Genre-specific Music Synthesizer Using GANs and LSTMs

Team: Mariana Luna, Claes Sjoborg, Max Speno **Project Mentor TA:** Benedict

1) Abstract

For this project, we study the problem of generating new melodies in three different genres (classical, rock and jazz) based on a training dataset containing melodies belonging to each genre. Two kinds of models were used: an LSTM (Long Short Term Memory Neural Network) and a GAN (Generative Adversarial Network). Our primary contributions were in application, as neither of the models had been used for this genre-specific and multi-instrumental melodies, and analysis, which include both subjective and objective metrics. All files and outputs can be seen on our github repository[1]. The best-sounding music produced by our models can be listened to on SoundCloud[2].

2) Introduction

The motivation behind this project was to explore the "creative" side of Machine Learning by using techniques that would create or generate new samples that were not present in the training data. In particular, we wanted to explore the models' creative ability over different genres, and observe whether a noticeable difference in performance between the genres could be evaluated using different techniques on similarly-sized datasets. One of the main questions we had posed at the beginning of the semester was whether we could find out how "replaceable" artists are. Most of the conversation around automation is generally directed towards non-creative jobs. In this project, we seeked to understand how feasible it would be to one day have these algorithms generating previously unseen content.

To accomplish these tasks, we compared and adapted existing model architectures for our specific genres, and observed the generative abilities of different models for each genre. We used existing model architectures such as a GAN and an LSTM. The three genres for which we produced music and analyzed the results were jazz, rock and classical music. Our inputs for our models were 100 songs of each genre that were 30 seconds long. The LSTM model generates 20 seconds of a melody after having been inputted the first portion of a melody. By contrast, MuseGAN generates completely new songs that are meant to resemble the training dataset.

3) Background

LSTM for music creation

A Long Short-Term Memory (LSTM) network is a type of Recurrent Neural Network that uses gradient descent to learn. LSTM networks have feedback connections which enables them to retain useful information about previous data in the sequence. As a result, LSTMs are able to recognize and encode long-term patterns through distinct gating mechanisms which makes them useful for problems like this where the network must remember information for longer periods of time. We followed the post "How to Generate Music using a LSTM Neural Network in

[1] Github Repository: https://github.com/marilunazz/CIS519-Final-Project
[2] SoundCloud: https://soundcloud.com/user-870062911

Keras'' from Towards Data Science[3], which is implemented in Python using the Keras library. Their training data consists of piano music from 'Final Fantasy' soundtracks with very distinct melodies. To calculate the loss for each iteration in training they use categorical cross entropy as each of the outputs belongs to a single class and there are more than two classes to work as they work with 352 distinct notes and chords. As stated before, this implementation is trained on solely one genre and one instrument. We trained three LSTMs using Jazz, Rock and Classic Kaggle datasets. These datasets contain melodies of 100 different melodies performed with multiple instruments.

MuseGan

The second model we worked with is a Generative Adversarial Network we based on another article in Towards Data Science[4]. A GAN is trained to generate new samples different from the dataset it was trained on. Any GAN is composed of at least of two parts:

1. A discriminator that learns to differentiate one class from another. The discriminator is essentially a classifier that is supposed to know when the input has been produced by the generator and when it is part of the training set.
2. A generator that learns to "trick" the discriminator. If it is not successful, it learns from its mistakes and readapts. The generator input is a randomly generated vector, and the output is something that is supposed to resemble the dataset so that it can trick the discriminator. In the case of MuseGAN, instead of trying to replicate melodies directly, the songs are first transformed into images using the functions midi2image and image2midi[5]. The images are then fed to the GAN, which attempts to replicate them. MuseGAN is intended to generate new melodies that are played solely using the piano.

4) Summary of Our Contributions

1. **Contribution in Application:**

Both models were originally intended for piano-only music. Our models have been trained on different genres (jazz, rock and classical), all of which are played with multiple instruments, and then generate images not present in the dataset that are then transformed into piano-only melodies.

2. **Contribution in Analysis:**

We have compared the performance of each model with respect to each other as well the genre they were trained on. We evaluated the resemblance of the music produced with respect to the genre they belong to. This will help us identify whether the complexity of the input music has a relationship to the performance of each model. To accomplish this, we used both a subjective and an objective approach.

