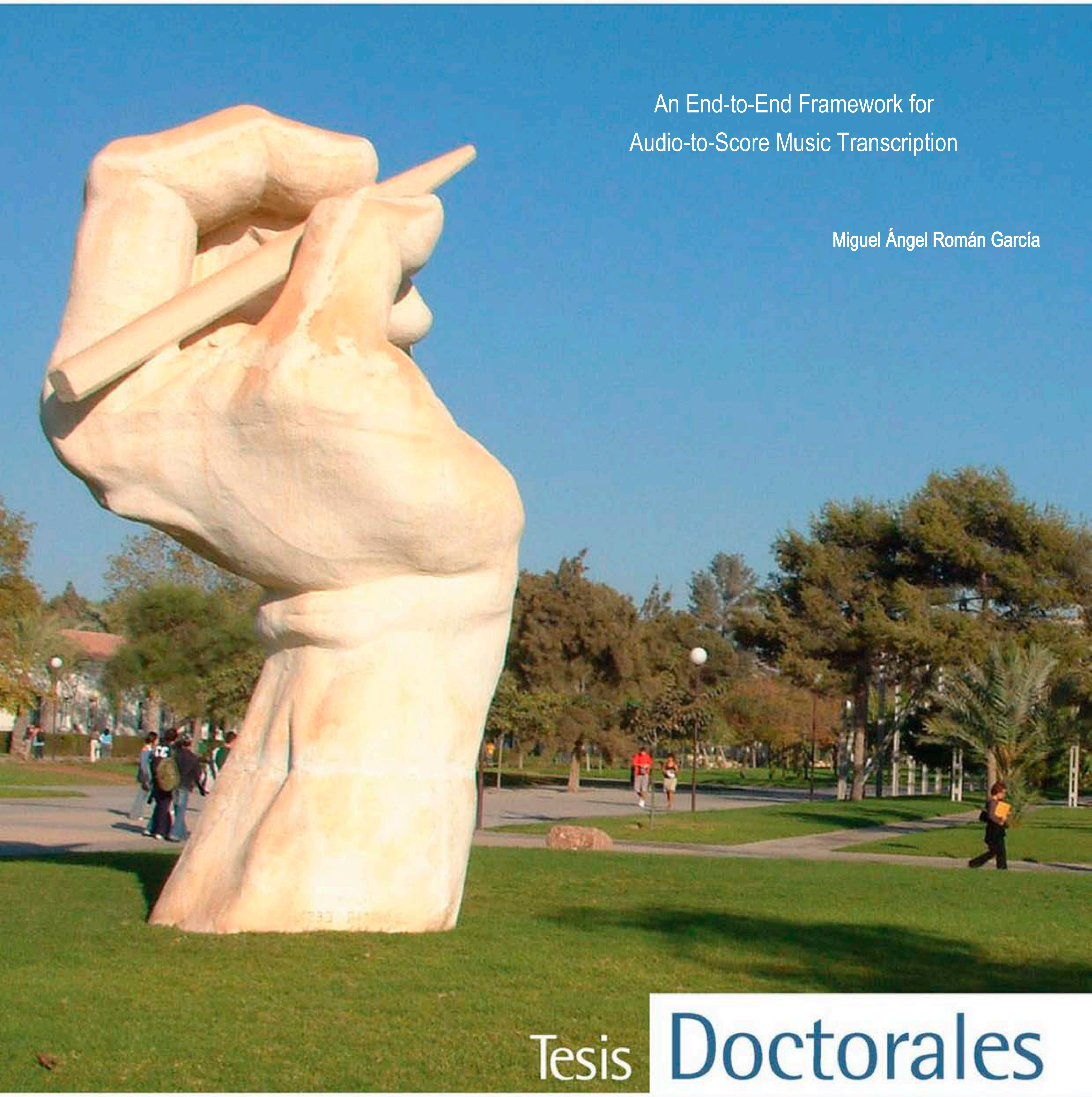


An End-to-End Framework for Audio-to-Score Music Transcription

Miguel Ángel Román García

Departamento de Lenguajes y Sistemas Informáticos
Escuela Politécnica Superior

An End-to-End Framework for Audio-to-Score Music Transcription

Miguel Ángel Román García

Tesis presentada para aspirar al grado de

DOCTOR POR LA UNIVERSIDAD DE ALICANTE

DOCTORADO EN INFORMATICA

Dirigida por

Dr. Antonio Pertusa Ibañez

Dr. Jorge Calvo Zaragoza

Esta tesis ha contado con el soporte del Ministerio de Ciencia e Innovación a través del proyecto HISPAMUS (No. TIN2017-86576-R) apoyado por fondos FEDER de la Unión Europea.

A Emilia, Gabriel y César.

Agradecimientos

En primer lugar quiero agradecer a mis dos directores de tesis, Antonio Pertusa y Jorge Calvo, por empujarme a hacer este trabajo, y por la paciencia que han mostrado para enseñarme a hacer ciencia, con rigurosidad, sin atajos y con el objetivo de aportar conocimientos a la comunidad científica. Llevando toda mi vida profesional en el mundo de la empresa, y teniendo un perfil más de ingeniero que de científico, realizar los estudios de doctorado ha sido una experiencia muy edificante, que me ha convertido en un mejor profesional, y es gracias a ellos.

También quería agradecer a la comunidad de ISMIR, por organizar unas conferencias anuales de primer nivel y darme la oportunidad de presentar mis trabajos ante una audiencia de científicos tan talentosos, que además son excelentes músicos. He aprendido mucho de sus publicaciones científicas, pero conocerlos en persona también me ha enseñado a que, con esfuerzo y dedicación, se puede vivir de aquello que nos apasiona.

A mis padres, Pascual y Magdalena, a los que debo la vida que me han regalado, a mis hermanos Juan Antonio y Pascual, y a mi cuñada Inma, les agradezco su apoyo incondicional en este y otros proyectos que he emprendido, mostrando una confianza ciega en mí y dándome la fuerza necesaria para llevarlos a cabo.

Y por último quería expresar de manera especial mi eterno e infinito agradecimiento a mi mujer Emilia y a nuestros dos maravillosos hijos Gabriel y César, a quienes dedico este trabajo, por soportar estoicamente las horas interminables que he pasado alejado de ellos, y por ser las personas que me motivan a mejorar en la vida.

Miguel Ángel Román García
20 de diciembre de 2020

Síntesis en castellano

Introducción

Una obra musical se escribe en una partitura como un medio para establecer un marco de referencia común con el que los músicos puedan interpretarla. Sin embargo, las partituras no siempre están disponibles para los músicos, así que persiste la necesidad de transcribir aquella música que escuchan para poder interpretarla con sus instrumentos. Sin embargo, las aplicaciones potenciales de un sistema capaz de hacer transcripción automática son mucho más amplias, como las de asistencia en la composición, ayuda en el aprendizaje musical inspeccionando interpretaciones, o permitiendo el análisis musical de improvisaciones de jazz entre otras. La Transcripción Musical Automática (en inglés *Automatic Music Transcription*, AMT) es una tarea del campo de la Recuperación de Información Musical (en inglés, *Music Information Retrieval*, MIR) cuyo objetivo es convertir señales de audio en una forma de notación musical a través de medios computacionales.

En una partitura escrita en notación musical moderna, los sonidos se representan mediante notas que se colocan en un pentagrama, que está compuesto de 5 líneas horizontales proyectando la evolución del tiempo de izquierda a derecha. La posición vertical de una nota en el pentagrama determina su altura o frecuencia, y la forma de la nota determina su valor o duración. El silencio se representa por una figura especial distinta según su valor o duración. Un pentagrama empieza por una clave, que define la referencia de frecuencia para las notas escritas en él. Tras la clave, el compás determina la métrica del tempo, y la armadura indica qué notas deben ser alteradas de acuerdo con la tonalidad de la pieza musical. Basado en el compás de la pieza musical, las notas y los silencios se dividen en grupos de igual duración, también llamados compases, separados por una barra vertical. Las notas cuya duración traspasa el límite de un compás se unen a otra nota del compás siguiente de la misma altura a través de una línea curva llamada ligadura. Pero esto es tan sólo lo más básico, ya que una partitura contiene información adicional para indicar cambios de dinámica, que afectan al volumen y frecuencia de las notas, y de tiempo, que afecta a la duración de las mismas.

La mayoría de la música que escuchamos hoy en día es polifónica, en la que coexisten sonidos de diferentes notas musicales de forma simultánea. Por otro lado tenemos la música monofónica, en la que en cada instante solo está sonando la misma nota de una o varias fuentes de sonido. La transcripción de música polifónica es una tarea de una gran complejidad incluso para músicos experimentados. Esto es debido principalmente a que las ondas acústicas procedentes de distintos instrumentos se superponen en el medio aéreo, provocando una pérdida de información que es crucial para poder realizar la transcripción, y que ha de compensarse mediante un amplio conocimiento a priori del lenguaje musical utilizado, o de otra música similar perteneciente al mismo estilo.

Esta pérdida de información se manifiesta principalmente en la dificultad de identificar las frecuencias fundamentales de los instrumentos que están sonando, que en última instancia determinan las notas de la partitura en términos musicales. Cada sonido de un instrumento contribuye individualmente a la onda acústica con su frecuencia fundamental y con múltiplos de la misma (2x, 3x, etc.), llamados armónicos. La distribución de amplitudes y fases de los armónicos determinan el timbre particular de un instrumento. Al mezclarse frecuencias fundamentales y armónicos de distintos instrumentos en una única onda acústica, el proceso de extracción de las notas (o frecuencias fundamentales) de cada fuente de sonido de forma independiente se convierte en un gran reto.

Este problema se agrava debido a que la música, que se puede definir como el arte de combinar los sonidos en el tiempo, típicamente se forma a base de patrones comunes en el dominio del tiempo y de la frecuencia, que generan un efecto agradable en las personas. Estos patrones crean un ritmo particular, que hace que las notas de los instrumentos tiendan a comenzar a la vez, y una armonía concreta, que provoca que las notas emitidas compartan entre sí armónicos y frecuencias fundamentales. Sorprendentemente, esta relación de sonidos dificulta la tarea de extraer las notas individualmente a partir del audio, en mayor medida que si el audio estuviese formado por sonidos aleatorios o estadísticamente independientes.

Además, una partitura musical sólo es una guía del compositor para reproducir su obra, y está expuesta a múltiples matices de interpretación influenciados por la personalidad del músico, su estado de ánimo, cultura, contexto histórico, etc. La notación musical es un lenguaje muy particular, que ha evolucionado en el tiempo junto con el arte de la música, y como cualquier lenguaje creado por el ser humano contiene ambigüedades (e.g., diferentes formas de representar el mismo sonido), simplificaciones (e.g., elementos orientados a reducir la cantidad de símbolos), redundancias (e.g., símbolos para mejorar la legibilidad de la partitura), intención implícita (e.g., asunciones basadas en el tipo de obra musical) y en general muchos grados de libertad que dan lugar a diferentes expresiones musicales. En conclusión, esto hace que no haya una única forma de convertir una partitura en audio y viceversa, dificultando tanto la transcripción monofónica como la polifónica.

Sin embargo, sabemos que un músico experto es capaz de realizar la transcripción musical de una manera bastante notable, asumiendo que se puede escuchar la pieza musical varias veces y que se está familiarizado con el estilo de la misma. Esta habilidad es una de las formas más convincentes de inteligencia que existen, teniendo en cuenta los retos de los que hemos hablado. Wolfgang Amadeus Mozart era ampliamente conocido por su extraordinaria capacidad de transcripción musical. A él se le atribuye la transcripción del Miserere de Allegri, una obra coral de 5 voces que el Vaticano guardaba en secreto, después de escucharla dos veces en 1770 durante la semana santa (única vez al año en la que se podía escuchar). Esta anécdota evidencia de manera clara que la transcripción musical automática puede llegar a ser factible únicamente con la información extraída de la onda acústica, junto con el conocimiento previo de cómo se reproduce una partitura, o lo que es lo mismo, con un modelo de lenguaje musical predeterminado.

El Aprendizaje Automático (en inglés, *Machine Learning*) es un campo de las Inteligencia Artificial que estudia algoritmos creados a partir de la experiencia, en vez de a partir de la programación explícita. Esta experiencia se obtiene normalmente de un conjunto de datos con numerosos ejemplos, que se utilizan para crear un modelo estadístico que pueda ser capaz de tomar decisiones a partir de nuevos ejemplos nunca vistos. En la aproximación típica de aprendizaje supervisado, los ejemplos están formados por características y etiquetas, y el modelo estadístico constituye una función que conecta el espacio de características con el espacio de etiquetas, que son el objetivo del aprendizaje. Para que esta conexión sea posible, es necesario tener suficientes ejemplos que representen adecuadamente la distribución de ambos espacios, y que exista una correlación entre ambos de manera que las características contengan la información necesaria para generar la etiqueta más probable. La posibilidad de utilizar redes neuronales profundas dentro del Aprendizaje Automático, llamado Aprendizaje Profundo o *Deep Learning* en inglés, ha permitido la resolución de tareas con una precisión que resulta inalcanzable por los programas convencionales basados en reglas.

Una aproximación útil para la resolución de problemas consiste en la descomposición en varios subproblemas más pequeños, que se pueden resolver de manera más fácil y cuyas soluciones intermedias se unen para producir la solución final. El Aprendizaje Profundo también puede ser utilizado para la resolución de problemas de extremo a extremo, que implica que la descomposición del problema se delega al proceso de aprendizaje y no tiene por qué ser expuesta de forma explícita. Esto supone una gran ventaja frente a sistemas de resolución de problemas basados en múltiples etapas o subproblemas, ya que los errores de una etapa no se propagan a la siguiente. El área de reconocimiento del habla es un claro ejemplo de cómo los métodos extremo a extremo basados en Aprendizaje Profundo han superado ampliamente cualquier enfoque previo basado en la descomposición del problema en varias etapas de procesamiento [1].

Objetivos

A pesar de que la meta final de la transcripción musical automática es producir una partitura correcta, debido a la gran complejidad del problema, la comunidad científica alude con frecuencia el término *Automatic Music Transcription* (AMT) para referirse a la obtención de un objetivo intermedio [2]. La referencia a AMT más extendida que podemos encontrar en la literatura se llama estimación de múltiples alturas, cuya meta es la extracción de todas las frecuencias fundamentales en cada instante de tiempo. En el siguiente nivel de la transcripción musical automática, la meta es identificar qué notas musicales (no frecuencias) están siendo reproducidas por los instrumentos en cada instante. Las notas musicales, caracterizadas por su nombre, tiempo de comienzo y de fin, se colocan en una notación musical llamada pianola (en inglés, *piano-roll*), que es una representación en dos dimensiones de la música donde el eje de ordenadas contiene todas las alturas posibles de menor a mayor frecuencia, y el eje de abscisas muestra la evolución de las notas en el tiempo. El siguiente paso natural hacia la meta final de la transcripción musical automática es el de identificar la fuente de sonido de cada nota, con el fin de obtener un *piano-roll* por cada instrumento, lo cual permitiría la transcripción de fragmentos de audio donde dos instrumentos emiten la misma nota a la vez. Esta tarea está estrechamente relacionada con la separación de distintas fuentes de audio.

Todos estos niveles de la transcripción musical automática han mostrado un éxito muy limitado en la literatura científica, y siguen siendo considerados problemas abiertos, principalmente debido a los retos intrínsecos del solapamiento armónico para música polifónica que hemos visto previamente. Por otro lado, el *piano-roll* no resulta una notación útil para que un músico pueda interpretar o analizar fácilmente la música representada. Por tanto, la necesidad de un sistema que produzca una partitura musical a partir del audio, que es la meta final de la transcripción musical automática, sigue prevaleciendo.

Para lograr obtener una partitura completa, una aproximación consiste en procesar el *piano-roll* para darle un sentido musical al nombre de las notas (e.g., elegir entre Sol♯ o La♭) y convertir los tiempos de inicio y fin de las notas en una duración canónica (e.g., elegir entre ♪.. o ♩). Además, se deben identificar otras estructuras musicales como el compás, la armadura, la posición de las barras de compás, los silencios, calderones, símbolos que afectan a la dinámica y la expresión musical, etc. Y finalmente, las reglas de formato de partitura deben aplicarse, como las que regulan la colocación de notas en pentagramas, claves, ligaduras, repeticiones, corchetes, etc. Existen varias herramientas que convierten un fichero de tipo MIDI, que almacena la información musical en un formato similar al *piano-roll*, en una partitura. Sin embargo, todavía están lejos de funcionar de manera totalmente autónoma, requiriendo de mucha intervención manual para producir la partitura deseada.

Otra aproximación, que es la que hemos seguido en esta tesis doctoral, trata de solucionar el problema en un único paso, aprendiendo la relación directa entre audio y partitura a partir de muchos ejemplos. El propósito de esta tesis es crear un esquema de referencia que realice la transcripción musical automática de extremo a extremo o de manera holística, es decir sin múltiples fases de procesamiento, gracias a la capacidad de las redes neuronales profundas. A esta tarea la llamamos Audio a Partitura (en inglés, *Audio-to-Score*, A2S). La entrada a nuestro esquema de referencia es la señal acústica representando un fragmento musical, y la salida es la secuencia de símbolos que se pueden obtener directamente en la partitura de música correspondiente.

Para lograr dicho propósito, nos planteamos los siguientes objetivos:

1. Buscar una arquitectura basada en redes neuronales, donde la entrada es la representación de una señal acústica de una longitud variable, y la

salida es una secuencia de caracteres también de longitud variable que representa su partitura.

2. Evaluar exhaustivamente el método propuesto para música monofónica, probando diferentes representaciones de entrada y de salida, y analizando el origen de los principales errores en la partitura estimada.

3. Validar el método propuesto con música polifónica, donde al reto de producir una partitura válida a partir del audio, presente también en la transcripción monofónica, se le añade el reto de detectar múltiples frecuencias simultáneas con solapamiento de armónicos.

Trabajos

Publicación I

Referencia:

- Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 34–41, Paris, France, September 2018. ISMIR

Este trabajo presenta un nuevo esquema de referencia para realizar transcripción musical automática de sonido a partitura, incluyendo la formulación del problema extremo a extremo con redes neuronales. Con el fin de probar la viabilidad del método, nos basamos en la arquitectura de una red neuronal Convolucional-Recurrente (*Convolutional Recurrent Neural Network* en inglés, CRNN) ya aplicada con éxito en el campo de reconocimiento del habla, que aunque sea una tarea diferente a la que estamos tratando resulta conveniente por utilizar los mismos tipos de datos a la entrada y a la salida. Tal y cómo se hace también en reconocimiento del habla, para poder entrenar con datos de entrada y salida no alineados en el tiempo nos apoyamos en el algoritmo *Connectionist Temporal Classification* (CTC) [5]. Como formato de entrada utilizamos el espectrograma del audio obtenido con la Transformada de Fourier de Tiempo Corto (*Short-Term Fourier Transform* en inglés, STFT) y para la salida creamos nuestro propio formato simple de codificación de partitura para música monofónica.

Para entrenar el modelo creamos un conjunto de datos de música monofónica a partir de la base de datos RISM (*Répertoire International des Sources Musicales*) [9], que es un catálogo de manuscritos de música clásica europea principalmente del periodo comprendido entre los años 1500 y 1800. En el catálogo de RISM no se almacena el contenido de las partituras, sino que contiene pequeños extractos (*incipits* en inglés) de algunas obras en formato de partitura digital. Estos extractos son utilizados habitualmente como parte de los índices de contenido de los manuscritos originales. En total se han usado más de 70.000 pequeños fragmentos de este tipo, generando el audio a partir de los mismos mediante herramientas de síntesis MIDI. Para esto se ha usado un sonido de piano como instrumento, y un tiempo de negra aleatorio en cada ejemplo para conseguir variabilidad en el tempo.

Con respecto a la evaluación del método que proponemos, y dado que no existe ninguna métrica específica para convertir una señal de audio a partitura (A2S), nos apoyamos en las métricas utilizadas con anterioridad en el reconocimiento del habla. Estas miden la distancia de edición entre la secuencia de caracteres de salida estimada y la salida esperada, tanto a nivel de palabras *Word Error Rate* (WER) como a nivel de caracteres *Character Error Rate* (CER). Además hemos añadido una métrica ideada para evaluar sistemas de AMT basados en *piano-roll* como salida [7] y la hemos adaptado a nuestro caso particular de A2S. Dicha métrica mide los errores de altura de las notas (E_p) los errores de duración de notas y silencios (E_d), los errores de notas y silencios no contabilizados (E_m) y los errores de notas y silencios sobrantes (E_e). Los resultados revelan que el método es capaz de aprender la tarea de AMT en nuestro entorno limitado, validando que esta aproximación al problema de AMT es prometedora y por tanto merece la pena profundizar más en la investigación.

Publicación II

Referencia:

- Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A Holistic Approach to Polyphonic Music Transcription with Neural Networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 731–737, Delft, The Netherlands, November 2019. ISMIR

Con el fin de probar que el método propuesto también puede funcionar con audio polifónico, en este trabajo se ha entrenado el modelo con música polifónica extraída del repositorio de partituras digitales *humdrum-data* [13]. En concreto se ha elegido música de 4 voces de las corales de Bach y de los cuartetos de cuerda de Haydn, Mozart y Beethoven. Para el experimento con las corales de Bach utilizamos audio previamente sintetizado con una fuente de sonido de órgano de tubos con muy alta calidad, y usando una coral completa por cada muestra de entrenamiento, obteniendo casi 6 horas de audio en total. Previamente se ha ido eliminando del audio de forma manual las repeticiones de secciones musicales, ya que no proporcionan información nueva y alargan innecesariamente el tamaño de las muestras de entrenamiento. Para el experimento con el conjunto de datos formado por los cuartetos de cuerda, se ha dividido cada una de las obras en fragmentos más cortos de 3 a 6 compases, que se han sintetizado con distintas fuentes de sonido MIDI según la tesitura del instrumento para el que fue escrito (violín, viola o violonchelo). En este caso se ha elegido para cada fragmento un tiempo de negra aleatorio dentro del rango esperado según el tipo de composición (por ejemplo, Allegro, Adagio, etc.). Con esta aproximación se han obtenido más de 20 horas de audio en total.

Muchas de las partituras digitales en formato ***kern* [6] que utilizamos contenían errores de cuantificación de las notas, que hubo que arreglar de forma manual con el fin de limpiar los datos de entrenamiento y no perjudicar el proceso de aprendizaje. Los repositorios con estas partituras digitales corregidas se han dejado disponibles en el dominio público.

