Audio2Hero: Using Transformers for Efficient Symbolic Music Generation

Matthew Banaag
Princeton University
Princeton, NJ
matthew.banaag@princeton.edu

Tim Gubski
Princeton University
Princeton, NJ
timgubski@princeton.edu

Abstract

Love Guitar Hero but don't have game files for your favorite song? Audio2Hero is an encoder-decoder transformer finetuned to generate Guitar Hero *chart* files (files which describe the level setup including buttons and note timings) from *any* audio! Beyond Guitar Hero, little prior work has been done on analyzing the feasibility of using transfer learning to generate new styles of music based on a learned ability to recognize rhythm. Through training Audio2Hero, this paper demonstrates that accuracy is not truly indicative of symbolic music sequence generation quality, and we propose alternatives such as *rhythmic similarity* (RS). Further, because Guitar Hero chart files encode rhythm separately from notes as strum events, Audio2Hero demonstrates the performance boosts resulting from explicit audio encoding, providing this as a potential future direction for training music transcription and generation models. Additionally, we wrote our own MIDI tokenizer, so hopefully no one else will ever have to spend two days forgetting if token ID 137 is a 3 note rest or not ever again.

1 Introduction

Since November 2005 people all around the world have loved playing the popular game Guitar Hero, and to this day a large community exists that loves creating and playing custom *charts* for the PC version of this game, Clone Hero. A dream of all Clone Hero fans is for a universal auto charter to exist, one that can take in the mp3 audio file for any song, and fully generate a fun-to-play rhythmically accurate chart for the song. Previous attempts have relied on 1 to 1 note mappings of manually converted MIDI versions of popular songs, or increasing or decreasing the difficulty of an existing chart by adding and removing notes. While a universal auto charter was previously thought to be impossible, the advent of Transformers and recent developments in using transformers to encode and decode music have made this dream ever closer to becoming a reality.

Figure 1: Clone Hero

In the past few years transformer models that are capable of encoding audio into MIDI files have become more prevalent. New techniques for tokenization of continuous audio and discretized music formats have made this possible, however existing methods are poorly documented and the metrics for evaluating their performance are not designed for audio and don't transfer well to this use case.

Our project builds upon one such model performing audio to MIDI conversions called Pop2Piano, built upon the encoder-decoder T5 transformer, and cross trains the model to generate rhythmically accurate charts for the game Clone Hero.

2 Prior Work

Stylized music generation can be reimagined as an Automated Music Transcription (AMT) task. AMT is a task which aims to discretely estimate note information from raw audio, typically by producing a MIDI representation, which is a file format that symbolically represents music. Recently, research has aimed to train neural networks to learn how to

predict this symbolic representation of raw audio, typically fed into these networks in the form of a spectrogram, which is a frequency-encoding of time-series data. Early research from Google Brain (Hawthorne et al., 2018) first attempted to solve AMT by transforming it into a more classical machine learning task of predicting onset and offset timings for notes alone using bidirectional LSTMs. While this was a great step forward, the model still lacked in performance. Other attempts were made at using different models to improve upon performance. NYU researchers attempted to use adversarial learning (Kim and Bello, 2019), however their model largely lacked in generalization ability. Some generalization issues have arisen from audio misalignment in data while training. Other researchers attempted to deal with the generalization problem in audio misalignment by training with regression on onset and offset times (Kong et al., 2021), though they still are computationally inefficient and lack the ability to explicitly represent musical style due to still running on GRUs.

In recent years, however, attention-based learning methods with transformers have revolutionized the state-of-the-art approach to AMT. Recently, researchers from MIT (Gong et al., 2021) created the Audio Spectrogram Transformer (AST), which reimagined the AMT problem as feeding in audio through its frequency representation as a spectrogram into a Vision Transformer (ViT) (Dosovitskiy et al., 2021). New literature has further modified the attention mechanism specifically for the task of generating symbolic music sequences using fine- and course-grained attention inspired by the repetitive structure of music (Yu et al., 2022), with the sparser computation allowing for the generation of sequences three times longer than was previously possible.

Most importantly for stylized music generation, recent work has shown great abilities for transformers to explicitly learn musical style as an encoded input to inform symbolic music generation. Using autoencoders, researchers from Stanford and Google Brain (Choi et al., 2020) improved performance of transformer-based generative modeling by building a global representation of music style during training. Especially interesting, prior work has shown that homophonic music (music with a predominant melody and harmony part) with user-assigned styles can be easily learned for music generation tasks (Lu and Su, 2018).