5) Detailed Description of Contributions

i. Using LSTM to generate new music based on genre

[3] "LSTM Networks | A detailed Explanation." Towards Data Science, 21, Oct. 2021, https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[4] "MuseGAN: Using GANs to generate original Music". Towards Data Science, 30th Jan 2021 *https://towardsdatascience.com/bachgan-using-gans-to-generate-original-baroque-music-10c521d39e52*

[5] Midi2Image, Image2Midi. Matthigatti. 22 October, 2021 *https://github.com/mathigatti/midi2img*

We initially trained our three LSTM models on the 100 songs pertaining to each genre. This original LSTM structure had a batch size of 512, 3 fully connected layers and trained on 5 epochs. This initial result gave us simple outputs where the models outputted a singular note being repeated. Although these results were not ideal, there is evidence that the LSTM was distinguishing the different genres as the notes were distinct. These sounds can be listened to on the SoundCloud under “LSTM” and they are suffixed with “FirstIter”.

For our next iteration we tried overfitting on smaller datasets per genre. This meant training on 10 songs per genre and increasing the epochs. There was a great improvement but not enough to distinguish an actual melody pertaining to the genre. All three of these generated tracks are a mixture of chords and notes which was a big improvement from only one note being played. These sounds can be listened to on the SoundCloud under “LSTM” and they are suffixed with “SecondIter”.

In the final iteration the hyperparameters were adjusted. We specified a custom batch size of 256 which helps reduce memory usage and 2 fully connected layers instead of 3, with 256 units each in order to train our model faster. Additionally we trained the model on purely piano music for each genre, each 30 seconds long, which were downloaded from distinct websites[6]. We hypothesized that the large amount of instruments in the original tracks were confusing the LSTM in analyzing which chord or note was being played as there was too much going on. Interestingly enough, the change in architecture and input files vastly improved the generated outputs on qualitative and quantitative notes. These sounds can be listened to on the SoundCloud under “LSTM” and they are suffixed with “Final”.

To validate our final model's iteration we have provided two samples for each genre that we can analyze in a qualitative and quantitative way. In our final iteration we managed to create some cohesive music for each genre. Whilst the melody didn’t necessarily match the genre closely, it showed huge improvement from initial efforts and with further training and larger computing power we think we could create pleasing melodies.

We initially hypothesized that the easiest genre to distinguish would be Classical music as it contains many regular patterns, the music is cyclic, and the chords are played at a regular tempo. In jazz, where a variety of instruments play more sporadically and there are not a lot of patterns, we thought the model would not predict this well. On the Classical music final sample we can hear an attempt at an C major and A minor scale, along with a variety of chords and the repetition of G and B. The repetition inside of this track indicates some sort of learning, especially C major scale which is very common in the songs from the Romantic Classical period (Bach, Mozart, Chopin) meaning that the model was able to understand some parts of the structure of classical music. For our final Jazz output, there were several distinguishing features that aligned with the genre. Interestingly, from a musical perspective, our model outputted a diminished seventh for the Jazz melody which is very important as this is a musical trope that is only present in the Jazz genre specifically. In this way, our output for the Jazz genre is distinguished from other genres. With our final Rock output there were no clear recognizable patterns that indicated the Rock genre.

In order to evaluate the music samples quantitatively we accessed the categorical cross-entropy loss produced in training by the model and compared the second and third iterations of our models with respect to their losses as seen below in Table 1 in the Appendix.

[6] Music Sources: http://www.piano-midi.de/, https://bushgrafts.com/midi/, https://www.midiworld.com/search/?q=rock

The categorical cross-entropy losses displayed above showcase how training on purely piano music for that genre affected the LSTMs learning. Specifically one can see how the most improved model was classical- this is directly related to the audio it produced as one can hear the attempt at C major and A minor scales with some clear repetition of the music which is part of the structure of a typical classical piano piece. Overall, we can see as we increased the complexity of our model and decreased the amount of data in order to overfit, our cross-entropy loss decreased across all genres.

ii. Using MuseGAN to generate new music based on genre

One of the main concerns of the project was that music is a complex process, and with a limited dataset and computational power it might be difficult to be able to create and train a full synthesizer. MuseGAN attempts to solve this complication by working solely with images. Both the discriminator and generator are trained on images that indicate the note that is being played. The methodology will be explained later on. See Figure 1 in the Appendix to see an example of an image representation of a song.

The dataset we trained the models on had 100 melodies of each genre. The models we have composed were trained on three genres: classical music, jazz, and rock.

