo teniendo un resultado como el que se puede apreciar en el fichero ‘mazurca_ma_esquerra_vals.mid’ que recuerda completamente al acompañamiento de un vals.

9 Mazurca opus 67 número 4: https://www.youtube.com/watch?v=AcxZRI6aews

9. Conclusiones y trabajo futuro

Al ser un trabajo de una envergadura tan grande la cantidad de valses que he podido usar han sido pocos y me he acabado quedando sin tiempo para hacer la parte de la mano derecha. Pero lo que está claro es que da de si para hacer un máster y un doctorado.

Pero al final las conclusiones que saco son varias.

Se puede componer para piano con una fidelidad decente si se usan redes no solo generativas, sino que tiene en cuenta el contexto tanto por delante o por detrás de lo que se quiere predecir. Pero tu capacidad de generar contenido bonito, decente y original dependerá directamente del tiempo que tengas para no solo recolectar canciones sino tenerlas en la mayor calidad posible. Quanta más fidelidad se quiera más tiempo se necesitará para la recolección de datos hasta tal punto que para tener resultados comercializables recomendaría mínimo 1 año o dos de recoger piezas en formato MIDI y después hacer redes neuronales por tipología (ya que si se hace por tipología y autor limita mucho la cantidad de piezas que se pueden conseguir). Por tipología me refiero: un conjunto de redes que componga un vals, otro diferente que componga un nocturno, otro que componga una sonata…

Para generar los acordes de la harmonía (que son los generadores infinitos que se usan hasta ahora) los resultados con capas LSTM dejan que desear. Llega a sacar una harmonía, pero no lo suficientemente consistente como para usarlo en una canción. No hace falta decir que ya solo este apartado podría ser un trabajo de final de máster. Por falta de tiempo no he podido probarlo (ya que tenía que canviar muchas partes del código) pero las redes que mejor funcionan y que más se están usando en este tipo de problemas son los Transformers. Ahora ya no tendrás como input un vector, sino que serán letras codificadas en ANSI. El problema de estas redes és que te limitas a los acordes que has definido, pero hay muchos casos (la mayoría de los casos) donde la mayoría de las notas que se tocan pueden no formar parte del acorde por ende tendrías una secuencia de acordes demasiado generalista necesitando un paso más para generar acordes usables en la composición de la mano izquierda, por eso mi intento en el enfoque con las LSTM. Aunque se tendría que observar los resultados de esta red si dispusiéramos de un banco de canciones en formato MIDI decente grande.

Viéndolo con perspectiva y con el tiempo o la gente necesaria para hacerlo sería muy interesante una red para componer sonatas para piano (por la gran cantidad de sonatas de diferentes autores de las que disponemos) pero teniendo en cuenta que hay sonatas que pueden durar 30 o 40 minutos el tiempo necesario para pasar todo a MIDI puede incrementarse drásticamente.

Como es lógico estos mismos conceptos son aplicables tanto a la música clásica como a la moderna pudiendo hacer un proyecto en el que se compone una canción pop o rock.

Viendo los resultados veo muy factible usar este tipo de redes como la de la mano izquierda como base para que un compositor pueda componer, y no solo eso, sino que también poder dar la opción de regenerar partes de la canción dependientes de lo que ya ha escrito el compositor. Con esto si un compositor sabe lo que quiere hacer y de que género hacerlo el ordenador puede ofrecer una plantilla editable.

Para mi ha sido un trabajo muy enriquecedor en el que he podido no solo disfrutar de la informática sino poderlo mezclar con mi otra pasión que es el piano. Aunque este trabajo no supone un final, sino que, todo lo contrario, este trabajo és el principio del camino de la música clásica ya que no solo se puede mejorar lo ya hecho y acabar con la red para la mano derecha sino que se puede focalizarse en muchos temas que se abren que antes parecían sin solución.

10. Referencias

MuseScore: https://musescore.org/es/handbook

DeepBach git repository: https://github.com/Ghadjeres/DeepBach

Proyecto DeepBach: https://www.flow-machines.com/history/projects/deepbach-polyphonic-music-generation-bach-chorales/

Paper de DeepBach: https://arxiv.org/abs/1612.01010

Music21 documentation: https://web.mit.edu/music21/doc/

Numpy documentation: https://numpy.org/doc/

Keras documentation: https://keras.io/api/