MuML: Musical Meta-Learning

Omer Gul
Computer Science Department
Stanford University
Stanford, CA
momergul@stanford.edu

Collin Schlager
Computer Science Department
Stanford University
Stanford, CA
schlager@stanford.edu

Graham Todd
Symsys Deparment
Stanford University
Stanford, CA
gdrtodd@stanford.edu

Extended Abstract

In the following work, we investigate the performance of meta-learning approaches on predicting sequences of music. Our work continues prior research in both artificial music generation and meta-learning and is motivated by a relative dearth of studies combining the two fields. We first investigated the ability of meta-learning approaches (specifically, MAML trained on batches of tokens that were produced from a custom encoding of MIDI files) to adapt to differing musical genres. We hypothesized that meta-learning would allow for artificial music generation models to rapidly acclimate to novel genres of music, which would in turn improve its ability to predict and generate other examples of music in that genre. To test this hypothesis, we used a subset of the Lakh MIDI dataset, splitting roughly 3800 songs into 13 distinct genres, with a varying number of songs in each. We compared our meta-learning approach against a simple baseline that more closely mirrors the "pretrain and fine-tune" approach common in the literature by training the model on batches of tokens without any distinction made by genre. We performed our experiments using both simple two-layer LSTMs and more powerful, attention-based Transformer models. Quantitatively, we found that the baseline and meta-learning models performed almost equivalently (as measured by negative log-likelihood on samples from the test set), falsifying our initial hypothesis. We also performed a qualitative analysis by implementing many automatic style metrics present in the literature and comparing the distribution over these metrics between the baseline and meta-learning models. We found that both the baseline and meta-learning models produced outputs which, broadly, matched the style of the target genre. Most notably, however, we observed that both the baseline and meta-learning models had higher performance *without* adaptation. We suspected that this may have been due to the noisy task definition: two songs within a single genre might differ substantially in musical content, which would make adaptation a lossy signal.

In order to test whether a finer task definition would aid our meta-learner's performance, we re-conducted our experiments on a distinct dataset: the Maestro collection of classical piano performances. Instead of defining tasks by genre, we instead assigned each individual song to its own task, with the intuition that individual songs are likely to exhibit much greater intra-task consistency than entire genres. Quantitatively, we found that the meta-learning model performed marginally better than the baseline on the new dataset, though this may have been an effect of noise. Qualitatively, however, we observed that the meta-learner was able to produce samples that aligned much more closely with the underlying distribution over musical features, when compared to the baseline. Surprisingly, adaptation remained ineffective in improving the performance of either model with regard to negative log-likelihood. While more work would be necessary to determine exactly why, we theorize that the negative log-likelihood may simply be an ineffective measure of a model's ability to produce music of a particular style, a fact which is supported by the overall low range of negative log-likelihoods observed across all models and experimental setups. We conclude that meta-learning is a promising approach for domain-specific music generation, but that extra care must be taken to select tasks with high internal consistency. We also encourage the use of qualitative metrics that attempt to capture the underlying features of music, rather than just the model's predictive capacity.

1 Introduction

Generative modeling methods have experienced immense improvements in recent years for both image [16] and text modalities [4][2], with techniques capturing the underlying distributions and structures with ever increasing accuracy. The field of music generation has been one beneficiary of the success of language modeling as the research community has produced a wide array of methods focused not only on developing better representations [3][20] and generative models [6][15][6] of music in general, but also on problems specific to music and musical performance [14][17].

This progress is highly encouraging, not only from a theoretical machine learning standpoint but also from a practical perspective, as improved music modeling techniques may unlock new avenues for human-AI collaboration. One application, which is the focus of our project, is for a model to generate pleasing, style-specific music samples. These computer-generated snippets could then aid and/or inspire a human composer in the composition of songs belonging to a particular style.

Although existing musical models are capable of producing high-quality samples [15], they are trained primarily as unconditional generative models and are thus incapable of adapting to a collection of samples of interest. The models that do allow for controllable music generation in different styles either rely on predefined style tokens and are thus limited in what they can adapt to [18] or can condition only on a single arbitrary input [3]. A further problem is that each of these methods assume the existence of a large amount of data for training, which renders their application to corpora containing novel styles or songs difficult.

We believe that meta-learning [8] can provide one solution to this problem. Specifically, in our project, we frame each collection of samples that a user provides as a task and require the model to correctly generate another piece belonging to the samples' style. These samples may be snippets from a genre of music or from a lengthy classical music composition. We hope that within this reframing, the meta-learner can discover the commonalities between the inputs in an unsupervised manner and thus generate samples conditioned on an arbitrary style specified by an arbitrary number of examples. We compare the performance of meta-learning and more standard pre-training approaches for sequential music prediction across multiple distinct neural architectures and datasets, present results for both quantitative and qualitative metrics, and provide a robust discussion and error analysis.