We prepared three kinds of models, each trained on the three genres with the purpose of generating new music. We found two functions (midi2image and image2midi) that are able to transform a black and white image into a piano song in midi format, and another that does the reverse. To see an example of these images, see Figure 2 in the Appendix

1. The first model we prepared sought to take advantage of the fact that these images shown in Figure 2 were already made available to us. We wanted to explore whether inputting these images in black and white into our GAN would yield music that is identifiable to the genre. The structure of the model is represented by Figure 3 in the Appendix. The best melody for each of the genres can be found on soundcloud[7] under "***Attempt-1***". The code can be found under the same name on Github.
2. The results of the *first model* were not satisfactory (see qualitative analysis later on). We hypothesized that the problem was with the inputs, which is why we decided to train and adapt a second bundle of models where each song (which contained several instruments) would be converted to an image using a function intended for piano melodies. We wanted to explore whether the piano outputs would still be able to resemble the genre they came from. The structure of this iteration can be seen in Figure 4 of the Appendix. The best samples produced by the second set of models (labeled as "***Attempt-2***") can be found on Soundcloud. The code can be found under the same name on Github. An example of a successful picture produced by the generator can be found on Figure 5 in the Appendix.
3. The results for the *second model* were more satisfactory (see qualitative analysis below). However, one of our main concerns was that the dataset was not large enough as each model was trained on 300 images. Our third set of models attempted to overfit slightly more on the dataset. Several attempts were made to accomplish this: we reduced or eliminated the dropout layers in the models, we increased the learning rate of the models, we experimented with reducing the leaky ReLU slope, reducing momentum in the Adam

[7] Soundcloud for MuseGAN outputs:
https://soundcloud.com/user-870062911/sets/gan?utm_source=clipboard&utm_medium=text&utm_campaign=social_sharing

optimizer, adding fully connected layers in the discriminator and generator as well as removing them. All attempts and outputs are saved under “Third Attempt” on Github. One of the main challenges for this set of models was to attempt to overfit without actually increasing computation time. Most of these attempts were unsuccessful, as they produced images akin to Figure 5 in the Appendix. All failed outputs and model design attempts can be found under “***Third Attempt***” on Github. The only model that was able to generate sensible results was the rock model, whose outputs can also be found on Github. The reason why these attempts were unsuccessful was that when working with GANs, one needs to balance the efficiency of the discriminator and the generator. If the discriminator gets too good at identifying the input, the generator will not be able to readapt. Likewise, if the discriminator is not a good classifier, the outputs created by the generator will also suffer. With a dataset of this size, where each image is composed of 106x106 pixels, we were not able to find this balance while overfitting the dataset.

Qualitative analysis of MuseGAN

For our Jazz outputs, we can see a distinct change from earlier attempts. In “First Attempt”, we can hear a bit more of a regular rhythm and notes played in a more regular sequence. This makes it less distinguishable from other genres. However, as we listen to later outputs (“Second Attempt”) of Jazz we can hear several distinguishing features of Jazz music. For example, there are several instances of ‘crushed’ notes (where two notes next to each other are played together) which is a clear feature of the Jazz genre. With our classical music output, in our first attempts you hear clear features of the genre, like regular rhythms and notes being played in regular time, but the music is simplistic. Our later classical music outputs sound very similar to genuine classical music of the baroque period. There are parallels with many famous works of classical music with dramatic openings of major chords followed by lighter notes. With our rock music, the only distinguishable feature that would align with the genre is the repeated notes and chords being played at a higher tempo. In later iterations, however, we can see a greater number of similarities with the genre. For example, the different modes of playing are done at different times. This is a classical feature of rock music that usually doesn’t have overlapping tunes.

Overall, although the model doesn’t output entirely pleasing tunes, there are several features of the genres that can be distinguished. As hypothesized, classical music came out the best with a melody that most closely resembles the genre. With more computing power we could continue improving the melodies until our music would be aligned with a normal classical/rock/jazz melody.

Objective analysis of MuseGAN

We wanted to take advantage of the fact that the GAN outputs were images. One of the most typical methods to evaluate image similarity is MSE (Mean Squared Error)[8]. Each model produced 10 images. We computed the average MSE each of the outputs had with respect to each genre’s dataset. The idea behind this was to evaluate whether on average outputs that were generated by the Classical generator, for example, were most similar to the Classical music dataset. The results of this analysis can be seen for all three sets of models under Table

[8] MSE as measure of similarity: https://medium.com/@datamonsters/a-quick-overview-of-methods-to-measure-the-similarity-between-images-f907166694ee

2, Table 3, and Table 4 in the Appendix. As can be seen, for all three attempts, all outputs, regardless of the genre they were trained on, were most similar to the classical dataset. While this may not be indicative of failure to reproduce the dataset they originally came from, it does agree with the qualitative analysis, in which classical music was the most recognizable genre.