En ambos casos, para generar las características del audio usado como entrada de la red se ha usado la Transformada de Fourier de Tiempo Corto, aplicando después un agrupamiento logarítmico de los intervalos de frecuencia alineados con las octavas de la música tonal con la que estamos trabajamos. La salida del modelo propuesto es directamente una secuencia de símbolos en formato ***kern*, incluyendo los caracteres de presentación como el tabulador para cambiar de voz y el salto de línea para avanzar en el tiempo.

En este caso se ha seguido utilizando la misma arquitectura basada en redes CRNN del trabajo anterior, aunque modificada para soportar secuencias más largas de salida. El algoritmo de CTC nos impone un número de caracteres en la salida menor o igual al número de ventanas (*frames*) de audio de entrada. Por tanto para poder entrenar con partituras polifónicas, con una mayor densidad de caracteres por segundo, una opción sería aumentar artificialmente el número de ventanas del audio de entrada, suponiendo un coste computacional innecesario. Para evitarlo, se ha incrementado el número de pasos de la etapa recurrente mediante el desdoble de las características extraídas del audio en la etapa convolucional, manteniendo un número de ventanas más adecuado en la entrada. Además se han ajustado todos los hiperparámetros con respecto a la publicación anterior para mejorar al máximo los resultados y acelerar la convergencia en la fase de entrenamiento.

Los resultados revelan unos valores de *Word Error Rate* (WER) y *Character Error Rate* (CER) mejores de lo esperado teniendo en cuenta que el modelo tiene que resolver el reto del solape de armónicos característico de la transcripción polifónica. Además hay que considerar que para el caso particular de la tarea de transcripción musical de audio a partitura con música polifónica existe un nuevo reto que hay que resolver, que consiste en colocar las notas estimadas en la voz correcta. Cada voz tiene un rango de notas distinto, que facilita la colocación de las notas estimadas por orden de altura. Pero muchas veces en una obra musical las voces se cruzan, generando una situación de ambigüedad. Para resolver esta ambigüedad el modelo debería discriminar en base al timbre de cada nota, lo que se corresponde directamente con otro problema de identificación de armónicos, o al seguimiento de voces característico del estilo de música en cuestión, que precisa del conocimiento aportado por un sólido modelo de lenguaje musical. Este hallazgo refuerza la idea de que el enfoque extremo a extremo es más adecuado que los enfoques tradicionales para resolver el problema de AMT de forma fiable.

Publicación III

Referencia:

- Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. Data representations for audio-to-score monophonic music transcription. *Expert Systems with Applications*, 162:113769, 2020

En este trabajo ponemos el foco en la evaluación del método propuesto en la Publicación I, realizando numerosos experimentos con más de 300.000 fragmentos de música monofónica sintetizada con distintos timbres y aplicando también variabilidad en el tempo. Por un lado se ha evaluado el sistema con distintas representaciones de entrada y de salida, realizando un estudio comparativo exhaustivo. Por otro lado se han analizado individualmente los errores de transcripción para comprender mejor su origen.

Tipos de representaciones de entrada evaluadas:

- Señal acústica (*Raw waveform*). En este caso se utiliza la onda acústica directamente como entrada del modelo, dividiéndola en fragmentos con una ventana temporal de un tamaño fijo, y usando SincNet [8] como primera capa de la red neuronal. SincNet extrae características de estos fragmentos de audio aprendiendo filtros paso-banda como parte del proceso de entrenamiento del modelo.

- *STFT*. Es el espectrograma que se obtiene con la Transformada de Fourier de Tiempo Corto (STFT en inglés).

- *LogSTFT*. Es el espectrograma basado en STFT al cual se le aplica un banco de filtros con espaciado logarítmico, de manera que los intervalos de frecuencia se alinean con los intervalos de las notas musicales. El objetivo es conseguir un número fijo de intervalos de frecuencia dentro de cada una de las octavas.

- *CQT*. Es el espectrograma que se obtiene al aplicar la transformada CQT (*Constant-Q Transform*) [15], donde los tamaños de ventanas temporales varían en función de los intervalos de frecuencia. El objetivo de esta transformada es conseguir que la resolución en frecuencia sea constante en todos los intervalos, con el coste de una peor resolución temporal para las frecuencias más bajas.

Tipos de representaciones de salida evaluadas:

- *CTC-friendly.* Es una notación de música creada específicamente para facilitar la decodificación del algoritmo de CTC, orientada a música monofónica.

- *Basada en PAE.* Esta notación está basada en el formato PAE (*Plaine And Easie code*) [3], añadiendo espacios para crear artificialmente palabras que permitan medir el error WER como en el resto de representaciones de salida. También ha sido necesario asegurar que los símbolos para cambiar de octava y de duración de las notas aparecen siempre en el mismo orden, y justo antes de la primera nota o silencio al que afectan para evitar ambigüedades en la representación. El formato PAE sólo puede representar un único pentagrama, así que no es adecuado para las necesidades de la música polifónica.

- *Basada en Kern.* Esta notación está basada en la representación ***kern* [6] del formato Humdrum, que soportan el amplio espectro de partituras con notación moderna, incluyendo varios pentagramas representando voces, ornamentación, cambios de tempo y dinámica, etc.

Para este trabajo hemos empleado la misma arquitectura basada en redes CRNN con CTC en la salida, pero se ha ajustado el tamaño de la red convolucional en función del número de características del audio de entrada. El objetivo de este ajuste es conseguir que todas las representaciones de entrada generen el mismo número de características antes de la etapa recurrente. De esta manera ninguna representación de entrada obtiene una ventaja con respecto a las demás en cuanto a capacidad de aprendizaje.

En cuanto a las métricas de evaluación, utilizamos las mismas que en el primer trabajo; por un lado tenemos las heredadas de la tarea de reconocimiento automático del habla, como son el WER y el CER, y también las heredadas del enfoque de AMT basado en una salida de *piano-roll* que hemos adaptado a nuestro problema. Además clasificamos todos los errores en la salida de forma individual para analizar cuales son las estructuras musicales que son más difíciles de identificar correctamente por nuestro modelo.

Los mejores resultados se han dado con la combinación de representación de entrada *LogSTFT* y de representación de salida *CTC-friendly*. No obstante lo más interesante de los resultados, fue comprobar que incluso para el caso de música monofónica, la tarea de A2S presenta el reto de identificar correctamente el valor de las notas (e.g., 𝅗𝅥 , ♩ , ♪ , etc.) que está relacionado con su duración. No obstante, la duración real del valor de las notas varía dentro de unos márgenes razonables de una pieza a otra, e incluso entre dos interpretaciones distintas de la misma pieza. Esto hace que la estimación de dichos valores, llamada cuantificación de notas, sea una tarea de gran dificultad que requiere un contexto mayor de audio para ser resuelta.

Relacionado con la cuantificación de notas, la información métrica de la partitura es otra de las estructuras musicales más difíciles de identificar correctamente por nuestro modelo. Esto incluye el compás de la obra (e.g., $\frac{4}{4}$, $\frac{3}{4}$, $\frac{6}{8}$, etc.), y la localización exacta de las barras de compás y silencios (e.g., 𝄼 , 𝄽 , 𝄾 , etc.). La información métrica es fundamental para generar compases de la misma duración a partir de las notas estimadas, y tiene un gran impacto en la interpretación ya que determina implícitamente las notas que se acentúan dentro de cada compás. La información métrica no está codificada explícitamente en cada ventana de audio, y al igual que ocurre con la cuantificación de notas, sólo se puede inferir a partir del contexto completo del audio y del modelo de lenguaje musical aprendido.

Conclusiones

Esta tesis doctoral presenta un nuevo enfoque en el área de la transcripción musical automática (AMT), definiendo la tarea de *Audio-to-Score* (A2S), que realiza la transcripción musical de extremo a extremo gracias a la capacidad de modelado de problemas que nos ofrecen las redes neuronales profundas. Este enfoque va un paso más allá de los sistemas de transcripción tradicionales, que están basados en predecir notas musicales en el formato de tiempo-frecuencia llamado (*piano-roll*). Las principales ventajas del enfoque propuesto frente a los métodos tradicionales son las siguientes:

- La salida es una partitura válida de música que puede ser directamente interpretada por músicos o analizada por musicólogos.

- La aproximación extremo a extremo evita que los errores de una etapa se propaguen a la siguiente.

- No precisa de anotaciones de alineamiento temporal entre el audio de entrada y la partitura de salida, dado que se aprende por el modelo de forma implícita.

- Mediante la aproximación extremo a extremo se aprende también un modelo de lenguaje musical que ayuda a reducir los errores de transcripción de manera global.

Los resultados sobre audio monofónico, donde no existe el problema de solapamiento de armónicos, muestran que el origen de la mayoría de errores está relacionado con la información métrica de la partitura tales como la duración de notas y silencios, la estimación del compás de la obra y la colocación de barras de compás. Dado que esta información no está codificada explícitamente en la señal de audio, la estimación correcta de esta información requiere un potente modelo de lenguaje musical previamente aprendido. La representación de entrada que ha obtenido los mejores resultados es el espectrograma con intervalos de frecuencia en escala logarítmica y alineados con las octavas musicales. La mejor representación de salida obtenida fue la que se creó específicamente para facilitar la decodificación *Connectionist Temporal Classification* (CTC) de nuestro modelo, tal y como se pretendía.

Los resultados sobre audio polifónico, a pesar de la naturaleza limitada de los experimentos, revelan que el método propuesto también es capaz de aprender a superar cada uno de los retos de AMT en el dominio del tiempo (e.g. cuantificación de notas) y en el dominio de la frecuencia (e.g. solapamiento de armónicos), pudiendo predecir las frecuencias fundamentales que determinan las notas y adicionalmente su asignación a la voz correspondiente.

Aunque la formulación de A2S presentada en esta tesis doctoral todavía está lejos de ser resuelta de una manera satisfactoria, creemos que los resultados arrojados por los experimentos abren una nueva vía de investigación en el campo de transcripción musical automática. A pesar de ello, existen dos factores importantes que en la actualidad limitan el progreso de la tarea A2S:

1. Aunque existen muchas grabaciones de audio con su correspondiente partitura digital, los requisitos de nuestro esquema de referencia con la formulación actual (i.e., fragmentos cortos de audio real polifónico con su correspondiente notación musical) limitan la cantidad de datos que pueden usarse para entrenar los modelos.

2. No existe una métrica de evaluación estándar para A2S con música polifónica que pondere los errores en base a su impacto musical, en lugar de tener en cuenta solo la distancia de edición entre la partitura digital estimada y la esperada. Una métrica adecuada también nos permitiría una comparación más precisa entre distintos métodos.

Además de trabajar para superar estos factores limitantes, existen otras vías por las cuales este trabajo puede tener continuidad:

1. Añadiendo un modelo de lenguaje en el decodificador, que puede mejorar sustancialmente los resultados al asegurar que la salida es sintácticamente y gramáticamente correcta desde un punto de vista musical. Este modelo de lenguaje puede entrenarse de manera no supervisada utilizando únicamente partituras digitales que están disponibles en el dominio público en grandes cantidades.

2. Evaluando modelos autoregresivos con mecanismos de atención, como por ejemplo la arquitectura *Transformer* [16], para así evitar la limitación impuesta por CTC que impide tener secuencias de salida de mayor longitud que el número de ventanas de audio presentes en la entrada.

3. Aumentando la cantidad de datos para mejorar la robustez de los modelos entrenados, aplicando diversas transformaciones al sonido como la mezcla con ruido añadido, variaciones de frecuencia y rango dinámico, síntesis de audio con varias fuentes de sonido distintas por cada tipo de instrumento, etc.

4. Creando un conjunto de datos específico para la tarea A2S que utilice audio real, después de trocear grabaciones de audio existentes automáticamente mediante una herramienta de seguimiento de partituras (*Score Following* en inglés).

Del mismo modo que el procesamiento de lenguaje natural (*Natural Language Processing* (NLP en inglés) ha experimentado un importante salto cualitativo en los últimos años gracias al uso de modelos más grandes como GPT-3 [4], la tarea A2S también podría beneficiarse si tuviéramos a nuestra disposición tal capacidad de cómputo. Las redes neuronales de mayor tamaño pueden procesar secuencias más largas,lo que permitiría entrenar directamente con grabaciones de audio real y su correspondiente partitura completa disponible en el dominio público. Recopilar los datos con esta aproximación es menos costoso, lo cual podría incrementar el tamaño de las muestras de entrenamiento en varios ordenes de magnitud. En definitiva, al incrementar la cantidad de datos de entrenamiento y el tamaño de los modelos de redes neuronales seríamos capaces de aprender mejores modelos de lenguaje musical, que puede ser la clave que permita solventar algunos de los retos que son intrínsecos a la transcripción automática de música.

Contents

Preface

Given that most of the research conducted as part of this thesis has been published in international peer-reviewed journals and conferences, this dissertation is arranged as a *thesis by publication*. This means that the core of the work is presented as reprints of such publications, keeping their original format.

The set of papers that form the body of work of this thesis are (in chronological order of publication):

1. Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 34–41, Paris, France, September 2018. ISMIR

2. Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A Holistic Approach to Polyphonic Music Transcription with Neural Networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 731–737, Delft, The Netherlands, November 2019. ISMIR

3. Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. Data representations for audio-to-score monophonic music transcription. *Expert Systems with Applications*, 162:113769, 2020

Following the guidelines of the doctoral school of Universidad de Alicante for writing thesis as compilation of papers, this document is organized as follows:

- **Part I: Introduction**. An initial section introducing the background of the thesis and a description of the set of contributions within the context of the thesis project.

- **Part II: Published work**. The compilation of papers that are already published.

- **Part III: Conclusions**. Summary of the contributions, general conclusions, and some lines about future research.

Considering that each paper presented in Part II is completely self-contained, Part I merely puts into context the research carried out without giving a deep insight into the background and related works. Similarly, the analysis of the results achieved can be found in the corresponding section on each publication, thus Part III only covers the most important conclusions of the whole work.

Part I

Preamble

Chapter 1

Introduction

A musical piece can be written in a music score notation, as a means to set a common framework for musicians at the time of its performance. However, music scores are not always available to music practitioners, so there is still a need for them to transcribe the music they listen to and wish to play or analyze. But the potential applications of an automated system capable of doing music transcription are much broader, such as aiding at music composition, helping in music learning by inspecting performances, or carrying out music analysis of jazz improvisations among others.

Automatic Music Transcription (AMT) is a Music Information Retrieval (MIR) field that aims at converting audio signals into a form of musical notation using computational means. Considering the complexity of this problem, most AMT methods perform multi-pitch detection followed by a post-processing stage that extracts the notes and other components of a music score. The purpose of this thesis is to create a framework that performs AMT in an end-to-end fashion. The input of the proposed model is the audio waveform representing a music fragment, and the output is the sequence of symbols (i.e., a digital score) that can be rendered into the corresponding human-readable score.

In a modern music score [14], sounds are represented by *notes* placed on a *staff*, which comprises 5 horizontal lines representing the flow of time from left to right. The vertical position of the note in the staff determines its pitch, and the shape of the note symbol determines its duration. Silence is represented by *rest* symbols, whose duration is also determined by its shape. A staff begins with a *clef* symbol, which defines the pitch reference for the written notes. After the clef, a *time signature* determines the metric information or rhythm, and a *key signature* determines which note pitches must be altered according to the tonality of the musical piece. Based on the time signature, notes and rests are divided in *measures* of equal duration using *barlines*. Notes whose duration go beyond the current measure are joined with another note of the same pitch in the next measure using a curved line named *tie*. Anyway, this is only the basics, as a modern music score usually contains other information to indicate changes in *dynamics* (i.e., affecting the volume of notes) and *tempo* (i.e., affecting the length of notes).

1.1 Main Challenges

Most of the music we listen today is polyphonic, which means that different sound events can be played simultaneously. On the other hand we have monophonic music, where only one pitch is playing at any given time. Polyphonic music transcription is a very challenging task even for trained musicians, due to the fact that sound waves from different sources are mixed together. This addition of waves implies a loss of information, which musicians need to complete based on their own experience with the music they play, listen or create.

One of the main AMT challenges is identifying the fundamental frequencies of all the sound events being produced by instruments, which determine the *pitches* in music language terminology. Each individual *pitch* contributes to the final audio waveform with its fundamental frequency together with its multiples (2x, 3x, etc.), named harmonics. The distribution of harmonic amplitudes and phases determines the particular *timbre* of an instrument. As fundamental frequencies and harmonics merge into a single audio signal, extracting back the fundamental frequencies of each independent source becomes a very demanding task.

Another reason making polyphonic music transcription challenging is the fact that sounds are not statistically independent in modern music. Music can be defined as the art of combining sounds in time, which hints that sound waves usually share common patterns in both time and frequency domains. These patterns build a particular rhythm, where note onsets and offsets occur at the same time, and a particular harmony, where notes share their fundamental frequencies or harmonics. This note dependency makes the task of separating the individual notes present in the audio signal more difficult than if they were made of random sound events.

And last but not least, music symbols can only be a guideline to perform a particular piece of music, and are subject to different interpretation nuances based on the musician character, mood, culture, historical context, etc. Music notation is a particular language that evolved together with the art of music, and as any other human-made language contains ambiguities (e.g., different ways of representing the same sound), simplifications (e.g., elements aimed at reducing the total number of symbols), redundancies (e.g., symbols to improve readability of the score), implicit intention (e.g., assumptions based on the type of the piece) and in general many degrees of freedom that may lead to different musical outcomes. In summary, there is no unique way to convert a music score into sound waves, and vice versa. This particular challenge applies to both monophonic and polyphonic music transcription.

Nevertheless, we know that expert musicians are capable of performing music transcription fairly well, provided that they listen to the musical piece several times and they are familiar with the style of the music being transcribed. This ability is one of the most compelling traits of human intelligence considering the aforementioned challenges. Wolfgang Amadeus Mozart was highly renowned by his extraordinary music transcription skills. He was allegedly able to transcribe Allegri's Miserere, a 5-voice choral piece the Vatican held in secrecy, after hearing it twice on the Easter of 1770. This anecdote is a strong evidence of how AMT is feasible just with the information carried by the audio waveform, together with prior knowledge of how written music should sound, or in other words, with a predetermined music language model.

1.2 Research Background

Even though the ultimate goal of AMT is producing a music score, due to its complexity, AMT is very often referred as methods aiming at some intermediate goal [2]. The most extended AMT objective that we can find in the literature is called multi-pitch estimation, whose goal is to obtain all the fundamental frequencies at any given time frame. In the next level of AMT, namely note tracking, the goal is to identify which notes (rather than pitches) are being played by instruments. The notes characterized by its pitch, onset and offset times, are placed in a piano-roll music notation format, which is a 2-dimensional representation of music where the y-axis contains all the possible notes ordered by frequency and the x-axis provides the evolution of the notes in time. The next natural step towards complete AMT is to identify the source of each note in order to generate one piano-roll per instrument, which allows the transcription of audio excerpts where two instruments are playing the same note simultaneously. This task is called stream level AMT, and is closely related to audio source separation.

These levels of AMT have shown limited success in the literature and they are still regarded as open problems, mainly due to the intrinsic challenge of harmonic overlapping previously mentioned. Furthermore, piano-rolls are not useful for humans to perform the underlying music they characterize, hence the need to obtain a music score, the final goal of AMT, still prevails.

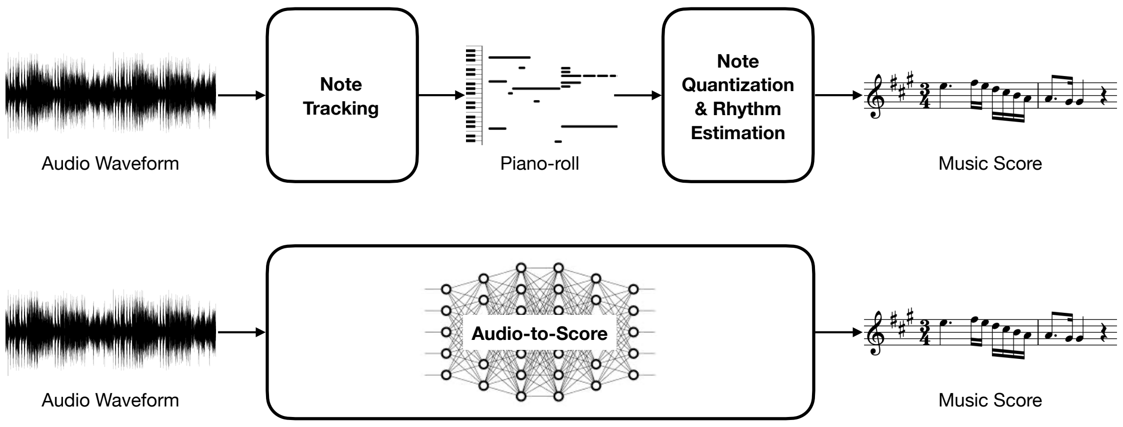
In order to satisfy this need, one approach is to try converting notes from a piano-roll notation into a music score, by extracting pitch spellings (e.g., G♯ or A♭) and timing quantization (e.g., ♪.. or ♩). Additionally, other music structures must be properly identified such as the time and key signatures, and the location of barlines, rests, fermata, dynamics and expression symbols, etc. And finally, typesetting rules must be obeyed, such as those regarding staves, clefs, note ties, repetitions, note beaming, etc. There are many proprietary software tools that convert MIDI files, a format that can store music in a piano-roll style, into a music score. However, they are still far from being fully automated, requiring a great deal of human intervention in order to render a valid score.

The other approach, which is the subject of this thesis, aims at solving the audio-to-score problem in one step, by learning the direct mapping between audio and score from many examples. Both approaches are depicted in Figure 1.1.

1.3 Proposed Approach

Machine Learning (ML) is the field of computer science that studies algorithms created by experience, rather than by explicit programming. This experience is normally obtained from data samples, which are used to build a statistical

Figure 1.1: Overview of A2S approaches. Top: multi-step methods based on piano-roll estimation. Bottom: end-to-end methods based on deep neural networks and the object of this thesis.



model that is capable of making decisions on new unseen data samples. In the typical supervised learning approach, samples are made of features and corresponding labels, and the statistical model constitutes a function that maps the input feature space to the output label space, which is the learning objective. This mapping is possible provided that there are enough data samples to represent the input and output spaces, and that the information required to generate the most probable label is somehow encoded in the input features. The possibility of using deep neural networks for ML, called Deep Learning (DL), has enabled a new set of applications that were previously unreachable with rule-based conventional programs.