One prior paper by researchers from Seoul National University and Rebellions Inc. explored the usage of this stylized music encoding to produce piano pop covers from synchronized audio datasets with a model called Pop2Piano (Choi and Lee, 2023). Pop2Piano is an audio spectrogram transformer built on a modified version of the T5-small encoder-decoder (Raffel et al., 2020), taking in an audio spectrogram and a style token to produce an output MIDI sequence.

Although much prior work has been done on learning explicit representations of musical style for generation, there is a gap in work that has been done on analyzing the feasibility of transferring models trained on specific musical styles to adopt to generating new styles of music. Prior work has not been done on explicitly measuring the ability of a model to generalize its rhythmic ability to sync notes from raw audio and its symbolic representation. Inspired by the musical style encodings of prior research and the Pop2Piano paper, this paper aims to finentune the Pop2Piano model for generating Clone Hero MIDI chart files, which serves as a litmus test for the ability of models to exhibit transfer learning across different genres.

3 Methods

3.1 Overview

MIDI files are a popular method for encoding music, encoding the song into discrete notes with unique instruments, start/end times, velocities, and volumes. This method of encoding is particularly useful for our application as Clone Hero charts use the MIDI format to store what buttons on the controller need to be pressed at what times. Because of their discrete nature MIDI files are easy to tokenize as at a given time step you can either turn a note on or off.

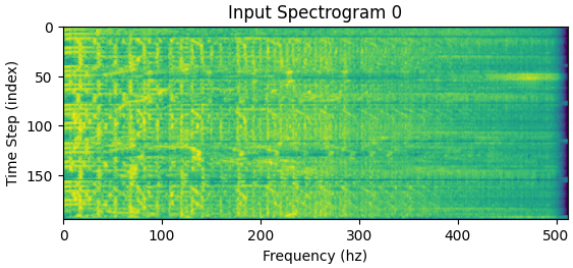

Figure 2: Log-Mel Spectrogram

Encoding continuous audio is a much more difficult problem, but there have been methods that

have made it much easier. One such method is log-mel spectrogram encoding, which samples the audio file and converts it to a time vs frequency representation of the original audio. This representation can be visualized as an image which allows us to utilize developments in image transformers to encode and process the image.

For synchronization the method of beatstep extraction is used to approximately determine the time step of every 8th note in the original audio. Because we dont know the original tempo of the song these beatsteps are calculated using the Rhythm Extractor 2013 algorithm provided by the python library Essentia (Bogdanov et al., 2013). These beatsteps will be used to ensure that the generated MIDI is on beat with the original audio.

3.2 Experimental Setup

To create an autocharter for Clone Hero we wanted to cross-train the Pop2Piano model, which was trained on 3000 hours of piano covers, using a dataset of Clone Hero charts such that it would retain its previous understanding of how beats work and how to rhythmically flow between notes while learning the smaller 5 button vocabulary of Clone Hero and mimicking the chart style of previous games.

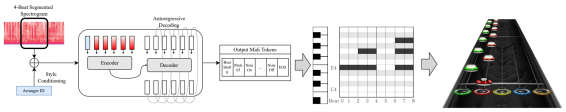

Figure 3: Audio2Hero Architecture

3.2.1 Data Collection

The Clone Hero community has collected thousands of charted songs both ripped from previous Guitar Hero and Rock Band games, as well as custom made charts. These are all publicly available for free and provide several .ogg files storing the audio for the backtrack and various instruments as well as a notes.midi file which stored the chart information for every instrument and difficulty that the song offered. This is possible because the MIDI file represents each unique instrument as its own track and each difficulty uses a non-overlapping range of pitches. A visualization of this MIDI is shown in Figure 4.