Comparison of models

We hypothesized that both model models would produce a significant difference between the classical, rock and jazz generated samples, having the rock and jazz samples be the most unrecognizable. Using the LSTM model, the classical music output was the easiest identifiable, the other two outputs would just be categorized as random sounds. This seemed to also be the case with the GAN output. The LTSM & GAN models produced very different outputs. The LTSM focused on slower, more drawn out notes and chords whereas all of the GAN's outputs were much higher tempo. Also, the GAN outputs seemed to more closely follow the musical structure. One feature of these models is that they have no inherent way of understanding musical structure/form. However, the GAN was more able to stick to these rules. For example, playing melodies in a certain key or keeping in regular time. Overall the GAN produced more recognisable melodies that more closely resembled the genre.

6) Compute/Other Resources Used

To acquire the data our group used the Jazz, Classical and Rock datasets found in the Music Genre GTZAN Dataset[9]. A collection of 10 genres, where we used 3: rock, classical and jazz. Each genre has 100 audio files each, all having a length of 30 seconds. Additionally it also contains images, visual representation, for each audio file which were created by converting the audio files to Mel Spectrograms to make this possible.

For MuseGAN, all computations were done on an AWS Sagemaker Notebook of type "ml.g4dn.xlarge", as we wanted to use GPUs for our GANs. Each model would train for upwards of 90 minutes. The total computation cost for MuseGAN was approximately 60 dollars.

7) Conclusions

Training MuseGAN and the LSTM on different genres of music did not result in a significant difference in the music generated. The most recognizable genre from the music generated was classical. The main challenge we faced when adapting these techniques was dealing with computation time and attempting to overfit the data a bit more while obtaining interesting results. Moreover, GANs are particularly challenging to train as there is a delicate balance to be struck between discriminator and generator effectiveness.

While we did obtain some promising results in the music generated, with the exception of classical music our original goal of producing music that resembles the genre the model was trained on was not achieved, so we did succeed in lowering the bar for entry for music creation and allow younger and less experienced creators to enter the field.

Our project also raises interesting ethical concerns. In the acting industry, movie studios are beginning to question if they need to pay actors large sums of money if they can create realistic CGI characters. If our solution were developed to include more genres, one could

[9] Kaggle dataset: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

potentially foresee a scenario where less musicians are paid, and instead people pay to listen to AI-generated music adjusted for their particular taste.

Appendix

Genre	Second Model Loss	Final Model Loss
Classical	4.388	2.923
Jazz	4.748	4.142
Rock	5.698	5.433

Table 1: Categorical Cross-Entropy Loss for the various LSTM models trained by genre

	Classical dataset	Rock dataset	Jazz dataset
Classical output	0.1259	0.1559	0.129
Rock output	0.169	0.193	0.1711
Jazz output	0.1456	0.176	0.149

Table 2: Average MSE comparison of the outputs of the "First Attempt" model set with respect to each dataset

	Classical dataset	Rock dataset	Jazz dataset
Classical output	0.0769	0.136	0.0821
Rock output	0.07341	0.13398	0.07845
Jazz output	0.07942	0.1402	0.0836

Table 3: Average MSE comparison of the outputs of the "Second Attempt" model set with respect to each dataset

	Classical dataset	**Rock dataset**	**Jazz dataset**
Classical output	0.03126	0.1027	0.03708
Rock output	0.06843	0.13239	0.073672
Jazz output	0.0312	0.10278	0.037081

Table 4: Average MSE comparison of the outputs of the "Third Attempt" model set with respect to each dataset

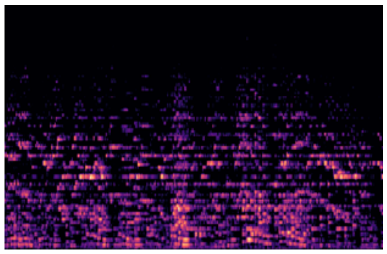


Figure 1: An image representation of a classical melody (given in Kaggle dataset)

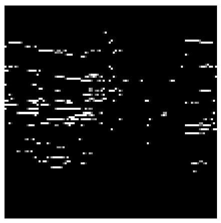


Figure 2: An image representation of a classical melody converted to an image using the midi2image function