A useful approach to problem-solving involves decomposing the problem into several smaller sub-problems, which can be solved independently, and together produce the final solution. Interestingly, DL models can be used as end-to-end problem-solvers, in the sense that the problem decomposition can be automatically delegated to the learning process and does not have to be made explicit. This is a clear advantage versus multi-step problem solvers, as the errors in one step do not cascade to the next one. Automatic Speech Recognition (ASR) is one example that shows how end-to-end DL methods outperform any previous approach based on multiple processing stages [1].

In order to validate this end-to-end framework that performs AMT using deep neural networks, we need a large amount of paired audio and score data samples. In an ideal scenario, we could just use existing music recordings and their published digital scores. Nevertheless, considering the typical duration of recorded music is in the range of several minutes, the amount of memory and computation required to process such long audio samples is as of today out of our reach. Contrary to the analogous speech recognition task, where the audio can be broken up by sentences or paragraphs as needed, music audio waves cannot be easily split at desired locations of the score. Moreover, all existing datasets targeting the AMT task are based on piano-roll output representations and lack music notation data to fulfill our purpose.

The scarceness of proper datasets, together with the lack of standard metrics to formally compare results, hinder A2S progress and force researchers to work on proof of concepts that are still far from being commercially viable solutions.

Chapter 2

Contributions

This chapter outlines the main contributions of this thesis concerning the field of Automatic Music Transcription (AMT). Since notation-level AMT is a fairly unexplored area of research, the main focus of this thesis is to validate a particular framework to tackle the AMT problem rather than to improve existing results or methods.

2.1 Audio-to-Score Formulation

We outline the Audio-to-Score (A2S) problem as a pattern recognition task. Let $\mathcal{X}$ be the domain of audio files and Σ the alphabet symbols that can render a valid score in western music notation. The aim of our A2S task is to find a function that maps any audio file into a sequence of symbols, i.e., a function $f : \mathcal{X} \rightarrow \Sigma^*$. The A2S task is therefore formulated as retrieving the most likely sequence of music symbols $\hat{\mathbf{s}}$ given an $\mathbf{x} \in \mathcal{X}$:

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \Sigma^*} P(\mathbf{s}|\mathbf{x}) \tag{2.1}$$

A Convolutional Recurrent Neural Network (CRNN) is implemented to model the posterior probability of equation 2.1, which has proven to be successful on similar sequence translation tasks like, for instance, speech recognition [1]. The input of our model is the audio information condensed in fixed time periods or frames. The output is the vector of likelihoods of each symbol in the alphabet at every input frame. Connectionist Temporal Classification (CTC) algorithm [5] facilitates training such model, as it takes into account all possible alignments of the likelihood vectors with the correct sequence of symbols, avoiding the hassle of manually annotating alignments in the ground truth data.

As there are no standard metrics to evaluate the performance of an A2S task, we adopt the metrics from the analogous speech recognition task, namely Word Error Rate (WER) and Character Error Rate (CER). They measure the transcription error by computing the edit distance between the prediction and the ground truth, at the word-level and character-level respectively. The edit distance is not the best indicator of music score errors, as it does not take into account their impact in terms of musical outcome. Nevertheless, we show that these metrics provide a useful way not only to evaluate our results, as the edit distance is directly related to the number of manual operations needed to correct the output score, but also to compare our own experiments.

2.2 Data Acquisition

The lack of large corpora of short sample pairs suitable to train our A2S models drove us into building our own data set. The approach we followed was to search for databases of music scores already in symbolic form. Being able to parse these scores gave us the flexibility to randomly split them into shorter fragments, and create the corresponding audio using standard MIDI synthesis tools. We sought to approximate some aspects of real audio by using different timbres and tempos. We understand synthetic data is not ideal for AMT, but it serves as a starting point to validate the feasibility of the A2S framework we are proposing in the absence of more suitable datasets.

2.2.1 RISM

The RISM (*Répertoire International des Sources Musicales*) database [9] is a catalog of over 1 million music manuscript sources, with an emphasis of western classical music from 1500 until 1800. The database does not include music scores in printed form, but it contains incipits of some of the music sources. An incipit is a small fragment extracted from the original score, normally found in sheet music for indexing purposes. The incipits are stored using PAE [3] format, which is a symbolic representation of one staff of music score.

The whole RISM database can be downloaded as a single XML file. We built a library[1] to parse this large XML file and extract the valid incipits from it, together with other metadata such as the key signature and tonality, time signature, clef and target instrument. The instrument metadata was used to select the corresponding MIDI program for audio synthesis, while the rest of the metadata were used to complement the incipit in order to create a complete music score. Figure 2.1 shows the multiple steps required to obtain our ground truth out of the RISM database (more details available in [12]).

Some of the incipits contained format errors, for example missing notes or rests that led to scores with incorrect metric information. Errors in training data act as noise, hurting the performance that can be achieved by machine learning models. Thus, we had to parse incipits in order to verify that each measure of the selected scores contained the correct number and lengths of notes and rests, according to the time signature of the incipit. Another example of format errors were ties placed between notes of different pitch (probably confused by slurs in the original score), which we also had to properly detect and fix.

2.2.2 Humdrum-data

Humdrum-data [13] is a repository of digital music scores in ***kern* [6] representation, a text-based file format that can render scores with multiple staffs.

[1]https://github.com/mangelroman/rism

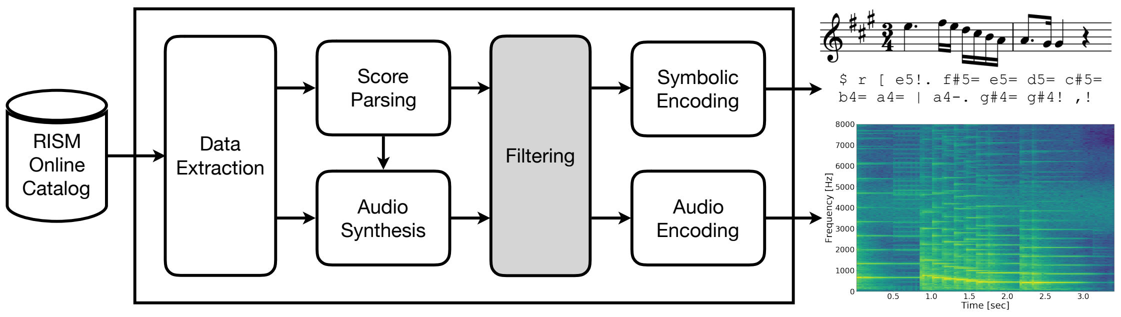


Figure 2.1: Training data acquisition pipeline for RISM database.

We used this data source to extract 4-voice music scores in order to validate our approach for polyphonic music. We created a **kern parser to split long scores randomly without causing formatting errors. Some of these contributions were submitted to humextra repository, a popular open source library to manage ***kern* files, and were accepted by its creator.[2]

As in the case of RISM database, many of the ***kern* scores we selected contained note quantization errors that needed to be corrected to avoid adding noise to our training data. Unfortunately, fixing these errors programmatically was not a simple task, so we had to manually edit the scores one by one. 2.2 provides an example of the nature of manual fixes we had to apply. The fixed scores are available in several public repositories.[3,4,5] Some of these fixes were also accepted in the official humdrum-data public repository.[6]

Since most scores from this database were missing metronome markings, we used tempo indications made by the composer (e.g. Allegro, Adagio, etc.) in order to later synthesize with a random tempo that is within the expected range. Figure 2.3 depicts the data acquisition pipeline for humdrum-data repository (more details available in [11]).

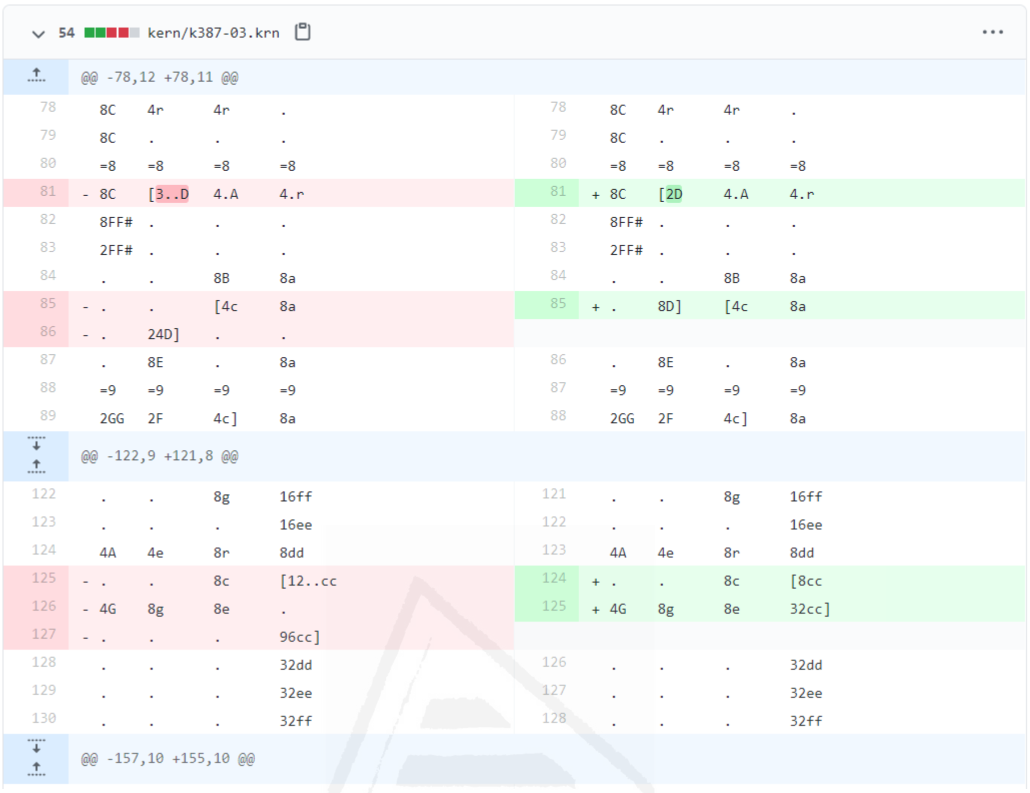
2.3 Input and Output Representations

We tested our A2S framework with different representations of input and output data, and carried out a thorough comparative analysis to evaluate the performance of our model considering all possible combinations (results and conclusions are fully detailed in [12]).

2.3.1 Input

- *Raw audio.* Raw audio waveforms were directly used as input by adding SincNet [8] as the first layer of the model. SincNet layer extracts audio features from waveforms by learning meaningful filters.

[2]`https://github.com/craigsapp/humextra/pull/12`
[3]`https://github.com/mangelroman/humdrum-haydn-quartets`
[4]`https://github.com/mangelroman/humdrum-mozart-quartets`
[5]`https://github.com/mangelroman/beethoven-string-quartets`
[6]`https://github.com/craigsapp/bach-370-chorales/pull/1`



```
54 kern/k387-03.krn

@@ -78,12 +78,11 @@
 78     8C    4r    4r    .          78     8C    4r    4r    .
 79     8C    .     .     .          79     8C    .     .     .
 80     =8    =8    =8    =8         80     =8    =8    =8    =8
 81   - 8C    [3..D 4.A   4.r        81   + 8C    [2D   4.A   4.r
 82     8FF#  .     .     .          82     8FF#  .     .     .
 83     2FF#  .     .     .          83     2FF#  .     .     .
 84     .     .     8B    8a         84     .     .     8B    8a
 85   - .     .     [4c   8a         85   + .     8D]   [4c   8a
 86   - .     24D]  .     .
 87     .     8E    .     8a         86     .     8E    .     8a
 88     =9    =9    =9    =9         87     =9    =9    =9    =9
 89     2GG   2F    4c]   8a         88     2GG   2F    4c]   8a
@@ -122,9 +121,8 @@
122     .     .     8g    16ff      121     .     .     8g    16ff
123     .     .     .     16ee      122     .     .     .     16ee
124     4A    4e    8r    8dd       123     4A    4e    8r    8dd
125   - .     .     8c    [12..cc   124   + .     .     8c    [8cc
126   - 4G    8g    8e    .         125   + 4G    8g    8e    32cc]
127   - .     .     .     96cc]
128     .     .     .     32dd      126     .     .     .     32dd
129     .     .     .     32ee      127     .     .     .     32ee
130     .     .     .     32ff      128     .     .     .     32ff
@@ -157,10 +155,10 @@
```

Figure 2.2: Common note quantization errors in humdrum-data repository that required manual fix.

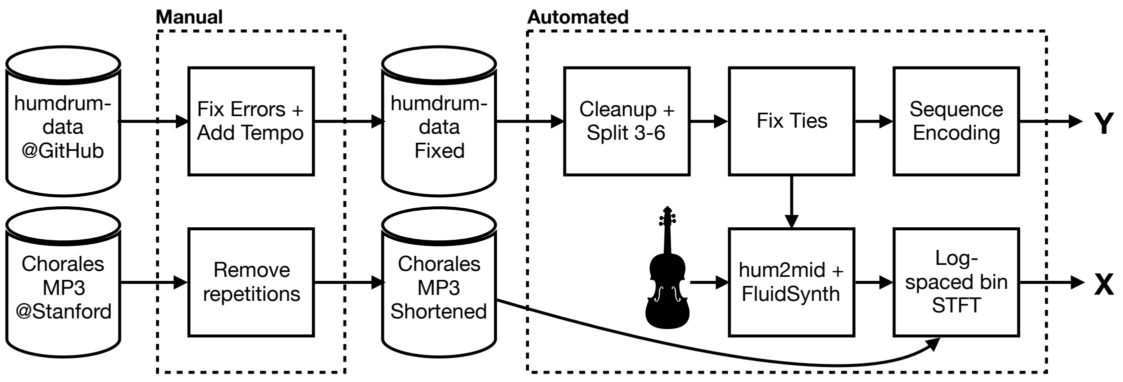


Figure 2.3: Training data acquisition pipeline for Humdrum data.

- *STFT*. The magnitude spectrogram obtained from the Short Time Fourier Transform.

- *LogSTFT*. A log-spaced filter bank was applied to the STFT magnitude spectrogram, with the purpose of having a fixed number of bins per octave.

- *CQT*. Constant-Q transform [15] spectrogram, where window sizes in the time domain vary in order to have a constant resolution for all frequency bins at the expense of inaccurate time resolution for lower frequencies.

2.3.2 Output

- *CTC-friendly.* This is a semantic music notation we specifically created to ease CTC decoding. It only supports monophonic music.

- *PAE-based.* This notation is based on the PAE [3] format, adding spaces to artificially create words to match the other output representations. We also had to ensure that changes of octave and duration applied to notes always appear in the same order and right before each note, to avoid data ambiguities. PAE can only represent one staff at a time, so it is not suitable for most polyphonic notation requirements.

- *Kern-based.* This notation is based Humdrum ***kern* representation [6], which supports the full spectrum of modern music scores, including multiple staffs, ornamentation, tempo markings, and dynamics.

2.4 Monophonic Audio-to-Score

We rigorously validated our A2S framework with more than 300,000 monophonic audio excerpts extracted from the RISM database. Although monophonic pitch estimation might be considered a solved problem given that it is not affected by frequency overlaps, converting predicted pitches into a valid score still presents some challenges that are worth addressing.

One of these challenges is note quantization, which consists of identifying the length of notes (e.g., 𝅗𝅥 , ♩ , ♪ , etc.) based on the audio information, and in relation to the rest of the predicted notes. The real duration of canonical note lengths varies within a reasonable range from piece to piece, and even between different interpretations of the same piece. This makes note quantization a much harder task than it may seem, requiring music language knowledge to properly assign canonical note lengths to predicted notes.

Another important challenge is estimating the correct metric information of the score, such as the time signature (e.g., $\begin{smallmatrix}4\\4\end{smallmatrix}$, $\begin{smallmatrix}3\\4\end{smallmatrix}$, $\begin{smallmatrix}6\\8\end{smallmatrix}$, etc.), along with the exact placement of barline symbols and rests (e.g., 𝄼 , 𝄽 , 𝄾 , etc.). This metric information is necessary to compose measures of equal time length out of the predicted notes, which is a fundamental part of the score as it determines the beat of the musical piece.

Additionally, our A2S monophonic framework also predicted the clef and the key signature, which aim at producing a more readable music score, but have no effect on its musical outcome.

2.5 Polyphonic Audio-to-Score

We also evaluated our A2S framework on polyphonic music, where the challenges regarding multi-pitch estimation are present, together with the specific challenges of the A2S task described in the previous section. On top of them, there is one extra challenge of positioning predicted notes in the correct staff by properly recognizing musical voicing.

We collected 4-voice music from Haydn, Mozart and Beethoven string quartets, and trained our model with excerpts of them. Additionally, we also used audio files from the Bach chorales, synthesized using a high quality pipe organ sound font.[7] We extracted the scores in ***kern* format from the humdrum-data database, adding a total of more than 26 hours of polyphonic audio fragments. By using ***kern* format as the output of our model, we can potentially support any score that can be written in modern music notation.

One limitation imposed by the CTC function, is that the number of output characters must be less or equal than the number of input frames. This creates a problem for polyphonic audio, as the number of output symbols increases by the number of voices, having the same number of input frames. One alternative to overcome this issue can be to artificially increase the number of input frames to enable more output characters. However, this hardly adds new information to our model, hence no better performance, at the cost of considerably more computation and memory requirements. The solution we devised involved changing the CRNN architecture to double the number of features extracted from the convolutional block while keeping the same number of frames in the input, thus meeting the maximum number of characters required for polyphonic music scores.

In order to reproduce the results of this polyphonic A2S work, source code and data are available in a public repository.[8]

[7]An example of such sound font can be found at `https://www.hauptwerk.com`

[8]`https://github.com/mangelroman/audio2score`

Part II

Published works

Chapter 3

An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts

Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 34–41, Paris, France, September 2018. ISMIR

AN END-TO-END FRAMEWORK FOR AUDIO-TO-SCORE MUSIC TRANSCRIPTION ON MONOPHONIC EXCERPTS

Miguel A. Román
U.I. for Computing Research
University of Alicante
Alicante, Spain
mroman@dlsi.ua.es

Antonio Pertusa
U.I. for Computing Research
University of Alicante
Alicante, Spain
pertusa@ua.es

Jorge Calvo-Zaragoza
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
jcalvo@prhlt.upv.es

ABSTRACT

In this work, we present an end-to-end framework for audio-to-score transcription. To the best of our knowledge, this is the first automatic music transcription approach which obtains directly a symbolic score from audio, instead of performing separate stages for piano-roll estimation (pitch detection and note tracking), meter detection or key estimation. The proposed method is based on a Convolutional Recurrent Neural Network architecture directly trained with pairs of spectrograms and their corresponding symbolic scores in Western notation. Unlike standard pitch estimation methods, the proposed architecture does not need the music symbols to be aligned with their audio frames thanks to a Connectionist Temporal Classification loss function. Training and evaluation were performed using a large dataset of short monophonic scores (incipits) from the RISM collection, that were synthesized to get the ground-truth data. Although there is still room for improvement, most musical symbols were correctly detected and the evaluation results validate the proposed approach. We believe that this end-to-end framework opens new avenues for automatic music transcription.

1. INTRODUCTION

Automatic Music Transcription (AMT) is a very relevant field within the Music Information Retrieval (MIR) community. This task can be defined as the automated process of converting an audio recording into any kind of musically-meaningful structured format. The usefulness of this process is very broad, especially for MIR algorithms such as content-based music search, symbolic music similarity, or symbolic musicological analysis.

However, this is a challenging task and state-of-the-art methods currently obtain a performance significantly below a human expert. In order to obtain a complete score from a waveform, it is necessary to perform pitch detection, note onset/offset detection, loudness estimation and quantization, instrument recognition, extraction of rhythmic information, and time quantization [2].

Most music transcription systems focus on two of these stages: pitch detection, where pitches at each time frame of the audio are estimated, and note tracking [32], where the estimations of the previous step are discretized into sequences of 3-tuples (onset, offset, pitch). The output in this case is a piano-roll, that is, a two-dimensional representation of notes across time [2]. Multiple pitch estimation techniques include spectrogram factorization methods [1, 3, 28] and discriminative approaches, which perform frame-by-frame pitch estimation using statistical models [10], signal processing methods [23, 35], or machine learning techniques [4] including deep neural networks [17, 27, 30]. Some works also integrate musical language models into the pitch estimation process to resolve output ambiguities [27, 34].

Supervised learning approaches for piano-roll estimation require the ground truth to be aligned for training. Matching pitches frame by frame with their corresponding waveform samples is a time-consuming task and, although there are some efforts in this direction with datasets such as MAPS [10], RWC [11] or MusicNet [29], currently there are no very large AMT corpora. Beyond the difficulty of performing an accurate annotation, frame-by-frame estimation has some additional issues to be taken into account. For example, when a whole note is played using a plucked string instrument such as a guitar, the quick decay of its harmonic amplitudes produces frames with a very low intensity at the end of the note, causing ambiguities when labeling the offset frames.

In addition, as pointed out in [2], AMT algorithms are usually developed independently to carry out individual tasks such as multiple pitch detection, beat tracking and instrument recognition. Some existing AMT methods, such as the ones proposed in [19–21], also include rhythm estimation and time quantization. Still, the challenge remains to combine the outputs of the individual tasks to perform joint estimation of all parameters, in order to avoid the cascading of errors when algorithms are combined sequentially.

In this work we intend to open a new framework to address the AMT task. Our proposal is to consider end-to-end machine learning strategies, with which this task can be carried out holistically. In other words, we aim at using a waveform as input, and directly obtaining a music score at the output taking into account all its components (pitches, note durations, time signature, key signature, etc.) jointly.

The task of directly estimating a symbolic score from audio is certainly different from that of estimating a piano-roll. While piano-roll estimation aims to extract what has been played from the audio as exact as possible, in the score transcription task the goal is to obtain a symbolic representation from what the musician read, which includes abstracting away some information such as loudness.

For this, we address score estimation using Deep Neural Networks. We specifically consider the use of a Convolutional Recurrent Neural Network, which is responsible of both processing the input spectrogram to extract meaningful features and predict an output sequence that represents the music contained in a given audio recording. Thanks to the Connectionist Temporal Classification (CTC) loss function, this kind of networks can be trained in terms of pairs (input, output), without needing of dividing the process into smaller stages or providing framewise annotations. The idea is that the prediction is forced to be encoded in terms of actual music-notation elements.