3.2.2 Preprocessing

We collected over 1000 charted songs totaling 50 hours of high-quality training data. To make our

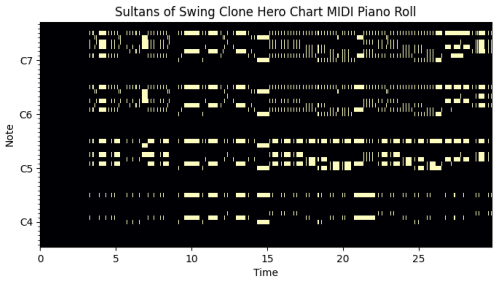

Figure 4: Clone Hero Chart MIDI

data compatible with the Pop2Piano pipeline we had to perform a few preprocessing steps. Firstly, while generating a chart for all 4 difficulties at the same time may be possible, we decided to limit the scope of this project to only generate medium difficulty songs. To facilitate this we filtered through all of the chart files and removed all of the notes that didnt correspond to a medium-difficulty chart. We also removed any instruments that weren't lead guitar, removed any tempo changes, and the events track which is used to signify song sections to Clone Hero but provides no musical value.

Our collected dataset also stored the audio is separate .ogg files, one for each instrument and the backing track. This is done so that if you mess up in the game the sound for your instrument stops playing, but this feature isn't strictly necessary. Because we want our model to work on an original audio file we merged all of these parts into a single .ogg file that contains the original song.

3.2.3 Tokenization

Tokenizing the ground truth MIDI files required us to create our own custom tokenization pipeline since the existing Pop2Piano tokenizer was only able to process very specifically encoded MIDI examples with a maximum length of 8 beats and with note start/end times being represented by the eighth note beat index as opposed to time. To handle any MIDI file without length or format restrictions our generalized tokenizer crops the input MIDI into 8 beat parts, and quantizes each notes start and end time to the nearest beat step extracted from the original audio. Additionally, to maintain a small vocabulary, each beat index was made to be relative to the start of the 8 beat sequence. The vocabulary consisted of:

1. **Buttons and Strums**: Representations of which button to turn on or off (including strum), 5 possible values.

2. **Velocity**: 2 possible values, representing whether to turn the note on or off.

3. **Beat Shift Tokens**: Representations of how long to wait between the next note change event, 8 possible values.

4. **EOS, Pad**: 2 possible values, represent sequence generation end and padding to maximum sequence generation length.

After creating our own more generalizable tokenizer we discovered a fairly fundamental problem with the existing MIDI encoder that caused the output of the encoder to be inconsistent with the sequences generated by the model and what the decoder was expecting. The original encoder represented when a note should be turned on or off by using a token that would encode the absolute index of the beat in the measure. However, the decoder operated by representing the start of a note using a token that encoded the beat delay since the last note start or end. To solve this inconsistency we rebuilt the encoder with a fix that would reconcile the encoding format with what the decoder and model expected. Its not clear why this was the case, but it is possible that this was how the model initially worked and down the line an optimization was made to switch from absolute to relative encoding and the change was only added to the decoder, as that is what most people would be using.

3.2.4 Training

Having finally figured out all of the processing steps for the data, we were finally ready to fine-tune the model to generate Clone Hero charts! To simplify training we decided to train on a subset of our dataset consisting of 480k tokens, and due to memory limitations, we decided to fine-tune only the decoder of the model. We froze all of the model parameters except for the last fully connected layer of the decoder, the final layer normalization, and the unembedding layer weights. To further reduce computation time we cache the processed ground truth MIDIs to reuse them for future epochs, as generating these took several seconds for every song. We used Cross Entropy Loss for the loss function, and we used the AdaFactor optimizer with relative step, warmup initialization, without scale parameter. With this parametrization, we finetuned the model to produce an output similar to the Clone Hero charts that we input as training data, given raw audio files.