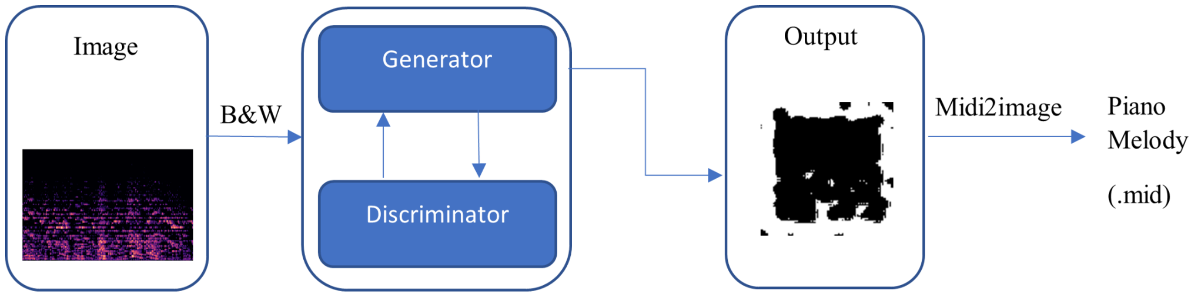


Figure 3: Structure of the first iteration of MuseGAN, where the images from Kaggle were inputted directly

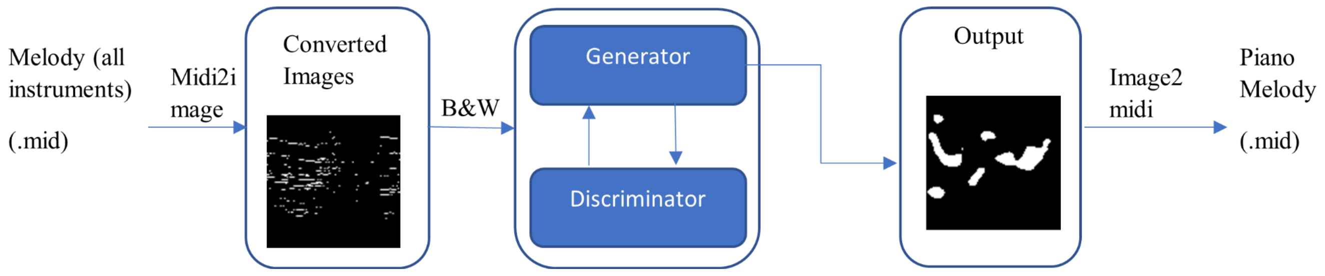


Figure 4: Structure of the second iteration of MuseGAN, where the songs were transformed into images using midi2image before inputting them into the GAN

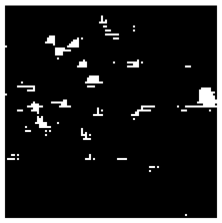


Figure 5: An example of an image successfully created by the generator. In particular, this corresponds to a rock song produced by the second set of models. This gets transformed into a piano-only melody using the function image2midi

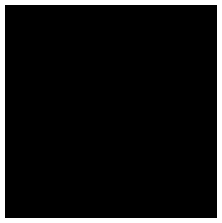


Figure 6: Most attempts to increase overfitting the dataset were unsuccessful, producing images like the one above

Broader Dissemination Information:

Your report title and the list of team members will be published on the class website. Would you also like your pdf report to be published?

NO

If your answer to the above question is yes, are there any other links to github / youtube / blog post / project website that you would like to publish alongside the report? If so, list them here.