It is important to emphasize that the objective of this work is not to outperform the accuracy of previous approaches, but to propose a framework with which to address the AMT task. In order to demonstrate the feasibility of this formulation, our experiments are restricted to a constrained scenario, using audio recordings from monophonic scores that were synthesized using a piano. We are aware that the main challenge in AMT is to deal with polyphonic real music. In a future work we plan to extend the proposed approach to detect polyphonic scores, although its effectiveness with sound mixtures is yet to be studied.

The evaluation results in this constrained scenario validates the proposed framework and show that the the proposed approach obtains reliable results, correctly detecting most musical symbols.

The rest of the paper is organized as follows: the corpus used for evaluation is described in Section 2; the holistic neural framework proposed for the AMT task is described in Section 3; the series of experiments carried out are detailed in Section 4; and finally, the conclusions of the current work are summarized in Section 5, pointing out some interesting avenues for future work as well.

2. DATASET

In order to get the ground truth for our framework, we used the RISM [1] collection [26], which currently contains more than one million incipits (short monophonic music excerpts). This corpus is very useful for music retrieval tasks because of its size and the fact that it contains real music written by human composers [31]. Spectrograms from synthesized incipits are the inputs to our method, and their corresponding symbolic scores are the outputs. The scheme of the proposed method can be seen in Figure 1.

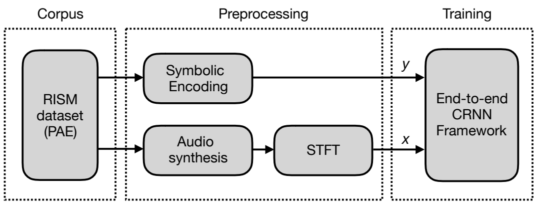


Figure 1: Data acquisition for training. RISM incipits are converted into our music notation format and magnitude spectrograms (Short-Time Fourier Transform, STFT) are also computed from synthesized versions of the incipits. The inputs of the proposed framework (x) are the symbolic data and the outputs are the spectrograms (y). Frame-by-frame alignment is not necessary.

2.1 Preprocessing

RISM incipits are formatted in Plaine & Easie Code (PAE). We randomly selected a subset of 71,400 incipits in Western notation and converted them into the music notation format that can be seen in Table 1, where each symbol is encoded using a single character. This notation is oriented to represent the music as a language, similarly to what a speech recognition system does. Following this analogy, we consider a music note as a *word* (for example, C♯4♩) containing several *characters* from an alphabet set Σ that can be seen in Table 1, and which is separated to other words by blank spaces. Rests are represented in the same way, with a word consisting of the rest symbol and its duration. In addition to notes and rests, the alphabet set includes clefs, key and time signatures, measure bars and note ties. Every musical symbol in Table 1 is encoded for our framework using a single element (one character).

In order to get the audio files, the RISM PAE incipits were converted into Music Encoding Initiative (MEI) format, and then translated again into MIDI using Meico [2], which unlike Verovio [3] takes into account the key signature.

The synthesis from MIDI files was performed using timidity with the piano program of the default soundfont, obtaining monoaural audio files at 16kHz. Then, magnitude spectrograms were calculated using a 64ms (1024 samples) Hamming window with a 16ms hop (256 samples). All incipits were synthesized using random tempo values in the range [96-144] bpm in order to make the network work with different speeds.

3. FRAMEWORK

We describe in this section the neural model that allows us to face the AMT task directly from an audio signal to a sequence of meaningful symbols.

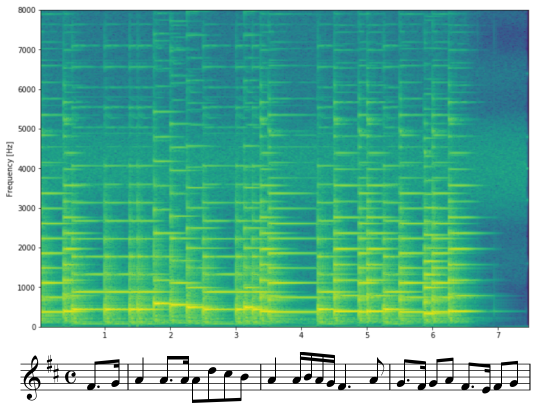
[1] The complete set of RISM incipits can be downloaded from `https://opac.rism.info/index.php?id=8&L=1&id=8`

[2] `https://github.com/cemfi/meico`

[3] `http://www.verovio.org/index.xhtml`

Class	Symbol	Count	Histogram
Global	Blank	1,526,051	
Clef	G2	39,337	
	F4	4,414	
	C1	22,468	
	C3	1,981	
	C4	3,200	
Key	D♭M	112	
	A♭M	1,065	
	E♭M	6,815	
	B♭M	8,950	
	FM	11,599	
	CM	15,488	
	GM	10,309	
	DM	10,861	
	AM	4,933	
	EM	1,216	
	BM	52	
Pitch	A	87,323	
	B	88,004	
	C	95,190	
	D	100,014	
	E	80,780	
	F	75,579	
	G	84,953	
	♭	70,557	
	♯	55,471	
	Rest	89,635	
Octave	2	2,937	
	3	44,170	
	4	274,590	
	5	284,081	
	6	6,065	
Duration	𝅝	8,686	
	𝅗𝅥	52,541	
	♩	172,933	
	♪	226,395	
	𝅘𝅥𝅯	72,124	
	𝅘𝅥𝅰	6,711	
	.	72,453	
	Tie	10,393	
Time	4/4	27,855	
	2/2	13,848	
	3/4	11,595	
	2/4	7,569	
	6/8	4,950	
	3/8	2,916	
	3/2	1,199	
	12/8	592	
	6/4	417	
	4/2	305	
	9/8	154	
	Barline	245,239	

Table 1: Symbols of the alphabet Σ. Notes are encoded using words of three to five symbols (for example, C♯4♩.).



$ q c F♯4♪. G4𝅘𝅥𝅯 | A4♩ a4♪. A4𝅘𝅥𝅯 A4♪ D5♪ C♯5♪ B4♪ | A4♩ A4𝅘𝅥𝅯 B4𝅘𝅥𝅯 A4𝅘𝅥𝅯 G4𝅘𝅥𝅯 F♯4♩. A4♪ | G4♪. F♯4𝅘𝅥𝅯 G4♪ A4♪ F♯4♪. E4𝅘𝅥𝅯 F♯4♪ G4♪

Figure 2: Example of a magnitude spectrogram (x) synthesized from a RISM incipit (center). The symbolic encoding representation used for the CRNN (y) is shown below, where the character '$' is the G2 clef, 'q' is the key signature DM, the symbol 'c' is used to encode 4/4 and ' | ' represents the barline. Similarly to speech recognition, words are separated by blank spaces.

Formally, let $\mathcal{X} = \{(x_1, y_1), (x_2, y_2), ...\}$ be our end-to-end application domain, where x_i is an audio recording represented by its magnitude spectrogram, and y_i denotes its corresponding ground-truth sequence from a fixed alphabet set Σ.

The problem of AMT can be reformulated as retrieving the most likely sequence of symbols $\hat{y}$ given an input spectrogram x. That is:

$$\hat{y} = \arg\max_{y \in \Sigma^*} P(y|x) \tag{1}$$

We formulate this statistical framework by means of Recurrent Neural Networks (RNN), as they allow handling sequences [12]. Ultimately, therefore, the RNN will be responsible of producing the sequence of musical symbols that fulfills Eq. 1. Nevertheless, on top of it, we add a Convolutional Neural Network (CNN), which learns how to process the input signal to represent it in a meaningful way for the task at issue [36]. Since both types of networks consist of feed-forward operations, the training stage can be carried out jointly by simply connecting the output of the last layer of the CNN with the input of the first layer of the RNN, which leads to a Convolutional Recurrent Neural Network (CRNN). A similar topology was previously applied to drum transcription in [33], although not in an end-to-end fashion.

Our work is conducted over a supervised learning scenario. Therefore, it is assumed that we can make use of a set $\mathcal{T} \subset \mathcal{X}$ with which to train the model. Initially, the traditional training mechanism for a CRNN needs to be provided with the expected output for each frame of the input. As introduced above, for each recording the training set only contains its corresponding sequence of expected symbols, without any kind of explicit information about their location within the input signal. This scenario can be solved by means of the so-called Connectionist Temporal Classification (CTC) loss function [13].

Given an input x, CTC provides a means to optimize the CRNN parameters in order to directly output its correct sequence y. In other works, CTC directly optimizes $P(y|x)$. Since the ground-truth is not aligned at the frame level, that is, it is unknown the alignment between the frames of the recurrent part and the output symbols, CTC integrates over all possible alignments. It only considers monotonic alignments (left-to-right constraint), which is a valid assumption in our task.

Although optimizing the aforementioned probability is computationally expensive, CTC performs a local optimization using an Expectation-Maximization algorithm similar to that used for training Hidden Markov Models [24]. However, given that CTC integrates over all possible alignments, its main limitation is that the cost of the optimization procedure grows rapidly with the length of the sequences.

Note that CTC is used only for training. At the inference stage, the CRNN still predicts a symbol for each frame of the recurrent block. To indicate a separation between symbols, or to handle those frames in which there is no symbol, CTC considers an additional symbol in the alphabet that indicates this situation (*blank* symbol).

3.1 Implementation details

Finding the best instantiation of a CRNN for the case of AMT is out of the scope of this work, but we are inspired by the *Deep Speech 2* [8] topology, which was especially designed for the task of Automatic Speech Recognition (ASR). Although ASR and AMT are different tasks they are related, and so the use of this architecture allows us to obtain valuable results without having to make an exhaustive search of the best neural topology.

Nonetheless, we made small modifications to the original architecture in order to adjust its behavior to AMT. The specification of our neural topology is detailed in Table 2. It consists of 2 convolutional layers and 3 recurrent layers. Convolutional layers are composed of convolutional filters followed by Batch Normalization [16], and the non-linear hard hyperbolic tangent (HardTanh) activation function [14]. Furthermore, bi-directional recurrent layers are configured as Gated Recurrent Units (GRU) [7], with Batch Normalization as well. On top of the last recurrent output, a fully-connected layer is placed with as many neurons as symbols of the vocabulary (plus 1, because of the *blank* symbol). The use of the *softmax* activation allows us to interpret the output of this last layer as a posterior probability over the vocabulary [6].

The training stage is carried out by providing pairs of spectrograms with their corresponding unaligned sequence of musical symbols. The optimization procedure follows stochastic gradient descent (SGD) [5] with Nesterov momentum of 0.9, gradient L2 Norm clipping of 400, and a mini-batch size of 20 samples, which modifies the network weights to minimize the CTC loss function through back-propagation. The learning rate was initially set to 0.0003, but it was annealed by a factor of 1.1 after each epoch to favor convergence. The model was trained during 20 epochs, fixing the weights according to the best result over the validation set.

Input($1024 \times T$)
Convolutional block
Conv($32, 41 \times 11, 2 \times 2$), BatchNorm(), HardTanh()
Conv($32, 21 \times 11, 2 \times 1$), BatchNorm(), HardTanh()
Recurrent block
B-GRU(1024), BatchNorm()
B-GRU(1024), BatchNorm()
B-GRU(1024), BatchNorm()
Dense($|\Sigma| + 1$), Softmax()

Table 2: Instantiation of the CRNN used in this work for audio-to-score AMT, consisting of 2 convolutional layers and 3 recurrent layers. Notation: Input($h \times w$) means an input spectrogram of height h and width w; Conv($n, k_h \times k_w, s_h \times s_w$) denotes a convolution operator of n filters, kernel size of $k_h \times k_w$, and stride of $s_h \times s_w$; BatchNorm() denotes a batch normalization procedure; HardTanh() represents the *hard hyperbolic tangent* activation; B-GRU(n) means a bi-directional Gated Recurrent Units of n neurons; $Dense(n)$ denotes a fully-connected layer of n neurons; and $Softmax()$ represents the *softmax* activation function. Σ denotes the character-wise alphabet considered.

Once the CRNN is trained with the previous procedure, it can be used to output a discrete symbol sequence from a given spectrogram. The model yields character-level predictions in each frame. In order to provide an actual symbol sequence, it is necessary to both collapse repeating characters and discarding *blank* characters. Since there could be several frame-level sequences that result in the same sequence of musical symbols, the final decoding is conducted by a beam search procedure [37], with a beam width set to 10.

4. EXPERIMENTS

4.1 Setup

The proposed framework is evaluated using the corpus described in Section 2.1.

Experiments are performed dividing the available data into three independent partitions: $49,980$ samples (118.03 hours) for training, $10,710$ samples (25.34 hours) for validation, and $10,710$ samples (25.36 hours) for the test set, which is used to evaluate the actual performance.

Given the differences with existing AMT approaches, our results are not directly comparable with any previous work. Likewise, there are no standard evaluation metrics with which to evaluate this framework.

Here, we propose a series of metrics especially considered for evaluating the presented approach. In particular, we are inspired by other tasks, like ASR or Optical Character Recognition (OCR), that are also formulated expecting a sequence of symbols as output. Analogously to these tasks, we also assume that the output consists of individual *characters* (pitches, durations, alterations, ...) that build complete *words* (such as notes). Therefore, the performance can be evaluated in terms of Character Error Rate (CER) and Word Error Rate (WER). These metrics are defined as the number of elementary editing operations (insertion, deletion, or substitution) to convert the hypotheses of the system into the ground-truth sequences, at the character and word level, respectively. They compute this cost in a normalized way according to the length of the ground-truth sequences. Even assuming that these metrics are not optimal for the task of AMT, we hope that they allow us to validate the approach and draw reasonable conclusions from our experimental results.

In order to get some baseline results that can be compared to other works, we also applied the evaluation metric used in [19] for piano-roll alignment tasks. The total number of notes in the ground truth is denoted by N_{GT}, that of estimated notes by N_{est}. The number of notes with pitch errors is denoted by N_p, that of extra notes by N_e, and that of missing notes by N_m. The number of matched notes is defined as $N_{match} = N_{GT} - N_m = N_{est} - N_e$. Then we define the pitch error rate as $E_p = N_p/N_{GT}$, extra note rate as $E_e = N_e/N_{est}$, and missing note rate as $E_m = N_m/N_{GT}$. Onset/offsets errors are also reported in [19]. As we are dealing with note durations instead of onsets/offsets, we include an alternative error metric E_d which is calculated similarly to the pitch error E_p but using note duration errors, denoted by N_d. Thus, we define the duration error rate as $E_d = N_d/N_{GT}$.

4.2 Results

Figure 3 shows the evolution of the errors during the training process. As can be seen, the convergence is fast and the best results on the validation set are obtained at epoch 18, reporting a CER of 5.53 and a WER of 15.98. In the test set, a CER of 5.36 and a WER of 15.67 are obtained. These results are very similar to those from the validation set, thus proving that there is no over-fitting and the model generalizes well.

After an in-depth analysis of the test set transcriptions obtained, we observed that the majority of errors are due to wrong time signatures, barline locations, and clefs. This result was expected in our prior analysis, as even for a human it would be difficult to identify them based on the short audio excerpts we provide to our model (the average number of music measures of the audio excerpts is 4.4). Furthermore, there are some time signatures that contain the same number of notes per measure and therefore they require more musical context to identify them correctly (e.g. 4/4 and 2/2 time signatures), as shown in Figure 4. In other cases, one of these specific errors causes the appearance of many others, as seem to happen with the time signature in the example of Figure 5. In order to address these ambiguities, normalization techniques could be employed (for instance, changing all 2/2 by their equivalent notation in 4/4).

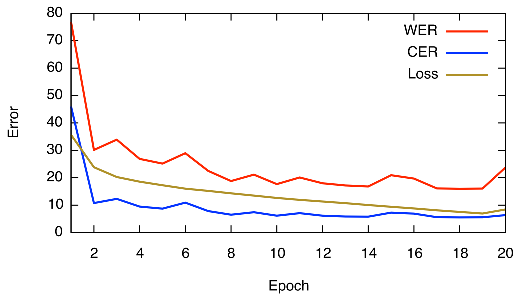
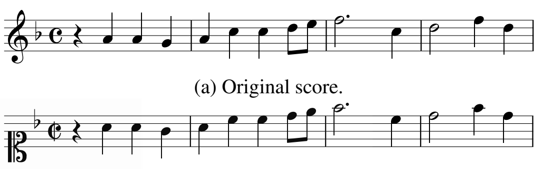


Figure 3: Evolution curves of the CTC loss, CER, and WER over the validation set with respect to the training epoch. The lowest WER (15.98) and CER (5.53) figures are obtained at epoch 18.



(a) Original score.

(b) Transcribed score.

Figure 4: Example of transcription performance. Note that the two mistakes made (clef and time signature) belong to music notation ambiguities.

In spite of all these difficulties, some samples are perfectly recognized, as the one depicted in Figure 6.

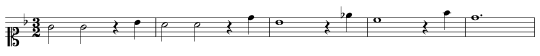
We provide the results of the evaluation metric proposed in [19] for estimated notes, and for estimated notes and rests combined (in this case, E_p does not change). As can be seen in Table 3, the error rates are quite low compared to [19], but this is due to the fact that our audio files are monophonic and synthesized. In addition, most transcription errors are due to wrong estimations of time signatures, subsequently yielding wrong barline locations as previously explained.

5. CONCLUSIONS

In this work, we propose a new formulation of AMT in the form of an audio-to-score task. In summary, the advantages of this formulation over piano-roll estimation are: 1) it is not required to have a frame-by-frame annotation aligned with the audio, therefore potentially more data

Table 3: Note pitch error rate (E_p), missing symbol rate (E_m), extra symbol rate (E_e) and symbol duration error rate (E_d) considering only notes and notes plus rests.

	E_p	E_m	E_e	E_d
Notes	0.99%	2.63%	1.81%	0.71%
Notes+Rests	0.99%	4.94%	2.51%	1.23%


(a) Original score.

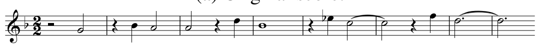

(b) Transcribed score.

Figure 5: Example of a transcription with several mistakes. Here, the unusual time signature 3/2 (wrongly detected) propagates the errors to the notes.

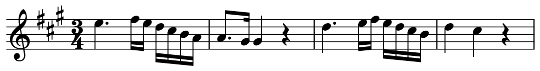


Figure 6: Example of a correctly transcribed score.

could be acquired for training; 2) the obtained outputs are musically meaningful; 3) the frame-by-frame annotation ambiguities are avoided, although on the other hand there are music notation ambiguities to deal with; 4) the task is addressed holistically instead of using a pipeline of individual processes, avoiding the cascading of errors when they are combined sequentially, and 5) musical models are implicitly inferred as it occurs with language models in speech recognition.

We validated the proposed framework using a CRNN with a CTC loss function trained on RISM incipits, correctly predicting around 84% of symbols for monophonic scores synthesized with a piano sound at different tempos. It is important to note that some symbols such as barlines, rests, ties, time signatures or key signatures were no explicitly present in spectrograms but they were correctly inferred from the context.

A qualitative analysis of the performance reported that many errors occurred because of music notation ambiguities. Although they decreased the WER and CER figures, ”wrong” outputs are musically correct and equivalent to the ground-truth scores in most cases.

As a future work, we are planning first to extend it for polyphonic sources, and also to perform instrument recognition. In order to deal with polyphony, a chord could be considered as a “word” containing “syllabus” (the individual notes), for example: `C4𝅗𝅥E4𝅗𝅥G4𝅗𝅥`. An additional symbol could be added to indicate the instrument (for example, `PC4♩` could represent a quarter note of pitch C4 played on Piano).

As pointed out in [18], previous experiments on deep neural networks dealing with framewise multiple pitch detection showed that unseen combinations are hard to detect. A partial solution to this problem might involve a modification of the loss function for the network to disentangle individual notes explicitly and learn to decompose a (nonlinear) mixture of signals into its constituent parts. We believe that, unlike what happens in this framewise detection, CTC loss may be able to break the observed glass-ceiling, given that ASR methods using this architecture are capable of generalizing to detect unseen words from its constituent (character) elements. Nonetheless, additional experiments on AMT should confirm this hypothesis.

Synthesized scores were used to perform the experiments, although ideally real data should be evaluated. For this, we are planning to use datasets such as Lakh [25], which contains audio files with their corresponding MIDIs. Given the computational cost of CTC, the proposed framework needs to use short segments. Therefore, it is necessary to have aligned barlines to split both the audio and the corresponding score ground truth into smaller pieces. This could be done using a score following method [9, 22]. This preprocessing could introduce some errors due to wrong alignments, but there is a more suitable alternative: to train the CRNN using full scores along with their complete real audio files, which is the ultimate goal of the proposed framework. This is possible and could be done by considering the recently proposed *online* CTC [15] function, which efficiently adapts to any sequence length.

Another obvious future work is to find a more adequate network architecture and evaluate alternative hyperparameters to increase the accuracy. CNN and RNN topologies evaluated in previous AMT works [17, 27] should be investigated for this task.

In conclusion, in this work we show that it is feasible to perform end-to-end transcription from monophonic audio files to scores. We are fully aware that experiments were made in a very controlled and simplified environment and there is still much work to do in order to perform a complete transcription. But we believe that the proposed framework opens a new exciting research area given the huge amount of data that could potentially be used for training, and its practical utility for musicians who could obtain directly a score from audio.

6. ACKNOWLEDGEMENT

This work was supported by the University of Alicante through grant GRE-16-04 and its University Institute for Computing Research (IUII), and by the Spanish Ministerio de Economía, Industria y Competitividad through HispaMus project (TIN2017-86576-R) and Juan de la Cierva - Formación grant (Ref. FJCI-2016-27873).

7. REFERENCES

[1] E. Benetos and S. Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, 2012.

[2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic Music Transcription: Challenges and Future Directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Conference on Music Information Retrieval*, pages 701–707, 2015.

[4] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing, pages 121–124, 2012.

[5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] H. Bourlard and C. Wellekens. Links Between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1167–1178, 1990.

[7] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantic sand Structure in Statistical Translation*, pages 103–111, 2014.

[8] D. Amodei et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *33nd International Conference on Machine Learning*, pages 173–182, 2016.

[9] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):38–43, 2006.

[10] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[11] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *3rd International Conference on Music Information Retrieval*, pages 287–288, 2002.

[12] A. Graves. *Supervised Sequence Labelling with recurrent neural networks*. PhD thesis, Technical University Munich, 2008.

[13] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *23rd International Conference on Machine Learning*, International Conference on Machine Learning, pages 369–376. ACM, 2006.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems 29*, pages 4107–4115, 2016.