4 Results

4.1 Benchmarks

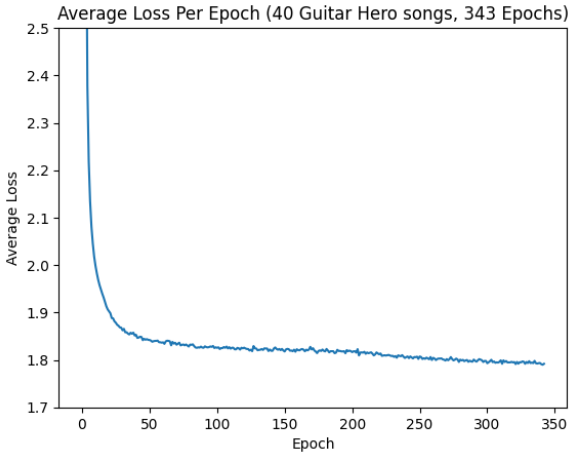

Figure 5: Training Cross Entropy Loss

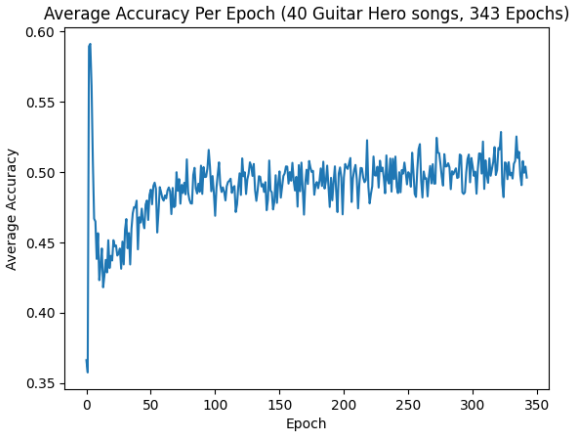

Figure 6: Training Accuracy

After training for over 300 epochs, we achieved a cross entropy loss of 1.79 with a best average accuracy (per epoch) of 59%. Prior work has shown that cross entropy works well as an objective loss function for generative music models (Garcia-Valencia, 2020) when used on melodies, which Audio2Hero is consistent with since the Clone Hero chart file itself is a melody. Interestingly, while the cross-entropy loss rapidly decreases until around epoch 40, where it decreases much more slowly, most accuracy gains began in this domain beyond 40 epochs. Presumably, this is because the model learns to realign its unembedding layer and final fully-connected layer in the modified T5-small to properly learn the Clone Hero vocabulary. As shown in prior work as well, randomness in music generation tends to happen relatively fast, though it

requires many more training iterations to develop the longer-term context that makes the song become a reasonable approximation of the original audio (Garcia-Valencia, 2020). Though training accuracy appears low, we find that this is indicative of the model being pretrained to encode pitch tokens as actual notes, rather than the notes which correspond to different Clone Hero buttons (for example, the Green Button on Medium difficulty is represented as the note C5 in the chart file (gui)).

In fact, the model's generated outputs have higher fidelity when analyzed with regard to other metrics. For example, testing with a generated transcription of the song *Dire Straits - Sultans of Swing*, 99% of the final tokens are valid Clone Hero chart file tokens. Seeing that the model converges to this small vocabulary of Clone Hero chart file tokens, one can see intuitively accuracy is not a great metric for this task given that much of the ground truth may be similar due to this limited vocabulary. Looking at these scored representations of the ground truth chart files (Figures 7 and 8), we see that many of rhythms between these two completely different songs are in fact similar (lots of long notes overlapping with similar buttons).

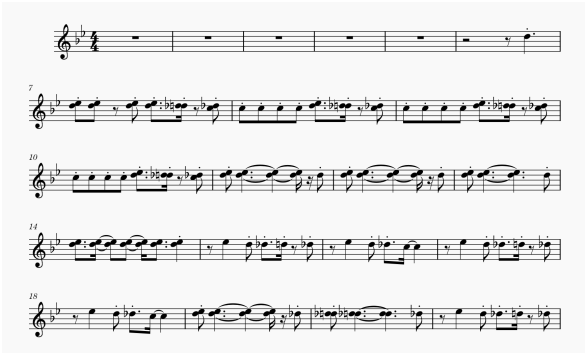

Figure 7: *Ace of Spades vs. Groundhog* Ground Truth MIDI Chart File

Using normalized dot-product similarity, we actually find that there is even only a mild correlation between the original raw audio and the ground truth MIDI chart files, noting that 50% of ground truth MIDI chart files are most correlated to their source raw audio (see Figure 9). To do this, we converted the ground truth MIDI chart files for 20 Clone Hero songs in the training dataset to time-series single-channel audio, and compared them with the time-series representation of the corresponding raw audio files. Note that this metric would solely account for rhythmic similarity, since the pitch information encoded from MIDI chart

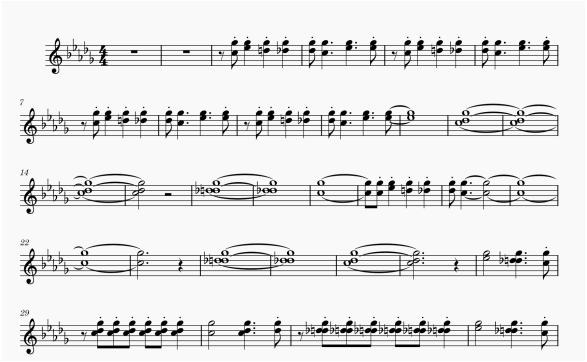

Figure 8: *Tom Petty - Runnin' Down a Dream* Ground Truth MIDI Chart File

files actually are encodings of the buttons rather than raw pitch.