(Exempted from page limit) **Work Report**

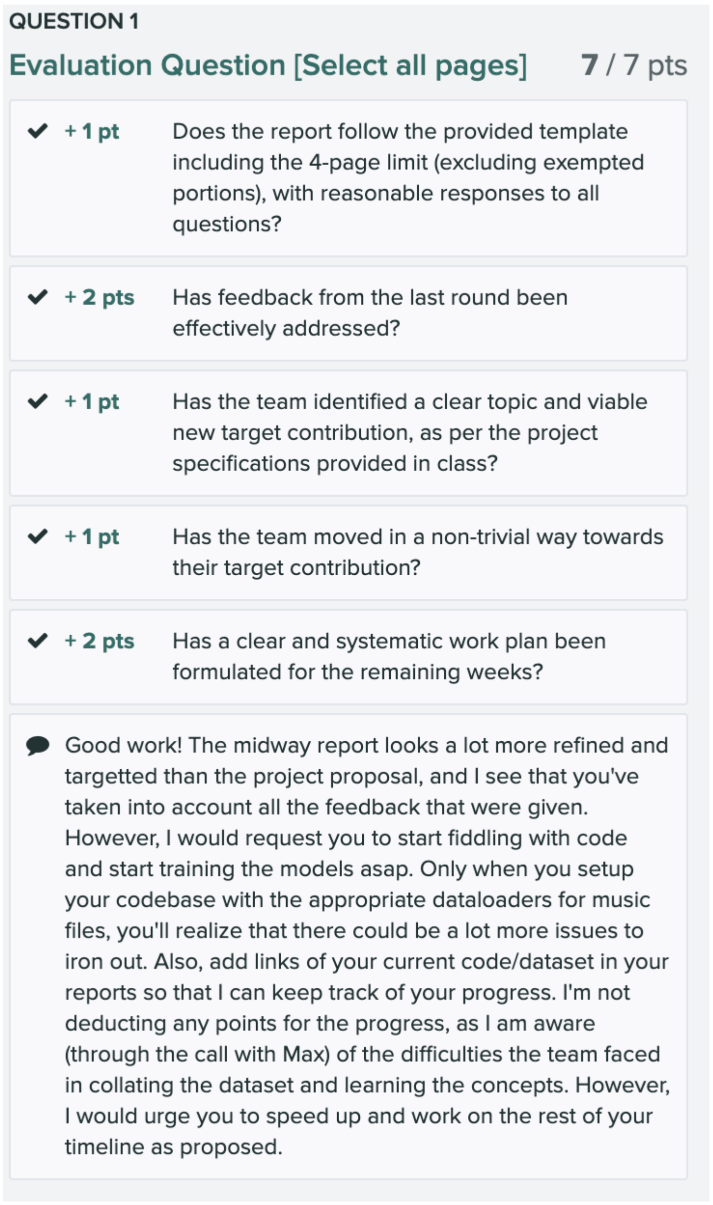
March

PERSON (S)	TASK (S)	Wk5				Wk6				Wk7				Wk8				Wk9			
		March																	April		
		S6	M7	W9	Th10	S12	M14	W16	Th17	S19	M21	W23	Th24	S27	M28	W30	Th31		F1	S2	S3
Max, Mariana, Claes	Identify algorithms/improvement Week 1	🟩																			
Claes	Collect data Week 1	🟩																			
Mariana, Max	Standardize algorithms/ data Week 2	🟩						🟩													
Claes	Identify performance metrics Week 2							🟩													
Claes	Comparison of initial performance Week 2									🟩											
Max, Mariana	Fine Tuning Week 3												🟩								
Max, Mariana	Analysis predictability with different music genres Week 3															🟩					

April

PERSON (S)	TASK (S)	April											
		S10	M11	W13	Th14	S16	M18	W20	Th21	S24	M25	W27	
Max, Mariana	Final Fine tuning				🟩								
Max, Mariana	Finalize sounds								🟩				
Claes	Create soundcloud									🟩			
Claes	Final Analysis									🟩			
All 3	Final Report First Draft										🟩		
All 3	Final Report											🟩	

(Exempted from page limit) Attach your midway report here, as a series of screenshots from Gradescope, starting with a screenshot of your main evaluation tab, and then screenshots of each page, including pdf comments. This is similar to how you were required to attach screenshots of the proposal in your midway report.



QUESTION 1

Evaluation Question [Select all pages] **7** / 7 pts

✓	**+ 1 pt**	Does the report follow the provided template including the 4-page limit (excluding exempted portions), with reasonable responses to all questions?
✓	**+ 2 pts**	Has feedback from the last round been effectively addressed?
✓	**+ 1 pt**	Has the team identified a clear topic and viable new target contribution, as per the project specifications provided in class?
✓	**+ 1 pt**	Has the team moved in a non-trivial way towards their target contribution?
✓	**+ 2 pts**	Has a clear and systematic work plan been formulated for the remaining weeks?

Good work! The midway report looks a lot more refined and targetted than the project proposal, and I see that you've taken into account all the feedback that were given. However, I would request you to start fiddling with code and start training the models asap. Only when you setup your codebase with the appropriate dataloaders for music files, you'll realize that there could be a lot more issues to iron out. Also, add links of your current code/dataset in your reports so that I can keep track of your progress. I'm not deducting any points for the progress, as I am aware (through the call with Max) of the difficulties the team faced in collating the dataset and learning the concepts. However, I would urge you to speed up and work on the rest of your timeline as proposed.

Music by Genre Synthesizer Comparison and Improvement

Team: Mariana Luna (519, Aadith's cohort), Claes Sjoborg (519, Mahesh's Cohort), Max Speno (419, Benedict's cohort) **Project Mentor TA:** Benedict

1) Introduction

We will be synthesizing the next portion of a song for three distinct genres by comparing the performance of two different Generative Adversarial Network (GAN) models. We have identified two Github repositories outlined below that we will adapt and improve to achieve our specific goal.

A GAN is composed of a discriminator, whose function will be to determine whether a specific song belongs to a specific genre. The inputs to the discriminator will be midi (.mid) files of training songs of three distinct genres (jazz, classical and rock). This GAN will be built and tuned three times, one for each genre, so that we will be able to compare the performance based on genre.