[15] K. Hwang and W. Sung. Online Sequence Training of Recurrent Neural Networks with Connectionist Temporal Classification. *CoRR*, abs/1511.06841, 2015.

[16] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July*, pages 448–456, 2015.

[17] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the Potential of Simple Framewise Approaches to Piano Transcription. In *17th International Conference on Music Information Retrieval*, 2016.

[18] R. Kelz and G. Widmer. An experimental analysis of the entanglement problem in neural-network-based music transcription systems. *arXiv preprint arXiv:1702.00025*, 2017.

[19] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2018.

[20] E. Nakamura, K. Yoshii, and S. Dixon. Note Value Recognition for Piano Transcription Using Markov Random Fields. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25:1542–1554, 2017.

[21] E. Nakamura, K. Yoshii, and S. Sagayama. Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25:794–806, 2017.

[22] N. Orio, S. Lemouton, and D. Schwarz. Score following: State of the art and new developments. In *Proc. of the 2003 Conference on New interfaces for Musical Expression*, pages 36–41. National University of Singapore, 2003.

[23] A. Pertusa and J. M. Iñesta. Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*, 2012(1):27, Feb 2012.

[24] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice hall, 1993.

[25] C. Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[26] RISM. Répertoire International des Sources Musicales, 2017.

[27] S. Sigtia, E. Benetos, and S. Dixon. An End-to-end Neural Network for polyphonic piano music transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 24(5):927–939, 2016.

[28] P. Smaragdis and J. C. Brown. Non-negative Matrix Factorization for Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[29] J. Thickstun, Z. Harchaoui, and S. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.

[30] J. Thickstun, Z. Harchaoui, D. Foster, and S. M. Kakade. Invariances and data augmentation for supervised music transcription. *arXiv preprint arXiv:1711.04845*, 2017.

[31] R. Typke, M. Den Hoed, J. De Nooijer, F. Wiering, and R. C. Veltkamp. A ground truth for half a million musical incipits. *Journal of Digital Information Management*, pages 34–39, 2005.

[32] J. J. Valero-Mas, E. Benetos, and J. M. Iñesta. A supervised classification approach for note tracking in polyphonic piano transcription. *Journal of New Music Research*, pages 1–15, 2018.

[33] R. Vogl, M. Dorfer, G. Widmer, and P. Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *18th International Conference on Music Information Retrieval*, pages 150–157, 2017.

[34] Q. Wang, R. Zhou, and Y. Yan. Polyphonic Piano Transcription with a Note-Based Music Language Model. *Applied Sciences*, 8(3), 2018.

[35] C. Yeh, A. Robel, and X. Rodet. Multiple fundamental frequency estimation of polyphonic music signals. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3. IEEE, 2005.

[36] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *13th European Conference on Computer Vision*, pages 818–833, September 2014.

[37] T. Zenkel, R. Sanabria, F. Metze, J. Niehues, M. Sperber, S. Stüker, and A. Waibel. Comparison of decoding strategies for CTC acoustic models. In *18th Annual Conference of the International Speech Communication Association*, pages 513–517, 2017.

Chapter 4

A Holistic Approach to Polyphonic Music Transcription with Neural Networks

Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A Holistic Approach to Polyphonic Music Transcription with Neural Networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 731–737, Delft, The Netherlands, November 2019. ISMIR

A HOLISTIC APPROACH TO POLYPHONIC MUSIC TRANSCRIPTION WITH NEURAL NETWORKS

Miguel A. Román
U.I. for Computing Research
University of Alicante, Spain
`mroman@dlsi.ua.es`

Antonio Pertusa
U.I. for Computing Research
University of Alicante, Spain
`pertusa@dlsi.ua.es`

Jorge Calvo-Zaragoza
U.I. for Computing Research
University of Alicante, Spain
`jcalvo@dlsi.ua.es`

ABSTRACT

We present a framework based on neural networks to extract music scores directly from polyphonic audio in an end-to-end fashion. Most previous Automatic Music Transcription (AMT) methods seek a piano-roll representation of the pitches, that can be further transformed into a score by incorporating tempo estimation, beat tracking, key estimation or rhythm quantization. Unlike these methods, our approach generates music notation directly from the input audio in a single stage. For this, we use a Convolutional Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) loss function which does not require annotated alignments of audio frames with the score rhythmic information. We trained our model using as input Haydn, Mozart, and Beethoven string quartets and Bach chorales synthesized with different tempos and expressive performances. The output is a textual representation of four-voice music scores based on **kern format. Although the proposed approach is evaluated in a simplified scenario, results show that this model can learn to transcribe scores directly from audio signals, opening a promising avenue towards complete AMT.

1. INTRODUCTION

Automatic music transcription (AMT) aims to convert acoustic music signals into any sort of music notation. Most of the music we listen today is polyphonic, where simultaneous sound events produced by different audio sources (i.e., instruments) are combined in a single acoustic waveform. This aggregation process entails loss of information, making the transcription task very challenging even for trained musicians. Moreover, the different sound events are highly correlated in time and frequency due to the rhythmic and harmonic patterns usually found in music, which complicates sound separation even further as we cannot rely on the statistical independence of the source signals. Therefore, in order to produce a proper music score from an audio signal, multiple complex sub-tasks must be involved such as multi-pitch estimation, note onset/offset detection, source separation, as well as other musical context information retrieval tasks like metering and tonality estimation.

As pointed out by [2], there are many approaches to tackle AMT, yet most works focus on solving only one intermediate goal of the whole problem. Frame-level transcription, also known as multi-pitch estimation, aims to detect which fundamental frequencies are present at each time step of the input signal. Note-level transcription goes a step further by estimating the notes characterized by their pitch and clock-time duration (onset and offset times), producing a piano-roll representation of the music. Stream-level transcription extends the note-level approach by associating each note with its originating instrument based on its timbre. Lastly, the notation-level transcription is the final goal of AMT, aiming to produce a music score with enough information to interpret the original recording.

In this work, we denote the notation-level transcription as Audio-to-Score (A2S) task, where the audio signal is processed to be converted into a symbolic music score. Even with a perfect transcription, the output of any A2S system cannot faithfully represent the music that was originally played. It must be considered that musical audio signals are often expressive performances, rather than simple mechanical translations of notes read from a staff. A particular score can be performed by a musician in many different ways, and similarly there are several ways to represent the same musical excerpt with standard music notation (e.g., a dotted half note is "the same" as a half note tied to a quarter note). Music scores can only be seen as guides to aid musicians, highly correlating but never fully explaining musical experience. This makes A2S a rather ill-defined problem without unique solutions.

Despite the above, our work aims to demonstrate that the A2S task can be performed in a single step. To this end, we make use of a deep neural network that is trained in an end-to-end fashion to produce a sequence of musical symbols that describes a feasible polyphonic score out of the input audio. Our experiments are conducted using Haydn, Mozart, and Beethoven string quartets and Bach chorales synthesized with different tempos and expressive performances. [1] Although the analysis of the current performance requires a deeper reasoning regarding evaluation

[1] The source code and data are available at `https://github.com/mangelroman/audio2score`.

metrics, we provide some results that account for the good performance of the proposed model and allow us to be optimistic about this line of research.

1.1 Related Work

There are recent AMT approaches using deep neural networks for the multi-pitch detection task [8, 11, 12]. For this, Short-Term Fourier Transform (STFT), log-frequency STFT or Mel spectrograms are usually fed to Convolutional Neural Networks (CNN) to extract piano-roll representations as output. Other works focus on producing music scores from unquantized MIDI representation [4].

One of the few methods aiming to extract a complete score directly from audio is that of [14]. In this work, a multi-pitch detection method with note tracking is used to get a piano-roll representation that is further converted into a quantized MIDI file by using a rhythm quantization method [15]. Afterwards, a score typesetting software such as MuseScore can be used to get a MusicXML file from the MIDI output.

To the best of our knowledge, there are only two approaches that perform A2S in a single stage, directly converting the input audio into any music notation format. This has the advantage that a wrong detection in a given stage (such as the multi-pitch detection) is not propagated through the next processing stages, avoiding error cascading. The only works addressing notation-level AMT in an end-to-end manner are those of [3] and [17]. Both works follow a supervised learning approach with deep neural networks to solve the AMT task in one step. Although they bring promising results, the proposed models include several limitations (e.g. monophonic audio in [17] and fixed input length in [3]) that cannot be disregarded when addressing the notation-level AMT problem as a whole.

In [3], authors show how a Convolutional-Recurrent Neural Network architecture (CRNN) [19] can learn all the basic tasks involved in notation-level AMT, but it is only a very limited proof of concept that cannot address most of the possible scores. In the second of these works [17], the AMT problem was addressed as an Automatic Speech Recognition (ASR) problem. By using monophonic audio as input and a sequence of symbols (analogously to the written language characters) as output, several methods that were originally developed for ASR can be used. In particular, [17] adopted an architecture inspired in DeepSpeech2 [1], which learns to map audio frames to a sequence of characters without any alignment.

Training with unaligned data, i.e. without needing the input audio frames to be aligned with the music symbols, is a clear advantage as much more data can be gathered without going through the tedious task of manually annotating the location of the output symbols in their corresponding input audio frames. Nevertheless, monophonic audio transcription does not exhibit the essential challenge coming from simultaneous sound events. The present work goes one step beyond by showing that a similar formulation can also be reliable for polyphonic music.

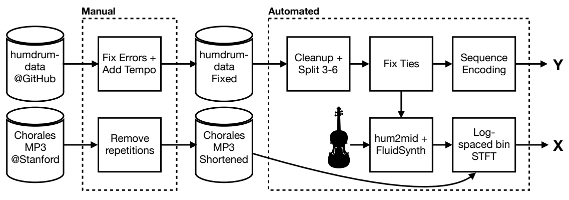


Figure 1: Data acquisition pipeline showing the manual and automated steps required to build the ground truth.

2. DATA

Our notation-level AMT approach, namely A2S, seeks to estimate which music score, modeled as a structure containing symbols from a fixed alphabet of music notation, would likely define that audio.

Let $\mathcal{X}$ be the domain of audio files and Σ the alphabet of music score symbols. The aim of our A2S is to compute a function that maps any audio file into a sequence of symbols, i.e., a function $f : \mathcal{X} \rightarrow \Sigma^*$.

2.1 Input Representation

The input representation of our model is the spectral information of the raw audio waveform over time, based on the STFT with log-spaced bins and log-scaled magnitude. In this type of spectrogram, frequency bins are aligned with equal-tempered music scales using 440Hz as the reference for A4 pitch. The sampling rate of the input audio files was 22,050Hz, and STFT was calculated with a Hamming window with size 92.88ms (2048 samples) and a hop of 23.22ms (512 samples). Only frequencies between pitches C2 and C7 were considered, extracting 48 bins per octave.

2.2 Output Representation

The output music notation of our model is a single sequence of symbols that can be used to render a multi-part western music score. These symbols represent both notes and rests with their corresponding duration, barlines, ties between notes, and fermatas. It is important to remark that in the context of the A2S task, notes are not the same as pitches. For example, pitch 349.23Hz can be represented as F4, E♯4 or G♭♭4 depending on the key signature.

We are not including clefs in our output representation, as they are only intended to aid in the score visualization and do not carry any musical information we can extract from the audio. For the sake of simplicity, we are also not including time signatures in the output sequences assuming they can be inferred from the predicted barlines for the type of scores in our training set. By the same rationale, key signatures are neither included assuming they can also be inferred from the predicted notes. Moreover, since most of our samples are made of fragments of much longer scores, they may not carry enough information to predict the correct key signature, therefore misleading the training process. Other music notation symbols such as slurs, grace notes, ornaments, and articulations marks are also left out of the scope of this work.

```
!!!COM: Mozart, Wolfgang Amadeus
!!!OTL: String Quartet No. 17 in B-flat Major
!!!OMD: Allegro vivace assai
!!!OMV: 1
**kern     **kern     **kern     **kern
*I:cello   *I:viola   *I:violn   *I:violn
*k[b-e-]   *k[b-e-]   *k[b-e-]   *k[b-e-]
*B-:       *B-:       *B-:       *B-:
*clefC     *clefF     *clefG     *clefG
*M6/8      *M6/8      *M6/8      *M6/8
8r         8r         8dd        8ff
=1         =1         =1         =1
4B-        4b-        8dd        8ff
.          .          8b-        8dd
8r         8f         8b-        8dd
4BB-       8f         8b-        8dd
.          8d         8f         8b-
8r         8d         8f         8b-
=2         =2         =2         =2
4D         4d         4f         4b-
8BB-       8f         8b-        8dd
4F         4.f        4a         4cc
8D         .          8b-        8dd
=3         =3         =3         =3
4C         8.g        4cc        8.ee-
.          16f        .          16dd
8r         8e-        8r         8cc
8E-        4e-        4.c        8g
8C         .          .          8b-
8F         8c         .          8a
=4         =4         =4         =4
4D         8B-        4f         4b-
.          8d         .          .
8BB-       8f         8b-        8dd
4F         4f         4a         4cc
8r         8r         8dd        8ff
=5         =5         =5         =5
4B-        4b-        16dd       16ff
.          .          16ee-      16gg
.          .          8dd        8ff
8r         8f         8b-        8dd
4BB-       16f        16b-       16dd
.          16g        16cc       16ee-
.          8f         8b-        8dd
8r         8d         8f         8b-
=6         =6         =6         =6
4D         4d         4f         4b-
8BB-       8f         8b-        8dd
4F         4.f        4a         4cc
8D         .          8b-        8dd
=7         =7         =7         =7
4C         8.g        4cc        8.ee-
.          16f        .          16dd
8r         8e-        8r         8cc
8E-        4.e-       4.c        8g
8C         .          .          8b-
8F         .          .          8a
=8         =8         =8         =8
```

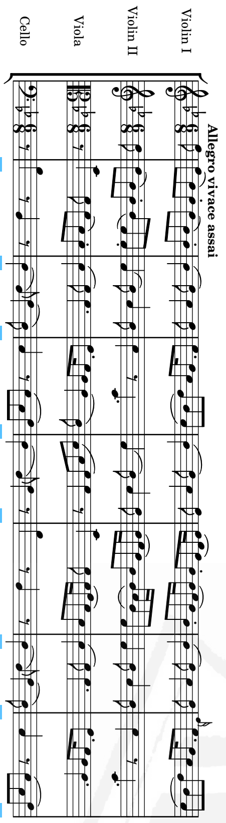


Figure 2: Example of one score in **kern format (left) representing the output of our model and the rendered western music score (right).

2.3 Data Preparation

As previously mentioned, the fact that we do not require data alignments is a clear advantage to easily build the ground truth needed to train our model. However, the majority of scores available in the public domain are usually in printed form, so we cannot automatically obtain the symbolic representation we need unless we make use of an Optical Music Recognition system. A lot of progress has been made in this area of study but unfortunately it is still insufficient to meet our precision needs, driving us to look for existing text based scores instead. After some analysis of the various types of music encoding formats within reach, we chose the humdrum toolkit [9] due to its versatility to represent polyphonic music.

The humdrum file format is a general-purpose human-readable 2D representation of music information intended to assist music researchers. The columns of the text file, separated by the tab character, represent the sources of information that produce music-related events. The lines of the text file represent the evolution of those events over time. The humdrum syntax defines the skeleton that contains other higher level schemes of music notation, like the **kern format our ground truth is based on. The **kern format is designed to encode the semantics of a western musical score, rather than the visual aspects of its printed

	Chorales	Quartets
Number of samples	352	34,512
Total duration	5.79h	20.25h
Max duration	120s	30s
Data Augmentation	No	Yes
Polyphony voices	4	4
Instruments	Pipe organ	Cello Viola Violin Flute
Pitch range	C2-A5	C2-E7
Shortest note	$1/16^{th}$	$1/64^{th}$
Irregular groups	None	Triplets
Tempo	♩ $\approx [60, 70]$	♩ $= [40, 200]$
Vocabulary Size	99	143
Train-test split %	80/20	70/30
Batch size	4	16

Table 1: Summary of the datasets' characteristics.

realization, matching nicely with the purpose of this work.

An example of a music excerpt encoded in **kern notation is shown in Figure 2 along with its associated sheet music excerpt. In this format, columns are called spines and they are associated with instruments, just like a pentagram in western sheet music. Spines may contain one single sound event or the combination of various sound events with the same canonical duration, namely a chord. Spines can also be split into two spines when two independent voices (excluding chords) occur for the same instrument. The newly created spine can be rejoined back to the original spine when the extra voice is no longer needed. This level of flexibility gives almost no restrictions to the kind of music it can support, making **kern a good candidate to endure future work.

We created two datasets out of the **kern files available in the humdrum-data repository [18]: the chorales dataset, containing 370 chorales of Bach, and the quartets dataset, containing most of the string quartets of Haydn, Mozart and Beethoven. In the chorales dataset we take each chorale as one training sample, and we use audio from expressive MIDI files synthesized with a high quality pipe organ soundfont [7]. As we did not synthesize the audio, we had to manually remove repetitions to ensure that samples are not unnecessarily long. In the quartets dataset we randomly split the scores in fragments of 3-6 measures each, and we synthesized the corresponding MIDI file obtained from the hum2mid tool, which converts **kern to MIDI using dynamic spines and articulation marks when available in the original **kern file. We removed grace notes and ornaments from the score as they cannot be properly synthesized. We also removed split spines and upper notes of all chords to ensure no more than 4 simultaneous voices were present at any given time. Samples with double dots, double sharps or double flats are out of the scope of this work and therefore discarded. On the training set only, we allow overlapping of fragments as a means of data augmentation technique. Table 1 summarizes the main characteristics of both datasets used in this work.

Figure 1 depicts the data acquisition pipeline we implemented to build our ground truth. The major inconvenience

Largo Assai	40	Allegro Moderato	120
Largo	50	Poco Allegro	124
Poco Largo	60	Allegro	130
Adagio	71	Molto Allegro	134
Poco Adagio	76	Allegro Assai	138
Andante	92	Vivace	150
Andantino	100	Allegro Vivace	160
Menuetto	112	Allegro Vivace Assai	170
Moderato	114	Poco Presto	180
Poco Allegretto	116	Presto	186
Allegretto	118	Presto Assai	200

Table 2: List of metronome markings chosen for classical music tempo annotations, given in number of quarter notes per minute.

was dealing with multiple errors present in the **kern files, such as invalid ties, wrong canonical duration of notes and rests, and missing metronome markings. While these errors do not prevent musical analysis of the scores, they become noisy labels that hinder our training process. For that reason, we had to revise all the scores manually to correct these errors and label the missing metronome markings, with the help of existing tempo annotations and the conversion shown in Table 2. Adding metronome markings ensured the synthesized audio perform at reasonable speeds according to the composer's intention. Additionally, a random scaling factor in the $\pm 6\%$ range was applied to each metronome marking to ensure tempo variability in all training samples.

The resulting **kern scores after the preprocessing stage were then encoded in a special symbolic notation intended to reduce the number of characters and ease the training process. Accordingly, each canonical duration for notes and rests including their dotted version were encoded with just one symbol of the vocabulary. Likewise, note pitches were also encoded with one symbol condensing name and octave. In our **kern dataset, barlines are always repeated for all spines, so only one barline is maintained in the output representation referring to all spines. The rest of characters are preserved in the output representation in the same way, i.e. tabs, new lines, tie symbols, fermatas and the "dot" character, which indicates that the previous note/rest still affects the current row.

3. METHOD

Once the input and output representations are defined, we can formulate the A2S task as retrieving the most likely sequence of score symbols $\hat{\mathbf{s}}$ given an audio file $\mathbf{x} \in \mathcal{X}$:

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \Sigma^*} P(\mathbf{s}|\mathbf{x}) \tag{1}$$

where Σ represents the set of characters necessary to encode the output as explained in the previous section (for instance, including "tab", "new-line" and "dot", among others). Additionally, Σ includes an *"empty"* symbol, denoted by ϵ, that is necessary to separate two or more instances of the same symbol that occur in consecutive frames.

Following successful approaches in other pattern recognition duties of similar formulation, we address this A2S with a holistic approach based on statistical models. Specifically, for learning the posterior probability provided in Eq. 1, we resort to Convolutional Recurrent Neural Networks (CRNN).

A CRNN is composed of one block of *convolutional* layers followed by another block of *recurrent* layers [19]. The convolutional block is in charge of learning how to extract relevant features from the input and the recurrent layers interpret these features in terms of sequences of musical symbols. The activations in the last convolutional layer can be seen as a sequence of feature vectors representing the input audio file, $\mathbf{x}$. Let W be the width (number of frames) of the input sequence $\mathbf{x}$. The length of the resulting features after the convolutional layer will be $L = \gamma W$, where $\gamma \leq 1$ is implicitly defined by the specific configuration of the convolutional block (which usually includes some type of down-sampling to reduce dimensionality).

The output activations of the convolutional block are then fed to the first layer of the recurrent block, and the activations of its last layer can be considered proper estimates of the posterior probabilities per frame:

$$P(\sigma \mid \mathbf{x}, j), \;\; 1 \leq l \leq L, \;\; \sigma \in \Sigma \tag{2}$$

3.1 Training

Convolutional neural networks can be trained through gradient descent using the well-known *Back Propagation* algorithm. RNN networks can be trained similarly by means of *Back Propagation Through Time* [21]. Therefore both the convolutional and recurrent blocks of a CRNN can be jointly trained by providing audio files annotated at the frame level.

In this work, however, we follow a holistic or "end-to-end" approach, which means that for each audio file we only provide its corresponding target transcript into score symbols, without any kind of explicit information about its segmentation into frames. A CRNN can be uniformly trained without this information by using the so-called Connectionist Temporal Classification (CTC) loss function [6]. The CTC training procedure is a form of Expectation-Maximization, similar to the backward-forward algorithm used for HMM training [16], that distributes the loss among all the frames to maximize Eq. 1 with respect to the ground-truth sequence.

3.2 Decoding

In order to solve Eq. 1, the most likely symbol is computed for each input feature vector of the recurrent block l, also referred as greedy decoding:

$$\hat{\sigma}_l \;=\; \arg\max_{\sigma \in \Sigma} P(\sigma \mid \mathbf{x}, l), \;\; 1 \leq l \leq L \tag{3}$$

Then, a pseudo-optimal sequence of musical symbols is obtained as $\hat{\mathbf{s}} \approx \mathcal{D}(\hat{\sigma})$, where $\hat{\sigma} = \hat{\sigma}_1 \ldots \hat{\sigma}_L$ and $\mathcal{D} : \Sigma^\star \rightarrow \Sigma^\star$ is a function which first merges all the consecutive frames with equal symbol, and then deletes all "empty" symbols [6].