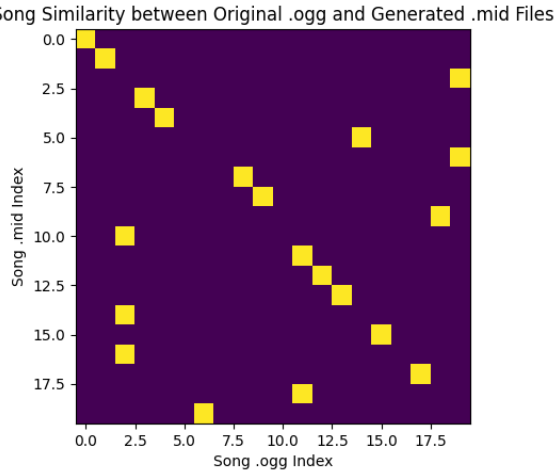

Figure 9: Song Similarity (most similar pair shown for each)

Better for measuring Audio2Hero's generation, we define the metric *rhythmic similarity* to measure how good a generation is at aligning rhythm tokens with note tokens. We define *aligned notes* to be a non-strum note that has the same onset time as a strum note. In the case of Clone Hero notes, rhythmic similarity (RS) measures

$$RS = \frac{\text{\# of aligned notes}}{\text{\# of non-strum notes}}$$

Measuring RS on *Dire Straits - Sultans of Swing*, we find that $RS = 100\%$, or in other words every note is matched to a strum. This rather interestingly suggests that a pre-trained music generation model that is fed inputs containing an explicit rhythm token is highly capable of aligning rhythms with target musical sequences (in this case, with

the melody). Intuitively, this makes sense since the model must learn implicit representations of rhythms through its generation of relative beat shift tokens. This suggests that in future, aligning symbolic musical sequences to audio may be improved by explictly passing in a rhythm token in addition to the direct MIDI tokenization when learning alignments. Let us analyze this rhythmic similarity from a qualitative perspective as well now.

4.1.1 Qualitative Observations

After 300 epochs of training, we can qualitatively observe that the model is outputting MIDIs that look far more similar to the Clone Hero charts that we provided. Most significantly the model has fully learned the vocabulary of the charts, almost entirely generating valid medium-difficulty notes. As stated in the previous section, the model learned that every note requires a strum note to be placed on pitch 78 which is proof that it is learning to generate valid charts.

Through the training, we observe that the model seems to be picking up the rhythmic flow of the song very well, and has a strumming pattern very similar to the Clone Hero charts style.

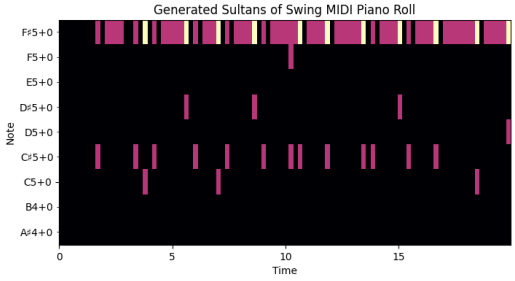

Figure 10: Generated MIDI File

However the model occasionally will add a strum token without a corresponding button token, which when imported directly into clone hero can sometimes result in a more sparse chart with gaps in a note sequence. We hypothesize that since all of the charts use the same pitch to represent strums, it was easier for the model to learn the significance of this token and to place it at appropriate places in the rhythm, whereas it was more difficult to assign a button press to every strum, since by the nature of having a smaller vocabulary a given note in the song no longer always corresponds to a specific button on the controller.

4.1.2 Post-processing

To fix the over-strumming problem we added a post-processing step that would randomly assign a button press to each strum without a button. In the post-processing step we also needed to do some small cleanup steps, like removing any notes that were generated outside the medium difficulty vocabulary, and adding some metadata to the MIDI which would allow Clone Hero to process it, like renaming the MIDI track to "PART GUITAR". We also needed to generate a song.ini file which gives Clone Hero information about the song title, artist, and duration. Our MIDI file post-processing is shown in the following figure.