The second model of the GAN is the actual generator. The generator is in charge of "fooling" the discriminator. If the generator fails in tricking the discriminator, the generator learns from this and adapts its parameters to produce a new output, which will again be inputted into the discriminator.

To evaluate the quality of our generators we will have to evaluate how well they perform on the discriminators we have designed. We would also like to use widely accepted GAN evaluation metrics such as the Inception Score (IS) and the Fréchet Inception Distance (FID). Additionally, we will evaluate these models in an objective and a subjective manner. Objectively, we would like our synthesis to resemble the genre they belong to, and compare how capable each GAN is of producing music belonging to a specific genre. Subjectively, we would also like to observe whether these models can add value to the melodies they originated from.

If we make progress towards solving this problem, we believe that this could have a significant impact on the music industry as a whole. If we are able to create reasonable and pleasing melodies, one can begin to question if we need the human expertise to create the music. We believe that we are lowering the bar for entry for music creation and allow younger/less experienced people to be able to create their own music for specific genres very easily.

Our project also raises interesting ethical concerns. In the acting industry, movie studios are beginning to question if they need to pay actors large sums of money if they can create realistic CGI characters. If our solution was developed to include more genres, we could begin to question whether musicians are needed at all. Instead of having to pay for a Spotify subscription, you could create your own music in the genre you like.

Additionally, an interesting part of this project will be in the comparison between the generative abilities of different genres. We could hypothesize that generating cohesive melodies of classical music would be easier because of the repetitive and structured nature of the form. Conversely, we could also say that because of Jazz's more free flowing and improvised nature, it may be more difficult to generate new melodies. We want to see if these hypotheses hold true and if a certain genre of music is more 'generative' than others.

2) How We Have Addressed Feedback From the Proposal Evaluations

The main feedback we received from our proposal was to constrain our problem. Initially we wanted to reconstruct songs, create songs, analyze models, and find the correlation between predictive ability and popularity, which is too broad. We will now focus on synthesizing songs in three distinct genres (classical, jazz, classical rock) and compare the outcome of the models.

Our proposal had identified several models that in fact do different things. We had stated that we intended to use Markov Models, LSTM (Long Short Term Memory Neural Networks - a kind of Recurrent Neural Network), as well as GANs. The problem with this is that LSTMs are built to predict the next part in a time series, while a GAN is intended to create something from the inputs that can trick the discriminator.

As such, our models will be based on two different GANs that we have found online. We will expand on this in the next section.

3) Prior Work We are Closely Building From

- The first model we will be using to accomplish our task will be a tensorflow model called "GANSynth". As previously specified, the input songs will be in MIDI files. The authors offer a description on how to retrain the model with new data. Unlike our specified goal, this GAN was not trained to produce music of a specified genre.
    - https://github.com/magenta/magenta/tree/main/magenta/models/gansynth
- The second model we will also test on different genres will be MuseGAN. This model has specifically been trained to generate pop music. We will need to adapt this model to be better suited to rock, classical, and jazz.
    - https://github.com/salu133445/musegan

Model references

1. Engel, Jesse & Agrawal, Kumar Krishn & Chen, Shuo & Gulrajani, Ishaan & Donahue, Chris & Roberts, Adam. 'GANSynth: Adversarial Neural Audio Synthesis'. 2018. Google AI.
2. Dong, H & Hsiao, W & Yang L & Yang Y. 2018. 'MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment'.

4) What We are Contributing

1. **Contribution(s) in Application:** While GANs have previously been used to generate music, our specific goal of generating genre-specific music for jazz, classical and rock has not. We will need to readapt the models we are basing our work on to achieve our specific goal.
2. **Contribution(s) in Analysis:** We will have two models trained three times for each genre (jazz, classical and rock). We will have to compare the performance of each model, as well as the performance with respect to each genre. We will evaluate the resemblance of the music produced with respect to the genre they belong to. This will help us identify whether the complexity of the input music has a relationship to the performance of each model.

5) Detailed Description of Each Proposed Contribution, Progress Towards It, and Any Difficulties Encountered So Far

We had some difficulties acquiring a significant dataset. Much of the software we encountered to download the data had issues with viruses. We have now downloaded several YouTube playlists for each genre and split each song into multiple fragments as the inputs to the models we are using are much shorter in length. MuseGAN approaches these tracks by summing their piano-rolls in order not to have empty bars; we have avoided these tracks. We have also converted the files to MIDI format. Additionally, we are currently implementing the MuseGAN set up for acquiring musically meaningful phrases to train and feed the model in order to obtain good segments in the end. This entails using structural features [Serrá et al., 2012] and obtain phrases accordingly. This piece has been a challenge, as the tracks are in distinct times considering the metric of four bars, we have had to structure the tracks and prune the longer segments. We have also set up the Amazon SageMaker notebooks with our chosen models.