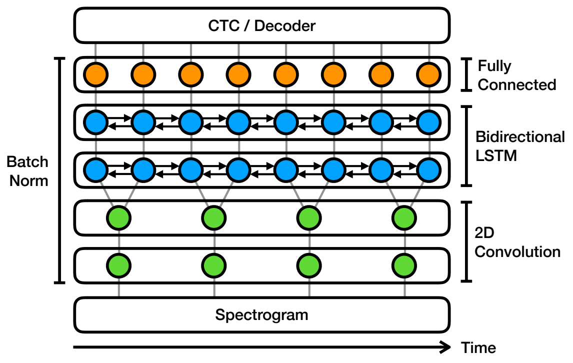


Figure 3: High-level architecture of the Convolutional-Recurrent Neural Network used in our experiments.

3.3 Architecture

The main building blocks of the CRNN considered for our experiments is illustrated in Figure 3.

The first two convolutional layers receive a 2D array containing the audio spectrogram described in section 2.1 and apply 16 filters of 3×3 with a stride of 2 in the frequency axis. We use filter striding to reduce the input dimensionality without the need of pooling layers.

For the Quartets dataset, output frames from the convolutional block are split in half, effectively doubling the number of frames feeding the next recurrent block. This is necessary to comply with the CTC loss function precondition for which the number of input frames must be greater or equal to the number of output symbols. Considering the high number of symbols per second in our sequence-based representation of a polyphonic score of the Quartets dataset, we apply this frame doubling technique at a lesser computational cost than increasing the density of the input spectogram.

The next two recurrent layers are based on Bidirectional Long Short-Term Memory (LSTM) cells, with 1024 hidden units each. The fully connected layer at the end of the recurrent block converts the output per-frame predictions to the size of the output representation vocabulary.

With the purpose of reducing overfitting, Batch Normalization layers [10] are added between any other layer excluding the input and output layers, as well as Dropout layers [20] added after all the convolutional layers and after the last recurrent layer, with a drop probability of 0.1 for the Quartets dataset and 0.2 for the Chorales dataset. A higher drop probability is required for the Chorales dataset since less data is available for training, which increases the risk of overfitting.

4. EXPERIMENTS

To the best of our knowledge, there are few specific evaluation metrics to measure the performance of a notation-level AMT method. In [14] an evaluation metric for note-level AMT is discussed, but it still cannot be directly applied to our task (e.g. we do not have note onsets and offsets). [17] adapts this metric to the A2S task by defining note duration errors instead of onsets/offsets errors. However, we believe this metric is still insufficient to properly evaluate a notation-level AMT method since, for instance, it does not take into account barlines and their effect in subsequent predictions of note durations and ties. The MV2H metric (Multi-pitch detection, Voice separation, Metrical alignment, note Value detection, and Harmonic analysis) was introduced in [13]. This metric is closer to our needs, although its source code uses timing information in seconds that is not provided by our method. We leave it as an open point for future work to establish a proper notation-level AMT metric.

In order to validate our method accuracy during training, we adopt the evaluation metrics from the ASR task as in [17], namely Word Error Rate (WER) and Character Error Rate (CER). They are defined as the number of elementary editing operations (insertion, deletion, or substitution) needed to convert the predicted sequences into the ground-truth sequences, at the word and character level respectively. Even though WER and CER are not specific to AMT, they provide a good indication of how close our score is to the ground-truth score.

In the context of our A2S task, we define words as any group of characters representing notes (including ties), rests and barlines in the output score. The “tab” and “new line” characters act as word separators, and only contribute to the CER calculation.

4.1 Training process

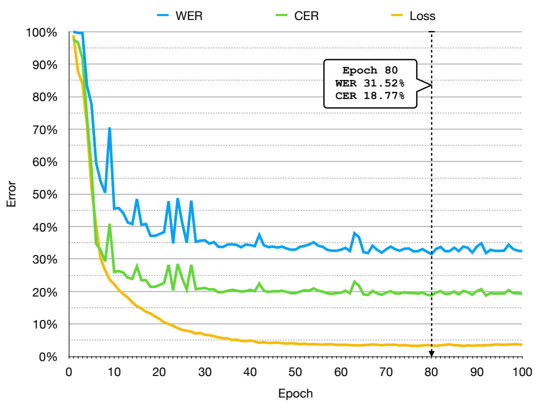
The models were trained for 100 epochs using mini-batch Stochastic Gradient Descent (SGD) optimizer, with Nesterov momentum of 0.9. Our learning rate scheduling consists of 2 cycles of 50 epochs each, starting at 0.0003 and annealing by 1.1 at every epoch. After each epoch, the WER and CER are calculated for the validation set, and the model with the lowest WER is appointed as the best model for testing purposes.

The Chorales dataset, whose samples are full-length chorales, is trained with a batch size of 4. The Quartets dataset, whose samples are small excerpts extracted from the full-length quartets, is trained with a batch size of 16. Figure 4 shows the evolution of the CTC loss, WER and CER at training time on both datasets. Each figure also highlights the epoch where the best model was obtained.

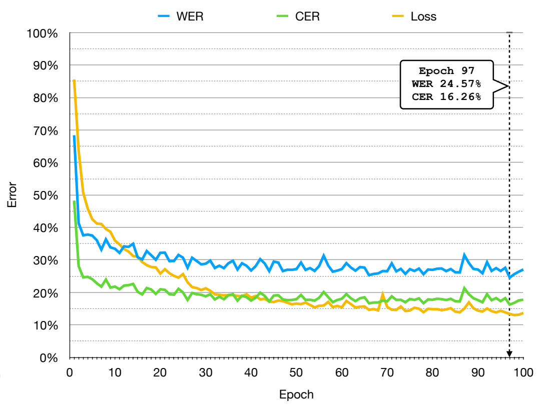
4.2 Results

The best model obtained after the training process is then evaluated against the test set for both the Chorales and Quartets datasets, giving a WER of 30.96% and CER of 18.10% for Chorales, and WER of 21.02% and CER of 13.53% for Quartets.

After analyzing all test predictions, we observe that the model occasionally generates sequences that do not comply with the **kern format. Nevertheless, we believe these formatting errors can be solved by providing more samples to the training set or imposing syntax constraints. As shown in Figure 5, most of the errors arise from wrongly

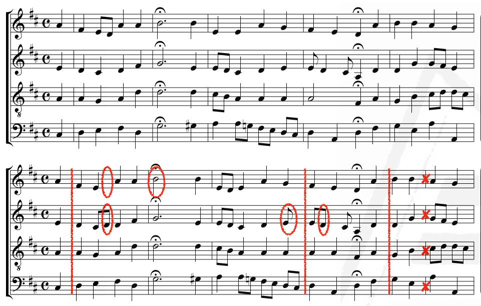


(a) Chorales dataset.

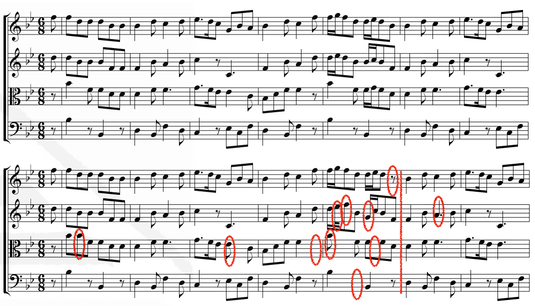


(b) Quartets dataset.

Figure 4: Evolution of loss, validation WER and CER during 100 epochs of training with a) Chorales dataset and b) Quartets dataset. Chorales WER is 30.96% and CER is 18.10%. Quartets WER is 18.10% and CER is 13.53%.



(a) Chorales dataset example



(b) Quartets dataset example

Figure 5: Excerpt of original (top row) and predicted scores (bottom row) from a test sample in a) Chorales dataset, b) Quartets dataset. The differences between original and prediction are highlight in red.

estimated note durations and barlines. Exchanging notes between voices is another common mistake our model makes, specially when voice pitches are too close or even when two voices cross their melodic lines.

The model struggles at predicting ties and triplets, which requires further analysis to determine whether it is related to barline errors, to the output representation format based on **kern, or to the lack of enough samples in the training set (i.e., ties and triplets are very infrequent in our training data compared to other symbols).

5. CONCLUSIONS

In this work, we focus on the A2S task, a hardly explored formulation consisting of extracting a full score from an audio file. Note that A2S resembles what a human would expect to get if it intends to visualize the input audio as a music score (e.g., MusicXML), unlike what most authors consider AMT where the output sequence format is intended to be further processed by a computer (e.g., MIDI).

The proposed methodology which performs the A2S task has the following advantages over other AMT methods: 1) Frame-level alignment of the ground truth is not needed; 2) The end-to-end approach avoids propagating errors from one stage to the other; 3) The output of our model is based on **kern format and can be straightforwardly translated to a valid music score.

We are aware this simplified scenario used for evaluation does not include real audio and some score symbols, but we argue the results provide the basis to open a new path of research towards notation-level AMT.

One of the main limitations of the proposed approach is the maximum-length of input sequences due to memory constraints. For example, this prevents an end-to-end training with complete songs, only allowing fragments of 2 minutes at a maximum on a typical training infrastructure. For this, we plan in a future work to explore other architectures such as the Transformer XL [5], a sequence-to-sequence model that can deal with much longer sequences. Other future works include defining an evaluation metric for A2S and building a dataset from real audio to validate the approach with actual music performances.

6. ACKNOWLEDGMENT

This work was funded by the Spanish Ministerio de Economia, Industria y Competitividad through HISPAMUS project (TIN2017-86576-R).

7. REFERENCES

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, and Zhenyao Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *Computer Research Repository*, abs/1512.02595, 2015.

[2] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic Music Transcription: An overview. *IEEE Signal Procesing Magazine*, 36(1):20–30, 2019.

[3] Ralf Gunter Correa Carvalho and Paris Smaragdis. Towards End-to-end Polyphonic Music Transcription: Transforming Music Audio Directly to a Score. In *IEEE Workshop for Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017.

[4] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing Human Piano Performances into Music Notation. In *Proc. of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York, USA*, 2016.

[5] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *Computer Research Repository*, abs/1901.02860, 2019.

[6] Alex Graves, Santiago Fernández, Faustino Gómez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proc. of the 23rd International Conference on Machine Learning*, International Conference on Machine Learning, pages 369–376. ACM, 2006.

[7] Margaret Greentree. Chorale Harmonizations by Bach. `http://sporadic.stanford.edu/Chorales/`.

[8] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proc. of the 19th International Society for Music Information Retrieval Conference*, pages 50–57, 2018.

[9] David Huron. The Humdrum Toolkit: Software for Music Research. `http://www.humdrum.org`.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July*, pages 448–456, 2015.

[11] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proc. of the 17th International Society for Music Information Retrieval Conference*, pages 475–481, 2016.

[12] Rainer Kelz and Gerhard Widmer. An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. In *Audio Engineering Society Conference: AES International Conference on Semantic Audio*, Jun 2017.

[13] Andrew McLeod and Mark Steedman. Evaluating automatic polyphonic music transcription. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France*, pages 42–49, 2018.

[14] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2018.

[15] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. Rhythm transcription of polyphonic piano music based on merged-output HMM for multiple voices. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 25(4):794–806, 2017.

[16] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice hall, 1993.

[17] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-End Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France*, 2018.

[18] Craig S. Sapp. humdrum-data. `https://github.com/humdrum-tools/humdrum-data.git`.

[19] Baoguang Shi, Xiang Bai, and Cong Yao. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2298–2304, 2017.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[21] Ronald J. Williams and David Zipser. Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In Yves Chauvin and David E. Rumelhart, editors, *Backpropagation: theory, architecture and applications*, pages 433–486. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

Chapter 5

Data representations for audio-to-score monophonic music transcription

Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. Data representations for audio-to-score monophonic music transcription. *Expert Systems with Applications*, 162:113769, 2020

Expert Systems with Applications 162 (2020) 113769

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Data representations for audio-to-score monophonic music transcription

Miguel A. Román *, Antonio Pertusa, Jorge Calvo-Zaragoza

U.I. for Computing Research, University of Alicante, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain

ARTICLE INFO

Article history:
Received 7 April 2020
Revised 24 June 2020
Accepted 15 July 2020
Available online 26 July 2020

Keywords:
Automatic music transcription
Audio processing
Neural networks
Audio to score
Monophonic music

ABSTRACT

This work presents an end-to-end method based on deep neural networks for audio-to-score music transcription of monophonic excerpts. Unlike existing music transcription methods, which normally perform pitch estimation, the proposed approach is formulated as an end-to-end task that outputs a notation-level music score. Using an audio file as input, modeled as a sequence of frames, a deep neural network is trained to provide a sequence of music symbols encoding a score, including key and time signatures, barlines, notes (with their pitch spelling and duration) and rests. Our framework is based on a Convolutional Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) loss function trained in an end-to-end fashion, without requiring to align the input frames with the output symbols. A total of 246,870 incipits from the Répertoire International des Sources Musicales online catalog were synthesized using different timbres and tempos to build the training data. Alternative input representations (raw audio, Short-Time Fourier Transform (STFT), log-spaced STFT and Constant-Q transform) were evaluated for this task, as well as different output representations (Plaine & Easie Code, Kern, and a purpose-designed output). Results show that it is feasible to directly infer score representations from audio files and most errors come from music notation ambiguities and metering (time signatures and barlines).

1. Introduction

Automatic music transcription (AMT) is a music information retrieval (MIR) task whose final goal is to extract a human-readable and interpretable representation (i.e., a music score), from an audio signal. A music score is a guide to perform a piece of music, and it can be represented in different ways. The most extended representation is the modern staff notation used in common Western music, which can be encoded as symbolic data for computational purposes, namely a digital music score. Precise high-level musical information can be extracted from digital music scores, making them suitable to process, retrieve and classify music content (Corrêa & Rodrigues, 2016). There are different digital music score formats such as Plaine & Easie Code (Brook, 1965), Kern (Sapp, 2005) or MusicXML (Good, 2001), among others.

The most obvious application of AMT is to help a musician write down the music notation of a performance from its audio recording, which is a time-consuming task when it is done by hand. Additionally, AMT can also be useful for other MIR tasks, such as plagiarism detection, artist identification, genre classification, real-time score following, music style transfer or accompaniment generation. In general, AMT enables symbolic music algorithms to analyze recorded music.

In a modern music score (Schobrun, 2005), sounds are represented by *notes* placed on a *staff*, which is made of 5 horizontal lines representing the flow of time. The vertical position of the note in the staff determines its pitch, and the shape of the note symbol determines its duration. Silence is represented by *rest* symbols, whose duration is also determined by its shape. A staff begins with a *clef* symbol, which defines the pitch reference for the written notes. After the clef, a *time signature* determines the metric information or rhythm, and a *key signature* determines which note pitches must be altered according to the tonality of the musical piece. Based on the time signature, notes and rests are divided in *measures* of equal duration using *barlines*. Notes whose duration go beyond the current measure are joined with another note of the same pitch in the next measure using a curved line named *tie*. However this is only the basics, as a modern music score usually contains other information to indicate changes in *dynamics* (i.e., volume of the played notes) and *tempo* (i.e., duration of the played notes).

In order to extract a modern score from an audio signal, it is necessary to estimate the notes (with their pitch and duration) and the rests (duration) together with other symbols intended to

* Corresponding author.
E-mail addresses: mroman@dlsi.ua.es (M.A. Román), pertusa@dlsi.ua.es (A. Pertusa), jcalvo@dlsi.ua.es (J. Calvo-Zaragoza).

https://doi.org/10.1016/j.eswa.2020.113769
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

aid the interpretation of the score, such as key signatures, metering information (time signature and barlines) and clefs. However, this is just the basic information since a score is also a mean of communication between the composer and the performer, in which the former expresses how the musical piece should be performed by the latter. Thus, a score also contains a myriad of symbols intended to affect the acoustic characteristics of the written notes (e.g., slurs, crescendo, rubato, pizzicato). The meaning of these symbols is vague and in some cases has changed over different periods (e.g., grace notes, tempo markings). As a score does not unequivocally represent a musical piece and vice versa, the output of any AMT system can only be an approximation of the score that a musician used to perform the recorded audio.

Experienced musicians are capable of writing down a score by listening to an audio recording, notwithstanding slight differences compared with the original score due to inherent notation ambiguities. Music students are initially trained by transcribing monophonic scores (only one note playing simultaneously) in melodic dictations. The polyphonic scenario (many notes playing simultaneously) is much more complex and even some expert musicians are not able to transcribe full scores correctly.

In this work, we propose to focus on a hardly explored formulation, namely the Audio-to-Score (A2S) task, consisting of extracting a full score from the audio file in an end-to-end fashion. To the best of our knowledge, addressing audio-to-score transcription in one step is only found in our preliminary work (Román, Pertusa, & Calvo-Zaragoza, 2018; Román, Pertusa, & Calvo-Zaragoza, 2019), and in Carvalho and Smaragdis (2017).

In Román et al. (2018), the AMT problem is formulated as an automatic speech recognition (ASR) problem. Using monophonic (i.e., only one note playing simultaneously) audio as input, and a sequence of symbols (analogously to the spoken words) as outputs enables the application of several methods that were originally developed for ASR. In particular, Román et al. (2018, 2019) adopted the Convolutional Recurrent Neural Network architecture (Shi, Bai, & Yao, 2017a), which is the same architecture employed by Deep Speech 2 (Amodei et al., 2015). The present approach extends the contents of Román et al. (2018) by training the model with a much larger dataset (246,870 vs 71,400), using different instruments for the synthesis of the audio excerpts (instead of only piano), evaluating different input and output representations, and analyzing the nature of errors.

The main goal of this work is to find the most adequate input and output representations for this problem, as well as to design a proper neural network architecture to tackle the A2S task from these data.

2. Background

Even though the final goal of AMT is to provide a valid score out of an audio recording, most authors working on this task focus on one intermediate goal (Benetos, Dixon, Duan, & Ewert, 2019), which is pitch estimation. AMT systems can be roughly classified into four categories (Benetos et al., 2019). (1) Frame-level AMT or pitch estimation outputs the fundamental frequencies present at all time frames of the input audio. This is usually performed in each frame independently, although contextual information is sometimes considered in a post-processing stage. (2) Note-level (or note tracking) AMT not only estimates pitches in each individual frame, but also connects pitch estimates over time, producing a piano-roll representation as the output similarly to frame-level methods, but characterizing each note with pitch, onset and offset times. (3) Stream-level AMT goes one step further by clustering notes belonging to the same audio source characterized by its timbre. (4) Finally, notation-level AMT aims at the final goal of AMT, generating a human-readable score, which is the task addressed in the present work.

Regarding frame-level AMT, established algorithms such as SPICE (Gfeller et al., 2019) or YIN (de Cheveigné & Kawahara, 2002) and its variants (such as probabilistic YIN (Mauch & Dixon, 2014)) can obtain very reliable results for monophonic pitch estimation, which can be considered an almost solved task. In consequence, most AMT researchers only target the polyphonic pitch estimation problem, which is an extremely challenging task that still leaves room for improvement. Polyphonic pitch estimation methods include signal processing based techniques (Su & Yang, 2015), Negative Matrix Factorization (NMF) (Benetos & Dixon, 2013) and neural networks (Kelz et al., 2016) among others.

For note-level AMT, deep neural networks achieve state of the art results (Hawthorne et al., 2018; Kim & Bello, 2019; Ycart, McLeod, Benetos, & Yoshii, 2019a; Ycart, Stoller, & Benetos, 2019b), particularly using Convolutional Neural Networks (CNN) to output a piano-roll representation.

At present, the main focus of AMT research is on note-level AMT, still an open problem. For this reason, there are very few works (Arora & Behera, 2015) addressing stream-level AMT, which is closely related to instrument source separation.

Likewise, few researchers address notation-level AMT, which is the subject of this work. Regarding polyphonic sources, Nakamura, Benetos, Yoshii, and Dixon (2018a) uses a pipeline of multi-pitch estimation, note tracking, rhythm quantization, and score typesetting for polyphonic music. The results are still unsatisfactory according to the authors, who also suggest the integration of a music language model to improve the metrics. Cogliati, Temperley, and Duan (2016) approaches the problem by focusing on rhythm quantization of MIDI files, but do not tackle AMT end to end. The first work that formulates notation-level AMT as a sequence-to-sequence translation is Carvalho and Smaragdis (2017), but they intend to show that a neural network is capable of learning the task and the nature of the experiments is still very simple (i.e., note duration is constant, all samples limited to one measure of 4/4 metering).

Nonetheless, to the best of our knowledge there is no previous work addressing monophonic notation-level AMT other than Román et al. (2018). Although frame-level pitch estimation can be accurately performed on monophonic audio, the task of obtaining a full monophonic score has not received attention from the community. As a matter of fact, the result of a monophonic pitch estimation or pitch tracking method, which shows the evolution of pitches over time, is not useful in most situations since it lacks musical meaning, whereas obtaining a complete score from a singer or a monophonic instrument would be very useful for musicians and music students. For example, jazz trumpet players could use a monophonic notation-level AMT system to extract the score out of their recorded performances, which very often include solo improvisations never played or written beforehand.

Although monophonic pitch estimation is a relatively simple task, the extraction of the full score out of the pitches is not trivial. Problems such as rhythmic quantization (i.e., note and rest durations, barline locations, time signature estimation) and pitch spelling (i.e., G♯ vs. A♭) are still open, notably if we consider the real duration of note symbols (e.g., ♩ 𝅗𝅥 ♪) is never the same from one composition to the other, and even from one performance to the other. Moreover, there can be different ways to represent the same audio content using the western score notation. These intrinsic challenges of monophonic notation-level AMT are common to polyphonic notation-level AMT. Hence, some of the lessons learned in this work can be useful for future notation-level AMT works addressing polyphonic music.

Notation-level transcription can be divided into various subtasks, including pitch estimation, note tracking, pitch spelling, instrument recognition, rhythmic quantization and extraction of music dynamics. Each of these sub-tasks is very challenging on its own, even for trained musicians, leading many researchers to address AMT only as a frame or stream level estimation problem (Cemgil, Kappen, & Barber, 2003; Benetos et al., 2019).