2 Related Work

Music Generation: Although work on artificial music generation with neural networks predates the Deep Learning era [7], the increase in computational resources has led to a flurry of new work. Methods such as the Music Transformer [15], MusicVAE [20], MuseGAN [6] and MuseNet [18] treat music as a sequence of tokens and strive towards learning generative models that can adequately model the long-term structure and polyphonic nature of music. In addition to these models that treat music as a general language modeling task, there has also been a variety of works focused on specific musical problems, such as iterative composition with counterpoint [14], drumming [9], and expressive dynamics in musical performance [17].

Related to our project, there have been two strains of methods for controlled music generation. MuseNet [18] is an example of the first strain where the model takes as input a composer or style token in addition to the music encoding. Although the model can adeptly generate samples from within the prescribed styles, its reliance on predefined tokens prevents it from generalizing to novel styles on demand. On the other hand, the second strain of methods rely on song embeddings for style transfer. For example, Choi *et al.* [3] use Music Transformers to learn global representations of performance style, which a model can then condition on to produce similar music, while Dinculescu *et al.* [5] use MusicVAE representations to train personalized generative models for music. Models from the second strain, once trained, can allow for fast adaptation to user input; however, successful training requires large amounts of data and computational power. Our goal is to investigate whether meta-learning can allow for similar few-shot transfer with limited resources.

Meta-Learning with Sequential Data: Meta-learning methods, specifically MAML [8], have been used in a variety of settings involving sequential data generation, such as low-resource machine translation [11] and code switched speech recognition [22]. Due to MAML's success within these settings, we choose to explore MAML as the meta-learning method for our application.

3 Datasets and Encoding

3.1 The Lakh MIDI Dataset

For our first experiments we used a subset of the Cleansed Lakh MIDI dataset [19], which is itself a subset of the Million Song Dataset (MSD) [1]. In specific, the Cleansed Lakh MIDI datset is a collection of 21,425 multitrack pianorolls that have been filtered from the larger Lakh dataset to only include songs with a time signature of 4/4. From this subset, we include only those songs for which we were able to obtain genre information, as defined by their corresponding appearance in the MSD. We were able to acquire this genre information for 3827 songs, comprising the following genres: Pop Rock, Folk, Country, Electronic, Blues, Latin, Reggae, RnB, Rap, International, Vocal, New Age, and Jazz. Pop Rock was the largest genre, with 2423 songs, while Blues and Vocal were tied for the smallest with 32 each. We split the songs at the genre level into training, validation, and test splits (see Table 1).

Table 1: Lakh Midi Dataset with Genre Information

Train		**Validation**		**Test**	
Genre	Song Count	Genre	Song Count	Genre	Song Count
Vocal	32	RnB	214	Country	271
Folk	63	Blues	32	Reggae	34
Pop Rock	2423	Latin	133	Jazz	101
International	58	-	-	-	-
Electronic	366	-	-	-	-
New Age	36	-	-	-	-

3.2 The Maestro Dataset

For our second set of experiments, we used the Maestro dataset [12] curated by Google Magenta. The Maestro dataset consists of 1282 MIDI files of classical piano performances, primarily representing compositions from the 17th to 20th centuries. In contrast to the Lakh dataset, the pieces in the Maestro dataset do not differ in genre, but rather differ in style between distinct compositions and even distinct performances of the same composition. We preserve the training splits in the Maestro dataset while producing the task splits for our experiments.

3.3 The Pitch-Duration-Advance Encoding

We use a novel encoding for MIDI files first designed for our music generation project for last year's offering of CS236. This encoding represents each note in a MIDI file with three tokens: a *pitch* token, a *duration* token, and an *advance* token, each ranging in value from 0 to 128. The *pitch* token simply indicates the note's frequency, with higher values indicating a higher frequency. We reserve a pitch value of 0 to indicate a rest, where no notes are being played. The *duration* token indicates how long the note should be held for, in number of 16th notes (triplets and other odd-metered notes are rounded to the nearest 16th note). Finally, *advance* token indicates the number of 16th notes before the next note should be played. We differentiate between *duration* and *advance* to allow for polyphony: by setting *advance* to 0, the next encoded note will be played at the same time as the previous one, forming a chord. Finally, we also provide each model a *positional* encoding for each token, which simply indicates whether the token represents a *pitch*, *duration*, or *advance*. Figure 1 provides an illustration of this encoding structure for a few example notes.