As was stated above, our project consists of an application as well as an analysis.

Application

1. GanSynth

GanSynth was developed by Tensorflow Magenta which was originally created by Google Brain employees. The paper describing its functionality can be found in the references.

Our application contribution is focused more on the training of the model. GANSynth was trained on the NSynth dataset, which contains 300,000 musical notes from 1,000 different instruments. By contrast, we would like to train the model on specific genre songs. While the input to the model will still be audio, there is a significant difference between our inputs and theirs. Adapting the GAN for our specific goal is a challenge. This will include modifying the hyperparameters of the model to be suited to our audio inputs (audio length, sample rate, …), and determine the loss function best suited to our limited dataset. Once the GAN has been trained, we will need to change the way it generates outputs. The current GANSynth generates notes, and if a note fails to be generated, it does so randomly. Since our output will be more complex, we will need to change how this random operation is performed.

2. MuseGAN

MuseGAN is an integration and extension of two distinct GANs, the multi-track model and the temporal model. The input to MuseGAN is composed of an inter-track time-independent random vectors $\mathbf{z}$, an intra-track time-independent random vectors $\mathbf{z}_i$, an inter-track time-dependent random vectors $\mathbf{z}_t$, and an intra-track time-dependent random vectors $\mathbf{z}_{i,t}$. We will be using their track-conditional approach where the bar sequence $\mathbf{y}$ is given to the GAN, and it tries to learn the temporal structure underlying that track and to generate the remaining tracks. The main difference with simply creating novel music compared to using a previous bar sequence $\mathbf{y}$, is that they add an additional encoder which is responsible for extracting useful inter-track features from the user-provided track.

The MuseGAN model wants to understand how pop music is composed, which is a variation and combination of different genres and instruments. It is trained on data with distinct instruments, specifically they pick only the piano-rolls that are in 4/4 time. The application contribution will be adapting the MuseGAN model to understand the different genres of music that we have suggested. This includes modifying hyperparameters like audio length, pitch, batch normalization, and changing the data preprocessing.

Analysis

As is outlined in the GANSynth paper [Eng et al.], the output produced by GANs is difficult to evaluate. We have outlined several possible metrics that we will use to determine the quality of the result of our applied models. We plan on evaluating the output using a qualitative analysis (human evaluation), as well as more quantitative metrics such as NDB (Number of Statistically Different Bins, measures whether the output is significantly different from the training set), IS (Inception Score, penalizes models whose output cannot be well classified into a single class), Pitch Accuracy and Entropy as well as Frechet Inception Distance. The MuseGAN paper proposes a metric QN which is the ratio of "qualified" notes (in %). They consider a note no shorter than three time steps, a 32th note, as a qualified note. This shows if the music is overly fragmented. Adding this metric to the GANSynth can help us quantify what we are hearing with regards to its specific genre.

6) Risk Mitigation Plan

We will start with two previously built models, GanSynth and MuseGAN. We understand that GANs are hard to train because they require large amounts of data and a lot of computation time. The paper describing GANSynth's architecture clarifies that locally coherent audio waveforms are more difficult to achieve with GANs. There is a strong likelihood that our models will underperform models like the 'magenta' model which trains on a very large dataset with an incredibly expensive memory of v100 GPU which would take 3-4 days to train accurately. Therefore, we will attempt to train the GanSynth and MuseGAN models on a bigger dataset but if this fails, we will resort to a simplified dataset, where each of our three chosen genres contain around 100 songs.

Our second goal is to compare the outputs of the two GANs on three different genres. Our minimum viable project would be to compare just two genres on two different GANs. Failure, in terms of our project, would be to synthesize songs that then digress from their original genre. However, we believe that we will still be able to create some novel music. In the event that both of our GANs are not outputting the 'expected' melodies, we can find specific examples in which they output reasonable melodies and compare them to ones that do not work as well. This can be done inside of each genre and finding similarities between failures throughout different genres. This can help us understand the structure of GANs and how they respond to different inputs.

(Exempted from page limit) Other Prior Work / References (apart from Sec 3) that are cited in the text:

3. Serra, J.; Mller, M.; Grosche, P.; and Arcos, J. L. 2012. ` Unsupervised detection of music boundaries by time series structure features. In AAAI.

(Exempted from page limit) Supplementary Materials if any (but not guaranteed to be considered during evaluation):