As shown in Fig. 1, recent AMT methods that produce a notation syntax as output usually split the task into two main stages: they first convert the audio signal into the piano-roll representation, and in a second stage, they convert the estimated piano-roll into the final musical notation. Commercial software like Audioscore[1] follows this two-stage approach. In this case the first stage is fully automatic, but the second stage requires human intervention to set the time signature and barlines manually. Nonetheless, this approach has some drawbacks, since a wrong detection in a given stage (such as the multi-pitch detection) can be propagated through the next processing stages (Román et al., 2019). This is the main motivation to use end-to-end methods, where pitches can be better predicted together with the time information and dynamics, and vice versa, thanks to the underlying music language model that can be learned from the training data.

All notation-level AMT methods, including both monophonic and polyphonic, suffer from two key limitations that hinder its progress. The first one is the scarcity of quality datasets. Even though there are many audio recordings with their corresponding scores, they are not aligned and, since they can not be split, samples are usually very long to fit the training of deep neural networks. Unlike Automatic Speech Recognition (ASR), there is no trivial way to break a continuous musical piece into smaller excerpts based on the score. The second key limitation is the lack of proper evaluation metrics that can measure how close two scores are from each other, apart from a character by character comparison. There are some attempts to develop evaluation metrics for notation-level AMT (e.g., (McLeod & Steedman, 2018)), but they are still far from becoming a *de facto* standard that allows a proper comparison between different methods.

3. Data acquisition

As previously mentioned, one of the key limitations of notation-level AMT is the lack of quality datasets for training neural networks, namely a vast collection of musical excerpts of real audio data together with the corresponding aligned score in a textual format (e.g., MusicXML).

Our end-to-end approach has the advantage of not requiring audio frames aligned with the corresponding score notation symbol, which allows us to build our own dataset from existing digital scores by synthesizing the associated audio. Synthetic data is not optimal for training neural networks but it has been found an effective and inexpensive alternative in the absence of real data (Tremblay et al., 2018).

Fig. 2 outlines the pipeline we developed to create our ground truth. Our data source is the Répertoire International des Sources Musicales (RISM) (RISM, 2017) online catalog, which contains more than 1 million music incipits, encoded in Plaine& Easie Code (Brook, 1965) more commonly known as PAE format. In the *Data Extraction* block, we select a subset of more than 300.000 incipits and extract the PAE contents along with other metadata such as clef, key signature, time signature, and target instrument. The next two blocks in the pipeline are the *Score Parsing* block and the *Audio Synthesis* block. The former removes multi-rests and grace notes, which will not be considered for this work, and discards scores with invalid format and with incorrect measure lengths. The latter converts the resulting PAE files into MIDI by sequentially running the tools `pae2kern` and `hum2mid` available at the humdrum extra toolkit (Sapp, 2013).

The General MIDI program was based on the instrument metadata extracted from RISM and a random tempo between 80 and 120 is chosen to ensure variability in the note duration. Audio files were synthesized from MIDI files via FluidSynth with the default sound font (FluidR3 GM Bank) at 22,050 Hz sampling rate and compressed to mono FLAC audio format to reduce disk space while preserving the audio content in a lossless manner.

After this initial processing stage, we ended up with 313,493 samples of FLAC (input) and PAE (output) files, which will form our training data. Aiming at curating the dataset, the *Filtering block* in Fig. 2 filters out those samples that do not have their clef in [G2, F4, C1, C3, C4], their key signature in [0–5 sharps, 0–5 flats], their time signature in [4/4, 2/2, 3/4, 2/4, 3/8, 6/8, 9/8, 12/8, 6/4, 4/2, 3/2], their note and rest durations in [1, 1/2, 1/4, 1/8, 1/16, 1/32] (including dotted versions), or their note pitches in the [C2-B6] range. The samples removed by this filter are music notation outliers, extremely rare music scores that are not sufficiently represented in our data.

Additionally, we discarded samples containing double dots, double flats, double sharps, measure repetitions, and PAE rhythmic patterns, as well as samples that change the initial clef/key/time signatures. And lastly, we also discarded audio samples longer than 18 s and those with a high number of output symbols per second, to guarantee a minimum 1:1 ratio between input frames and output symbols required for CTC loss function.

After this final filtering stage, we ended up with 246,870 samples that were randomly divided into training (70%), validation (15%) and test (15%) sets. The same sets were used for all the experiments to ensure results can be compared across all input/output representations and models.

4. Data representation

Under this end-to-end scenario, the input of the proposed model is a sequence of audio frames, and the output is a sequence of text characters representing the score. Given the nature of our approach, we intend to analyze the impact of the data representation with regards to both input and output formats. In the following sections, we describe the different alternatives that we have considered.

4.1. Inputs

All input representations were obtained from an audio waveform sampled at 22,050 Hz, and they apply the same window stride to ensure that input sequences have equal length regardless of its representation.

- *Raw audio*. Raw waveforms were directly used as input by adding SincNet (Ravanelli & Bengio, 2018) as the first layer of the model. SincNet layer extracts audio features from waveforms by discovering meaningful filters for our problem. SincNet is based on parameterized 'sinc' functions, which implement band-pass filters whose coefficients (low and high cutoff frequencies) are learned during training. A set of 144 filters were used with a window length of 2048 samples and a window stride of 512 samples. A Hamming window was selected to control spectral leakage after applying the filters. The initial filter coefficients were initialized with Mel-scaled bins across the valid frequency range (from 0 to Nyquist).
- *STFT*. The magnitude spectrogram from the Short Time Fourier Transform was also evaluated, using a Hamming window of 2048 samples and hop size of 512 samples.

[1] https://www.neuratron.com/audioscore.htm.

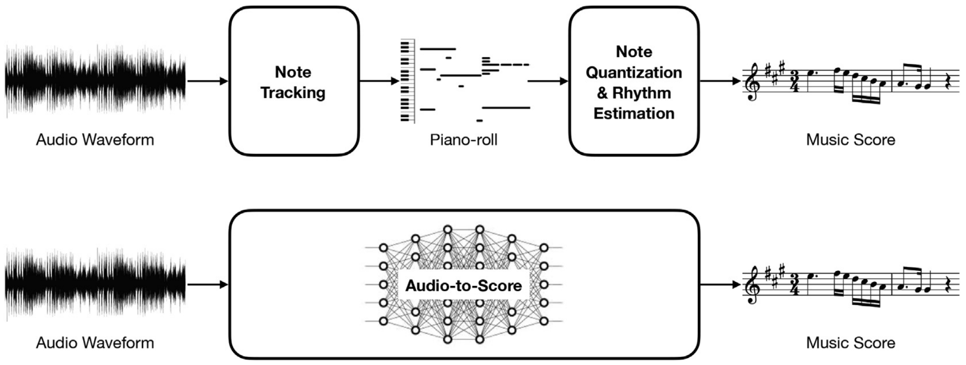


Fig. 1. Overview of notation-level automatic music transcription methods. Top: two-stage methods based on piano-roll estimation. Bottom: end-to-end methods based on deep neural networks and the object of this study.

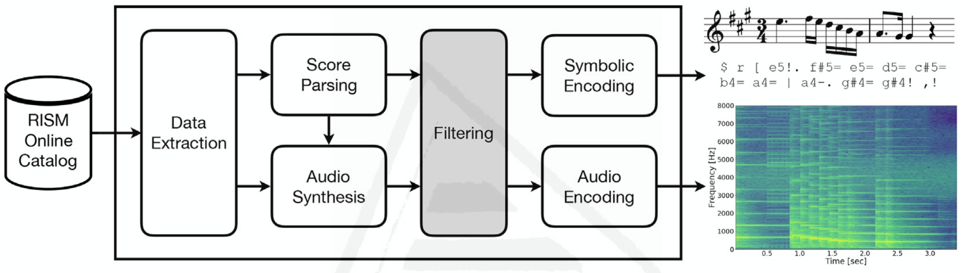


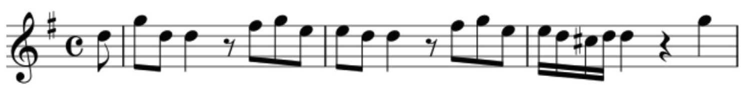
Fig. 2. Training data acquisition pipeline.

- *LogSTFT.* A log-spaced filter bank was applied to the previous STFT to get 24 bins per octave, starting from C2 and extending across 6 octaves.
- *CQT.* Constant-Q transform (Schörkhuber & Klapuri, 2010) spectrogram, using 24 bins per octave, starting from C2 and extending across 6 octaves. In this type of spectrogram, the window size is variable in order to guarantee the same resolution in the frequency axis at the cost of penalizing the resolution in the time axis. However, the hop size is constant and set to 512 samples in order to ensure the same number of frames as the previous representations.

4.2. Outputs

Different output representations were obtained from the original input incipits encoded in standard PAE format. Fig. 3 shows an example of a score excerpt and how it was encoded to the 3 output representations we considered.

- *CTC-friendly.* This is a semantic notation we created to ease Connectionist Temporal Classification (CTC) decoding, explained in more detail in the following section. It is formed by a sequence of words, separated by spaces, of one of these types: clef, key signature, time signature, barline, rest, note, tie. A single-character word was used to encode clef, key signature, time signature, barline and tie. Multiple characters were used to encode notes, including the pitch spelling (e.g., a♯4, c2, b♭3) followed by one character to encode the canonical duration, plus the standard notation dot character to extend the duration when needed. Explicit sharp and flat notes were encoded regardless of key signature or measure accidentals. Rests were encoded using one special symbol followed by the same duration encoding used for notes. The alphabet size is 52 characters, and the average word size is 2.53 characters.
- *PAE-based.* This notation is based on the PAE (Brook, 1965) format with the following changes to match the requirement of our output sequences: (1) clef, key signature and time signature were encoded following the PAE format and placed at the beginning of the incipit without their special characters and always in the same order; (2) notes, rests, ties, and barlines were split by spaces; (3) note/rest modifiers (i.e., octave, duration, accidental) were placed right before the first note or rest that they affect to; (4) note modifiers always follow this precedence: duration, octave, accidental; (5) there are implicit sharp and flat notes based on the key signature and measure-bound accidentals. The alphabet size is 24 characters and the average word size is 1.87 characters.
- *Kern-based.* This notation is based on the Humdrum (Huron, 1995) Kern format, which represents music in lines of tab-separated columns. Each line represents an instant of time where one or more note onsets exist, whereas each column, or spine in Humdrum's nomenclature, represents a different pentagram or voice in a score. As we are dealing with monophonic excerpts, our kern files have one spine only. We take advantage of this fact to create a one-line representation of the score by replacing the carriage return characters with spaces. Moreover, we make the following changes to match our output sequence format: (1) clef, key signature and time signature were encoded as described in the previous PAE format section; (2) comment records and interpretation records were removed; (3) middle notes of ties do not have any special character. Only the initial and final tie characters (i.e., '[' and ']')



CTC-friendly encoding

```
$ t ( d5- | g5- d5- d5! ,- f#5- g5- e5- | e5- d5- d5! ,- f#5- g5- e5- |
e5= d5= c#5= d5= d5! ,! g5! |
```

PAE-based encoding

```
G-2 xF 4/4 8''D / G D 4D 8- F G E / E D 4D 8- F G E / 6E D xC D 4D - G /
```

Kern-based encoding

```
G-2 xF 4/4 8dd = 8gg 8dd 4dd 8r 8ff# 8gg 8ee = 8ee 8dd 4dd 8r 8ff# 8gg 8ee =
16ee 16dd 16cc# 16dd 4dd 4r 4gg =
```

Fig. 3. Example of score encoding to the output representations we considered.

were considered and they were placed next to the affected notes; (4) there are explicit sharp and flat notes regardless of key signature or measure-bound accidentals. The alphabet size is 31 characters and the average word size is 2.64 characters.

5. Method

We built our architecture based on Román et al. (2018), which is an improved version of the Convolutional Recurrent Neural Network (CRNN) trained with Connectionist Temporal Classification (CTC) loss function.

Our notation-level AMT approach aims at estimating which music score, modeled as a structure containing symbols from a fixed alphabet of music notation, would define the input audio most likely. This is what we denote by the Audio-to-Score (A2S) task. Note that A2S resembles what a human would expect to get if it intends to visualize an audio file as a music score (e.g., MusicXML), unlike traditional AMT where the output is a construction intended to be further processed by a computer (e.g., piano-roll, MIDI, etc.).

Let $\mathscr{X}$ be the domain of audio files and Σ the alphabet of music score symbols. The aim of our A2S is to compute a function that maps any audio file into a sequence of symbols, i.e., a function $f:\mathscr{X}\rightarrow\Sigma^{*}$. The A2S task is therefore formulated as retrieving the most likely sequence of music symbols $\hat{\mathbf{s}}$ given an $\mathbf{x}\in\mathscr{X}$:

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in\Sigma^{*}} P(\mathbf{s}|\mathbf{x}) \tag{1}$$

Following successful approaches in other sequence labeling tasks, we address the A2S with an end-to-end approach based on statistical models. For learning the posterior probability of Eq. (1), we resort to Convolutional Recurrent Neural Networks (CRNN).

A CRNN consists of one block of *convolutional* layers connected to a block of *recurrent* layers (Shi, Bai, & Yao, 2017b). Each convolutional layer usually includes a max-pooling layer (or filters with a stride greater than 1) that reduces the dimensionality of its output. The convolutional block is in charge of learning relevant features from the input and the recurrent layers process these features in terms of sequences of musical symbols.

Activations of the last convolutional layer can be seen as a sequence of feature vectors representing the input audio file, $\mathbf{x}$. Let M be the width of the input sequence $\mathbf{x}$. The length of the resulting features after the convolutional layer will be $J=\gamma M$, where $\gamma\leqslant 1$ is defined by the specific max-pooling parameters.

These activations are then fed to the first layer of the recurrent block, and the unit activations of the last recurrent layer are considered estimates of the posterior probabilities per frame:

$$P(\sigma|\mathbf{x},j), \quad 1\leqslant j\leqslant J, \quad \sigma\in\Sigma\cup\{\epsilon\} \tag{2}$$

where ϵ is a special *"non-character"* symbol that is necessary to detect two or more consecutive instances of the same symbol (Graves, 2008).

5.1. Training

Convolutional neural networks can be trained through gradient descent using the well-known *Back Propagation* algorithm. RNN networks can be trained similarly using *Back Propagation Through Time* (Williams & Zipser, 1995). Therefore both the convolutional and recurrent blocks of a CRNN can be jointly trained by providing audio files annotated at the frame level.

In this work, we follow an end-to-end approach, which means that for each audio file we only provide its corresponding target transcription of score symbols, without any kind of explicit information about its segmentation into frames. A CRNN can be uniformly trained without this information by using the Connectionist Temporal Classification (CTC) loss function (Graves, Fernández, Gomez, & Schmidhuber, 2006). The CTC training procedure is a form of Expectation–Maximization, similar to the backward-forward algorithm used for HMM training (Rabiner & Juang, 1993), that distributes the loss among all the frames to locally maximize Eq. (1) with respect to the ground-truth sequence.

To improve convergence and prevent overfitting, Batch Normalization layers (Ioffe & Szegedy, 2015) are also added between any other layer excluding the input and output layers.

5.2. Decoding

For solving Eq. (1), the most likely symbol is computed for each input feature vector of the recurrent block j:

$$\hat{\sigma}_j = \arg\max_{\sigma\in\Sigma\cup\{\epsilon\}} P(\sigma|\mathbf{x},j), 1\leqslant j\leqslant J \tag{3}$$

Then, an approximately optimal sequence of musical symbols is obtained as $\hat{\mathbf{s}}\approx\mathscr{F}(\hat{\sigma})$, where $\hat{\sigma}=\hat{\sigma}_1\ldots\hat{\sigma}_J$ and $\mathscr{F}:\Sigma'^{\star}\rightarrow\Sigma^{\star}$ is a function which first merges all the consecutive characters such that $\hat{\sigma}_j=\hat{\sigma}_{j-1}$ and then deletes all the non-character symbols $(\sigma_j=\epsilon)$ (Graves et al., 2006). This is known as greedy decoding.

In our work, we replaced greedy decoding with CTC-based beam search decoding, which applies beam search algorithm to find the n-best paths with one particularity: when exploring new paths, if two or more of them lead to the same sequence (after merging consecutive characters and deleting the non-character symbol), they are grouped into a single path and their probabilities are added up. After all frames are decoded, the path with the highest probability is chosen as the output sequence.

5.3. Implementation details

In order to adapt the topology for the different input/output configurations and improve the accuracy, we made the following changes to the model described in Román et al. (2018).

First, we adapted the convolutional stage depending on the input representation to ensure the same number of features per frame in the recurrent stage, and we also reduced the size of the filters to increase the resolution in the frequency and time axis. Table 1 summarizes the convolutional stage hyperparameters we selected following this rationale.

In addition, zero-padding was applied not only at the input layer (to match the length of the largest sequence of the batch), but in the same manner at the output of all the layers of the convolutional stage.

For the raw waveform input representation only, we added an extra filter bank layer before the convolutional layers with learnable filter bands, based on Ravanelli and Bengio (2018), and changed 2D convolutions by 1D convolutions 5. The reason behind this change is to give freedom to the model to learn filters that might not be adjacent in the frequency axis.

Finally, we also reduced the number of recurrent layers from 3 to 2, increased the number of hidden units from 1,024 to 1,536 and replaced Gated Recurrent Units (GRU) with LSTM cells. We made all these changes after trying different values and finding a trade-off between reasonable performance and computational cost.

Fig. 4 depicts the high level CRNN architecture used for spectrogram-based input representations and 5 depicts its counterpart for raw waveform input representation.

In order to produce comparable results for all the training models, we used the same set of samples for the training, validation and test. With the same motivation, we set the same training hyper-parameters summarized in Table 2.

Table 1
Convolutional layers hyper-parameters based on input representation.

Input	Number of Filters	Kernel Size	Stride
Raw	2048, 2048	3, 3	1, 1
STFT	8, 8	9×3, 3×3	2×1, 1×1
LogSTFT	16, 16	3×3, 3×3	1×1, 1×1
CQT	16, 16	3×3, 3×3	1×1, 1×1

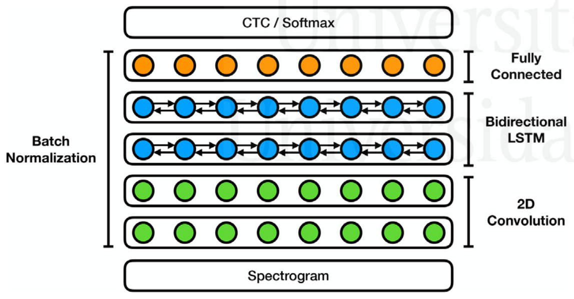


Fig. 4. CRNN with spectrogram input.

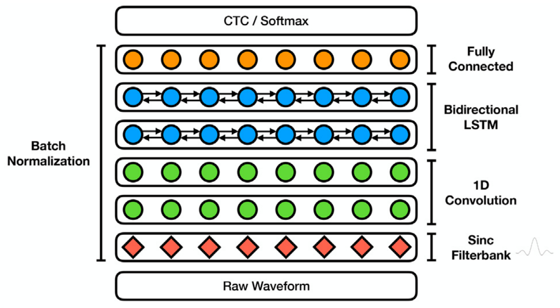


Fig. 5. CRNN with raw waveform input.

Table 2
Selected hyper-parameters for all the models.

Batch size	20
Epochs	25
Optimizer	SGD with Nesterov
Learning rate	3e-4
Momentum	0.9
Learning rate annealing	1.1

6. Evaluation

6.1. Metrics

Unfortunately, there are not existing standard A2S evaluation metrics that can be adopted for notation-based methods, since this is still an open problem due to the complexity of the score notation. For example, a good metric should count score errors only once, hence if the time signature is wrongly predicted, barlines should not be counted as errors based on their ground truth location. Likewise, errors caused by music notation ambiguities should be dismissed, for example, a quarter dotted note should be equivalent to a quarter note tied to an eighth note.

Having this in mind, we borrowed the most common evaluation metrics from the ASR task, namely Word Error Rate (WER) and Character Error Rate (CER), to validate our results. As our output is also made by words separated by spaces, we can use these metrics effortlessly. WER is an indication of the number of words incorrectly predicted and is measured as the edit distance of the sequence of output words with respect to the ground truth. CER is an indication of the number of characters incorrectly predicted and is measured as the edit distance of the sequence of output characters (after removing the spaces) with respect to the ground truth. At training time, we keep the model with the lowest WER in the validation set, calculated after each epoch. We chose to minimize WER as it gives an indication of how many musical symbols (e.g., notes) should be manually edited by a musician to fix the output score, while CER does not provide a direct relationship between errors and musical symbols.

Additionally, we also used the evaluation metrics of Nakamura, Benetos, Yoshii, and Dixon (2018b) in order to provide a reference for AMT readers who are knowledgeable of existing note-level AMT methods, even though our work belongs to the notation-level AMT category. These evaluation metrics also provide an alternative way to compare our experiments beyond using WER and CER. The total number of notes in the ground truth is denoted by N_{GT}, that of estimated notes by N_{est}. The number of notes with pitch errors is denoted by N_p, that of extra notes by N_e, and that of missing notes by N_m. The number of matched notes is defined as $N_{match} = N_{GT} - N_m = N_{est} - N_e$. Then we define the pitch error rate as $E_p = N_p/N_{GT}$, extra note rate as $E_e = N_e/N_{est}$, and missing note rate as $E_m = N_m/N_{GT}$. Onset/offsets errors are also reported in Nakamura et al. (2018b). As we are dealing with note durations instead of onsets/offsets, we include an alternative error metric E_d which is calculated similarly to the pitch error E_p but using note duration errors, denoted by N_d. Thus, we define the duration error rate as $E_d = N_d/N_{GT}$.

6.2. Results

Table 3 shows the WER and CER metrics for all the combinations of input and output representations, measured on the test set using the CTC beam search decoding with 10 search paths. Concerning the input signal, it can be observed that spectrogram-based representations perform similarly, whereas a raw input representation provide significantly worse results. With respect to the output representation, CTC-friendly reports the best results on average, whereas both PAE and Kern-based output representations perform similarly, with a slightly worse performance. According to these results, the best combination is that of LogSTFT input with CTC-friendly output, that yields 9.96 and 3.15 of WER and CER, respectively.

The results of applying (Nakamura et al., 2018b) evaluation metrics as explained in the previous section are reported in Table 4. As can be noted, CTC-friendly output representations score the lowest in all error categories. However, the choice of the input representation hardly affects the magnitude of errors for any given output representation.