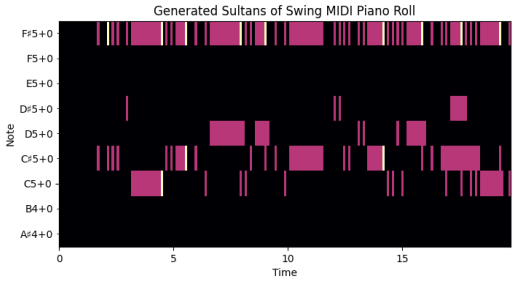

Figure 11: Post-Processed MIDI File

5 Next Steps

Our immediate goal is to build out a website with a Flask server running our generation pipeline so we can share our auto charter with the Clone Hero community and discord server. Hopefully, we get some good feedback on the charter's performance on a wide variety of songs through this and get more of an intuition about what needs to be fixed or improved for future versions. In the meantime, we have a few ideas for how to improve our model and generation pipeline.

Our dataset contained over 480k tokens of Clone Hero charts, mostly limited to rock music as that was the style of songs in the Guitar Hero games. Expanding our dataset with more songs and a greater variety of genres could make our model provide better charts for a wider range of music styles. We would like to train on a larger compiled dataset and assess the effect of significantly increasing our training time.

Currently, we are batching an entire song as about 100 sequences, and computing the loss on the entire song batch. While this does help the model learn how to represent a song in a more local manner, a potential way to improve model performance

would be to instead generate batches from random subsampling of a diverse set of songs. This may potentially help to improve generalization.

Due to computational constraints, we were only able to fine-tune the final few layers of the decoder. In the future if we obtain access to more computation power it would be interesting to see how fine-tuning more of the model parameters would affect its performance. Alternatively, we would also like to see if training the entire model from scratch as opposed to fine-tuning a pre-trained model could provide better results.

6 Usage

Our model can be accessed on HuggingFace at https://huggingface.co/mb6611/audio2hero/, and our GitHub repository can be found at https://github.com/mb6611/484-clonehero. A link to the demo can be found here: https://youtu.be/y23ZDX2WFg0. The instructions for uploading custom songs for CloneHero(clo) can be reference in the documentation here: https://wiki.clonehero.net/books/clone-hero-manual.

7 Conclusion

In this paper, we describe a general process to convert raw audio files to Guitar Hero chart files using Audio2Hero, a finetuned encoder-decoder transformer. Through the process, we discover that accuracy is not necessarily indicative of symbolic music generation quality, and may be better replaced by rhythmic-based similarity metrics like our proposed *rhythmic similarity* metric. Further, we discover that music generation models transfer knowledge of rhythm through finetuning well (enough to generate some sick riffs on Guitar Hero), and their training may benefit from an explicit encoding of rhythm separate from the actual MIDI tokenization.

And finally as a short addendum to the learnings of this paper, we propose that (after hours of wrangling through existing MIDI tokenization pipelines while heavily caffeinated) a modernized general MIDI tokenizer standard should be developed to speed up research in the field.

References

Clonehero documentation. https://wiki.clonehero.net/books/clone-hero-manual. Accessed: 2024-05-08.

Guitargame chartformats. https://github.com/TheNathannator/GuitarGame_ChartFormats/tree/main. Accessed: 2024-05-08.

Dmitry Bogdanov, N Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, Jose Zapata, and Xavier Serra. 2013. Essentia: an audio analysis library for music information retrieval.

Jongho Choi and Kyogu Lee. 2023. Pop2piano : Pop audio-based piano cover generation.

Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel. 2020. Encoding musical style with transformer autoencoders.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

Sebastian Garcia-Valencia. 2020. Cross entropy as objective function for music generative models.

Yuan Gong, Yu-An Chung, and James Glass. 2021. Ast: Audio spectrogram transformer.

Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. 2018. Onsets and frames: Dual-objective piano transcription.

Jong Wook Kim and Juan Pablo Bello. 2019. Adversarial learning for improved onsets and frames music transcription.

Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. 2021. High-resolution piano transcription with pedals by regressing onset and offset times.

Wei Tsung Lu and Li Su. 2018. Transferring the style of homophonic music using recurrent neural networks and autoregressive model. In *International Society for Music Information Retrieval Conference*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. 2022. Museformer: Transformer with fine- and coarse-grained attention for music generation.