Overall, extra note errors (E_e) are very scarce in all scenarios, which is expected for monophonic transcription, and duration errors (E_d) are notably higher when considering rests as well as notes. PAE-based output representations exhibit the worst pitch error (E_p) and duration error (E_d), which makes sense considering the implicit duration, octave, and accidentals for most notes in this particular representation. On the other hand, Kern-based output representations score the worst missing notes errors (E_m), due to the higher number of symbols required to represent a note.

For a better comparison of the evaluated input/output representations, we group notation-level transcription errors into 8 categories that are shown in Table 5.

As can be seen, highest errors are related to *Barline* and *TimeSig*, which is expected as many incipits do not contain enough contextual information to properly predict them. We should notice here that audio is not synthesized with dynamic expression, thus without accents indicating strong and weak beats inside a measure, which also hampers the estimation of metric information. For spectrogram-based input representations, CQT has considerably higher *Barline* and *TimeSig* error rates, which indicates that the variable window size used to create this kind of spectrogram is negatively affecting the time-based predictions as expected (i.e., CQT improves frequency resolution at the expense of lower time resolution).

Focusing on the output representation, Kern-based has the highest *Format* and *Tie* errors, which suggests that its higher verbosity and the different way to represent ties (i.e., tie character is added to each note instead of being an independent word between notes) are more difficult to learn. On the other hand, CTC-friendly representation displays the lowest *Note* errors of all, while PAE-based again exhibits the highest *Note* errors, which is explained by the way it encodes notes with implicit modifiers (i.e., whenever a note modifier such as the change of duration is incorrectly predicted, it will very likely affect the prediction of multiple subsequent notes). Nonetheless, PAE-based output representation has the lowest *Format* errors, due to having the smallest alphabet size. Based on *Format* errors, the addition of a language model to the CTC decoding stage seems a very promising approach to increase accuracy.

Regarding *Clef*, *KeySig* and *TimeSig* errors, CTC-friendly generally shows better results, suggesting that the minimal character encoding for these symbols benefits its prediction accuracy. Overall, *KeySig* error rates are significantly lower than *Clef* error rates, hinting a higher correlation between note spellings and key signature than between note spellings and clef.

Fig. 6 depicts the distribution in absolute terms of transcription errors based on its category for each input/output representation. We notice a very similar distribution regardless of the input representation, being *Barline* and *TimeSig* the most prominent source of errors in all cases. *Note* errors are also high due to the fact that they are the most frequent symbol of every score. This figure also illustrates the highest proportion of *Note* errors for PAE-based representation and *Format*/*Tie* errors for Kern-based representation, as we previously noticed.

Finally, we have also measured WER and CER for different types of instruments using our best model (i.e., logSTFT and

Table 3
Word Error Rate (WER)/ Character Error Rate (CER) using different inputs and outputs. Best result is highlighted in bold text.

	Raw	STFT	LogSTFT	CQT	Average
CTC	12.18/ 3.94	10.29/ 3.24	**9.96/ 3.15**	10.59/ 3.32	10.76 / 3.41
PAE	11.65/ 4.82	11.11/ 4.53	11.08/ 4.51	11.10/ 4.53	11.24/ 4.60
Kern	12.10/ 4.57	11.53/ 4.22	11.31 / 4.15	11.29/ 4.18	11.56/ 4.28
Average	11.98/ 4.44	10.98/ 4.00	10.78/ 3.94	10.99/ 4.01	

Table 4
Note-level transcription errors for different inputs and outputs. Best result is highlighted in bold text.

	E_p	E_m (+rests)	E_e (+rests)	E_d (+rests)
Raw + CTC	0.50	0.61 (0.67)	0.12 (0.13)	1.19 (1.49)
Raw + PAE	1.74	0.70 (0.78)	0.42 (0.40)	2.27 (2.56)
Raw + Kern	1.32	1.23 (1.34)	0.23 (0.22)	1.20 (1.46)
STFT + CTC	0.42	0.32 (0.35)	0.09 (0.09)	0.76 (1.06)
STFT + PAE	1.50	0.48 (0.52)	0.15 (0.14)	1.41 (1.72)
STFT + Kern	1.07	0.84 (0.90)	0.16 (0.15)	0.81 (1.10)
LogSTFT + CTC	0.44	**0.25 (0.28)**	0.09 (0.08)	**0.71 (1.04)**
LogSTFT + PAE	1.49	0.48 (0.53)	0.17 (0.16)	1.36 (1.70)
LogSTFT + Kern	1.12	0.86 (0.93)	0.16 (0.15)	0.82 (1.11)
CQT + CTC	**0.40**	0.38 (0.44)	**0.08 (0.08)**	0.78 (1.08)
CQT + PAE	1.53	0.47 (0.50)	0.15 (0.15)	1.52 (1.83)
CQT + Kern	1.16	0.73 (0.77)	0.15 (0.15)	0.84 (1.16)

Table 5
Transcription errors by category for different inputs and outputs. *Clef*, *KeySig* and *TimeSig* represent the prediction errors of the single word representing clef, key signature and time signature, respectively. *Note* shows the errors in the prediction of note pitch and duration, plus errors of missing and extra notes, whereas *Rest* represents the errors of rest duration, plus errors of missing and extra rests. *Barline* shows the errors in barline location, based on the predicted time signature. Finally, *Tie* accounts for invalid ties in the predicted score, and *Format* represents the formatting errors present in the output representation. Best result is highlighted in bold text.

	Clef	KeySig	TimeSig	Note	Rest	Barline	Tie	Format
Raw + CTC	19.93	13.38	38.07	2.41	5.74	32.27	10.03	0.79
Raw + PAE	21.20	14.53	38.47	5.12	7.07	31.60	5.84	0.38
Raw + Kern	22.53	14.39	38.92	3.96	6.58	33.38	43.93	1.25
STFT + CTC	**19.74**	13.74	37.99	1.58	**4.74**	28.93	11.97	0.51
STFT + PAE	21.91	13.31	38.50	3.53	5.59	30.50	**4.65**	0.38
STFT + Kern	21.02	14.03	39.74	2.88	5.46	34.65	39.59	0.93
LogSTFT + CTC	19.98	**13.26**	37.81	**1.48**	4.93	**28.04**	8.45	0.47
LogSTFT + PAE	21.48	13.59	38.15	3.50	6.08	29.91	4.74	0.35
LogSTFT + Kern	21.41	13.95	39.55	2.95	5.58	31.54	36.04	0.95
CQT + CTC	20.27	13.33	**36.74**	1.65	4.96	32.41	16.95	0.72
CQT + PAE	20.58	14.01	38.09	3.65	5.72	31.19	4.84	**0.33**
CQT + Kern	21.75	14.01	39.74	2.88	5.50	33.96	33.21	0.88

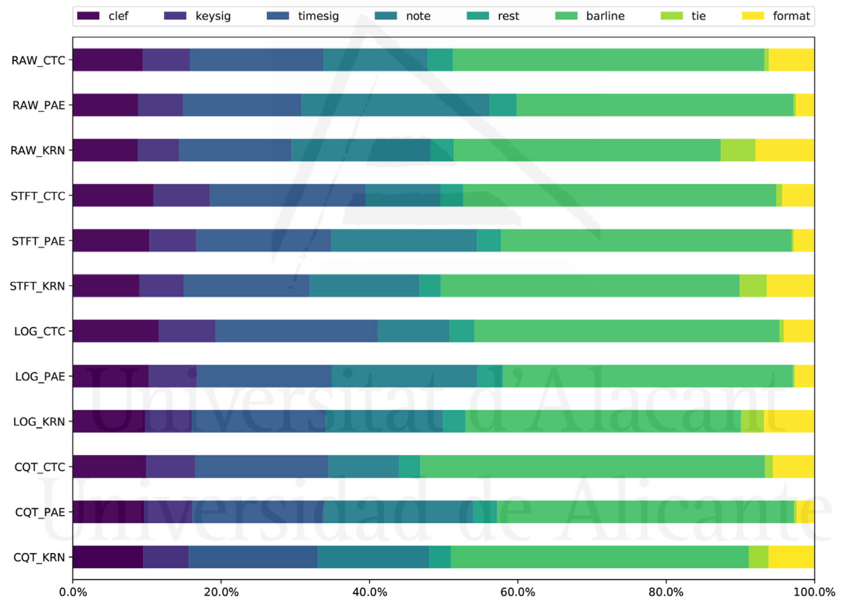


Fig. 6. Distribution of transcription errors by category for each input and output representation.

Table 6
WER and CER of test samples split by instrument type using the best model.

	WER	CER	Samples
Piano	10.20	3.26	20961
Harpichord	11.45	3.50	1508
Organ	12.05	4.04	2020
Strings	9.60	3.06	10293
Winds	11.08	3.65	1938

CTC-friendly), as shown in Table 6. These results reveal small differences in error rates, which suggests that the model can learn to transcribe multiple timbres with little effect on performance. No further comparative analysis can be derived from these results, since instruments are not equally represented in our test set.

7. Conclusions and future work

Although monophonic pitch estimation is considered an almost solved problem, producing a human-readable score from monophonic audio still presents some challenges we are addressing in this work. Our Audio-to-Score task goes one step beyond more traditional AMT systems based on pitch estimation, by also learning other musical information such as note duration, rests, clef, key signature, time signature, barlines, and ties.

In this work, we analyzed different input/output representations to perform notation-level AMT in an end-to-end manner using Convolutional Recurrent Neural Networks. For this, 246,870 incipits from RISM were synthesized using different timbres and random tempos, and the raw audio, STFT, LogSTFT and CQT were extracted to be used as input. To encode the music score in a textual format, PAE, Kern and a designed CTC-friendly language were considered. Different configurations of the CRNN were experimentally tuned for each input and output representation.

We tested this framework for all input and output combinations, performing a thorough error analysis to bring a comparative evaluation. The best results were obtained using logarithmic STFT as input and CTC-friendly as output format, which can be converted with simple symbol manipulations to Kern or PAE, making it a valid language to represent a music score. The comparative evaluation shows that using a proper representation of data can lead to shorter training cycles and lower error rates.

We classified prediction errors based on their musical meaning, i.e., clef, time signature, key signature, note pitch, note duration, rest, barline location, tie, and format errors. We observed that time-related symbols have the highest error rates, including barline location, time signature and note/rest duration, in that specific order.

In general, these results are promising to build a practical application for musicians that performs monophonic A2S music transcription, considering that approximately 10% of the predicted symbols require manual editing to yield the correct score. However, it is still not clear how these results would translate to polyphonic music. AMT for polyphonic music is still an open problem, but as Natural Language Processing thrived just by scaling up end-to-end language models, we believe AMT could experience a similar leap of progress once we procure sufficient amount of data and build music language models that can be trained with very long audio sequences.

Future work includes adding a language model to the decoder, which is expected to boost the performance by ensuring the output is syntactically and gramatically correct, thus making it suitable to render a valid graphic score with a music engraving software like Verovio (Swiss RISM Office, 2017). In addition, we plan to validate this framework with polyphonic audio, as well as to evaluate alternative neural network architectures such as *seq2seq* auto-regressive models with attention mechanisms, which perform better in the analogous speech recognition task by predicting output characters conditioned to those already predicted, while CTC-based models assume statistical independence of output characters. Moreover, auto-regressive models allow output sequences longer than its corresponding input sequences (Sutskever, Vinyals, & Le, 2014), a key advantage for polyphonic music that requires much more characters than monophonic music to output a score, with the same number of audio frames.

CRediT authorship contribution statement

Miguel A. Román: Software, Validation, Data curation, Investigation, Writing - original draft, Visualization. **Antonio Pertusa:** Conceptualization, Supervision, Writing - review & editing. **Jorge Calvo-Zaragoza:** Methodology, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the Spanish "Ministerio de Ciencia e Innovación" through Project HISPAMUS (No. TIN2017-86576-R supported by EU FEDER funds).

References

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A., Jun, B., LeGresley, P., Lin, L., & Zhu, Z. (2015). Deep speech 2: End-to-end speech recognition in english and mandarin. ArXiv e-prints.

Arora, V., & Behera, L. (2015). Multiple F0 estimation and source clustering of polyphonic music audio using PLCA and HMRFs. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 23*, 278–287.

Benetos, E., & Dixon, S. (2013). Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *The Journal of the Acoustical Society of America, 133*, 1727–1741. https://doi.org/10.1121/1.4790351.

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2019). Automatic music transcription: an overview. *IEEE Signal Procesing Magazine, 36*, 20–30.

Brook, B. S. (1965). The simplified plaine and easie code system for notating music: a proposal for international adoption. *Fontes Artis Musicae, 12*, 156–160. http://www.jstor.org/stable/23504707.

Carvalho, R. G. C., & Smaragdis, P. (2017). Towards end-to-end polyphonic music transcription: transforming music audio directly to a score. In *IEEE Workshop for Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE.

Cemgil, A. T., Kappen, H. J., & Barber, D. (2003). Generative model based polyphonic music transcription. In *IEEE workshop on applications of signal processing to audio and acoustics* (pp. 181–184).

Cogliati, A., Temperley, D., & Duan, Z. (2016). Transcribing Human Piano Performances into Music Notation. In *Proc. of the 17th International society for music information retrieval conference, ISMIR 2016, New York, USA*.

Corrêa, D. C., & Rodrigues, F. A. (2016). A survey on symbolic data-based music genre classification. *Expert Systems with Applications, 60*, 190–210. https://doi.org/10.1016/j.eswa.2016.04.008.

de Cheveigné, A., & Kawahara, H. (2002). YIN, A fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America, 111*, 1917–1930. https://doi.org/10.1121/1.1458024.

Gfeller, B., Frank, C., Roblek, D., Sharifi, M., Tagliasacchi, M., & Velimirović, M. (2019). Spice: Self-supervised pitch estimation. arXiv:1910.11664.

Good, M. (2001). MusicXML for notation and analysis. In W. B. Hewlett & E. Selfridge-Field (Eds.), *The virtual score: representation, retrieval, restoration, volume 12 of computing in musicology* (pp. 113–124). MIT Press.

Graves, A. (2008). Supervised Sequence Labelling with recurrent neural networks. Ph.D. thesis Technical University Munich.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *23rd International conference on machine learning international conference on machine learning* (pp. 369–376). ACM.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., & Eck, D. (2018). Onsets and frames: dual-objective piano transcription. In *Proc. of the 19th international society for music information retrieval conference* (pp. 50–57).

Huron, D. (1995). *The humdrum toolkit: reference manual*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on machine learning, ICML 2015, 6–11 July Lille, France* (pp. 448–456).

Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., & Widmer, G. (2016). On the potential of simple framewise approaches to piano transcription. In *17th International conference on music information retrieval*.

Kim, J. W., & Bello, J. P. (2019). Adversarial learning for improved onsets and frames music transcription. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 670–677). http://archives.ismir.net/ismir2019/paper/000081.pdf.

McLeod, A., & Steedman, M. (2018). Evaluating automatic polyphonic music transcription. In *Proceedings of the 19th international society for music information retrieval conference, ISMIR 2018, September 23–27, 2018 Paris, France* (pp. 42–49).

Nakamura, E., Benetos, E., Yoshii, K., & Dixon, S. (2018a). Towards complete polyphonic music transcription: integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE.

Nakamura, E., Benetos, E., Yoshii, K., & Dixon, S. (2018b). Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice hall.

Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with sincnet. In *2018 IEEE spoken language technology workshop (SLT)* (pp. 1021–1028).

RISM. (2017). Répertoire International des Sources Musicales. http://www.rism.info/..

Román, M. A., Pertusa, A., & Calvo-Zaragoza, J. (2018). An end-to-end framework for audio-to-score music transcription on monophonic excerpts. In *Proceedings of the 19th international society for music information retrieval conference, ISMIR 2018, Paris, France*.

Román, M. A., Pertusa, A., & Calvo-Zaragoza, J. (2019). A holistic approach to polyphonic music transcription with neural networks. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 731–737). http://archives.ismir.net/ismir2019/paper/000089.pdf.

Sapp, C. S. (2005). Online database of scores in the humdrum file format. In *Proceedings of the 6th international conference on music information retrieval, London, UK*. . http://ismir2005.ismir.net/proceedings/3123.pdf.

Sapp, C. S. (2013). humextra. https://github.com/craigsapp/humextra.
Schobrun, M. (2005). *The everything reading music book: a step-by-step introduction to understanding music notation and theory. Everything series.* Adams Media.
Schörkhuber, C., & Klapuri, A. (2010). Constant-Q transform toolbox for music processing. In *Proc. of the 7th sound and music computing conference, Barcelona, Spain.*
Shi, B., Bai, X., & Yao, C. (2017a). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*, 2298–2304.
Shi, B., Bai, X., & Yao, C. (2017b). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*, 2298–2304.
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 27th International conference on neural information processing systems – NIPS'14* (Vol. 2, pp. 3104–3112). Cambridge, MA, USA: MIT Press.
Su, L., & Yang, Y. (2015). Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23*, 1600–1612. https://doi.org/10.1109/TASLP.2015.2442411.
Swiss RISM Office. (2017). verovio. https://github.com/rism-ch/verovio.
Mauch, M., & Dixon, S. (2014). PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP 2014). In press.
Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. CoRR, abs/1804.06516. http://arxiv.org/abs/1804.06516. arXiv:1804.06516.
Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin & D. E. Rumelhart (Eds.), *Backpropagation: theory, architecture and applications* (pp. 433–486). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.
Ycart, A., McLeod, A., Benetos, E., & Yoshii, K. (2019a). Blending acoustic and language model predictions for automatic music transcription. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), Proceedings of the *20th International society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 454–461). http://archives.ismir.net/ismir2019/paper/000054.pdf.
Ycart, A., Stoller, D., & Benetos, E. (2019b). A comparative study of neural models for polyphonic music sequence transduction. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 470–477). http://archives.ismir.net/ismir2019/paper/000056.pdf.

Part III

Conclusion

Chapter 6

Concluding remarks

This chapter ends this dissertation. It includes a summary of the research carried out, a brief discussion based on its results, and some of the main opened lines of future work.

6.1 Summary

This thesis presents a new approach in the area of AMT, defines the Audio-to-Score (A2S) task and proposes a framework to perform A2S in an end-to-end manner, thanks to the expressive power of deep neural networks. This framework goes one step beyond most conventional AMT systems, which aim at predicting notes in a piano-roll notation. The main advantages of our approach compared to previous methods are:

1. The output of our method is a music score that can be directly interpreted by musicians or analyzed by musicologists.
2. The end-to-end nature avoids errors from one stage cascading to other stages.
3. Annotations of alignment information between audio frames and music scores are not required since they are implicitly learned.
4. A music language model is learned by the end-to-end approach, which also helps on reducing transcription errors globally.

6.2 Discussion

Results on monophonic audio show that the source of most errors is related to the metric information of the music score such as note and rest duration, time signature estimation, and barline location. This makes sense since information is not encoded explicitly in the audio signal frames and must be inferred taking into account the learned music language model and the musical context extracted from the full-length audio.

The input representation that led to the best performance was the STFT with log-spaced bins aligned with music octaves. The best output representation was the one we specifically created to ease CTC-decoding, as it was intended. We also verified that the framework is capable of performing well with a fixed number of polyphonic voices, dealing with every A2S challenge in the time domain (e.g. note quantization) and in the frequency domain (e.g. harmonic overlapping).

Although the A2S formulation we present in this dissertation is still far from being solved, especially in the case of polyphonic music, we believe the results of our experiments open a new path of research in the area of AMT. Nonetheless, there are two important factors that currently hamper A2S progress:

1. Although there exist many audio files with their corresponding digital scores, the requirements of the framework in its current formulation (i.e., short fragments of polyphonic real audio and their matching music score excerpts) limit the amount of data that can be used for training.

2. There are no standard polyphonic evaluation metrics to measure A2S performance based on the musical outcome rather than on the edit distance of predicted and expected output sequences. Proper metrics would also allow an adequate comparison of different methods.

Similarly to how Natural Language Processing (NLP) has experimented a giant leap over the past few years thanks to bigger models like GPT-3 [4], the A2S task could also benefit as more computing power becomes available. Larger neural networks can process longer sequences, which allows training directly with audio recordings and their corresponding digital music scores in the public domain. Collecting data by this approach requires less effort and can increase the size of AMT training data by several orders of magnitude. In summary, by increasing both the size of neural network models and the amount of training data, we would be able to learn better music language models that might be the key to overcome some of the intrinsic challenges of AMT.

6.3 Future work

Besides working towards overcoming the aforementioned limiting factors of A2S progress, there are other ways in which this work can be continued:

1. Adding a language model to the decoder, which will boost performance by ensuring the output is syntactically and grammatically correct. This language model can be trained in an unsupervised way by using large datasets of existing digital music scores.

2. Assessing auto-regressive models with attention mechanisms like the Transformer architecture [16], to remove the CTC limitation of having output sequences no longer than the number of input frames. This architecture is also capable of learning better implicit language models than recurrent neural networks for similar tasks.

3. Augmenting data to improve the robustness of the models, by applying multiple transformations to the audio, like noise addition, pitch shifting, dynamic range variation, audio synthesis with alternative sound fonts per instrument, etc.

4. Creating a new dataset for the A2S task with real audio, after splitting existing recordings automatically with a score following tool.

Bibliography

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *ArXiv e-prints*, 12 2015.

[2] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic Music Transcription: An overview. *IEEE Signal Procesing Magazine*, 36(1):20–30, 2019.

[3] Barry S. Brook. The simplified "plaine and easie code system" for notating music : A proposal for international adoption. *Fontes Artis Musicae*, 12(2/3):156–160, 1965.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[5] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *23rd International Conference on Machine Learning*, International Conference on Machine Learning, pages 369–376. ACM, 2006.

[6] David Huron. **kern representation, 1998.

[7] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2018.

[8] M. Ravanelli and Y. Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028, 2018.

[9] RISM Organization. Répertoire International des Sources Musicales, 2015.

[10] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 34–41, Paris, France, September 2018. ISMIR.

[11] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A Holistic Approach to Polyphonic Music Transcription with Neural Networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 731–737, Delft, The Netherlands, November 2019. ISMIR.

[12] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. Data representations for audio-to-score monophonic music transcription. *Expert Systems with Applications*, 162:113769, 2020.

[13] Craig S. Sapp. humdrum-data. `https://github.com/humdrum-tools/humdrum-data.git`, 2014.

[14] Marc Schobrun. *The Everything Reading Music Book: A Step-By-Step Introduction to Understanding Music Notation And Theory*. Everything series. Adams Media, 2005.

[15] Christian Schörkhuber and Anssi Klapuri. Constant-Q transform toolbox for music processing. In *Proc. of the 7th Sound and Music Computing Conference, Barcelona, Spain*, July 2010.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.