4 Models

We use two distinct neural architectures for our experiments: a simple **LSTM** [13] and the attention-based **Transformer** [21] model. For the LSTM, we use a two-layer, single-directional implementation. At each step, the LSTM is given a concatenation of the embedding for the current token and the current position. In this sense, the *token embeddings* are shared for each of the three kinds of tokens, while the *positional embedding* is used to distinguish between them. As is standard for recurrent

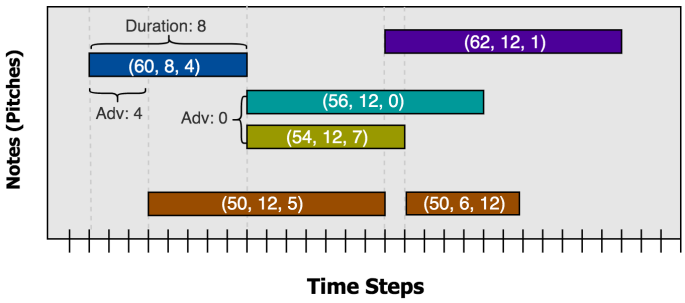


Figure 1: The three-part encoding of a midi soundtrack. Each note is represented by a 3-tuple of (`pitch`, `duration`, `advance`). Notice that two notes are played simultaneously when the `advance` token is 0, which enables the encoding of chords and other polyphonic structures.

models, the LSTM produces a distribution over the next possible token value (note that, since the order of the tokens is *always* pitch-duration-advance, we do not ask the model to predict the type of the following token, only its value), and the error signal is the Cross-Entropy loss between this predicted distribution and the actual token value.

The Transformer proceeds in much the same way, except that an attention mask is used in place of sequential token production. We use the same encoding structure and error signal as with the LSTM, and similarly proceed in a uni-directional manner. We have elected not to use the Music Transformer [15] because the input lengths that we are concerned with are short enough that we expect the benefits associated with the more complex Music Transformer to be minimal.

5 Lakh Dataset Experimental Setup

5.1 Task Definition

As mentioned above, we have split the Lakh dataset into distinct genres. We thus define a *task* in this context as a collection of distinct musical snippets from a particular genre, split into a *support* and *query* set. Specifically, for each training step we first sample a genre from the training split of genres. Then we sample $K_{train} + K_{test}$ distinct songs from that genre. For each sampled song, we take a random window of tokens representing 120 musical notes. The first K_{train} windows are concatenated into the support set, while the remaining form the query set. As is typical for MAML, the model is trained on the support set during the inner loop and evaluated by its performance on the query set. Concretely, for the case of a single inner-update and focusing on a given task i, the objective is to minimize the loss over the meta-parameters θ by first updating the parameters using gradient descent on the support set, $\mathcal{D}_i^{support}$, and using these learned parameters to evaluate performance on the query set, $\mathcal{D}_i^{query}$. This objective can be concisely written as follows:

$$\min_{\theta} \sum_{\mathcal{T}_i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{support}), \mathcal{D}_i^{query})$$

The hypothesis implicit in this task definition is that adapting to snippets of music from a particular genre will improve the model's ability to predict novel sequences of music from the same genre.

5.2 Baseline

This hypothesis is primarily compared against a simple, genre-agnostic baseline. The baseline model, which uses the exact same architecture as the meta-learners, is simply trained on batches of tokens from all songs in the training split of the dataset, regardless of which genre they belong to. This approach is more akin to existing research in music prediction and generation. The baseline models are evaluated in the same manner as models trained using MAML. Given a task, the models are first finetuned on the support set and evaluated on the query set afterwards. In order to provide an additional test to our hypothesis, we also provide a zero-shot adaptation test to assess how important the adaptation step is for both the MAML and baseline models.

5.3 Details

In the following experiments, the models were trained for 15000 iterations. Both the LSTM and Transformer used an embedding dimension and hidden dimension of 128. Each model was trained on windows of 120 tokens corresponding to 40 distinct notes. The models were trained using the `Adam` optimizer, with an initial learning rate of 0.003. We note that, due to the need for sparse second-order gradient operations, both MAML models used simple SGD as their inner optimizer, but still used `Adam` as the outer optimizer. Facebook Research's `Higher` package was used to implement the higher-order gradients over the outer loop. [10].

6 Lakh Dataset Results

6.1 Quantitative Results

The quantitative results to the experiments described in the section prior can be seen on Table 2 below. The column labeled "Standard" denotes the models' negative log likelihood performance when evaluated in the standard fashion following finetuning on a given task's support set, while "Zero-Shot" denotes the models' zero-shot transfer performance.

Table 2: Negative Log Likelihood Performance on the Lakh Midi Dataset

Model Name	Standard	Zero-Shot
MAML (LSTM)	1.71 ± 0.098	1.66 ± 0.095
MAML (Transformer)	1.63 ± 0.098	1.57 ± 0.096
Baseline (LSTM)	$\mathbf{1.61 \pm 0.099}$	1.57 ± 0.10
Baseline (Transformer)	1.63 ± 0.096	$\mathbf{1.56 \pm 0.098}$

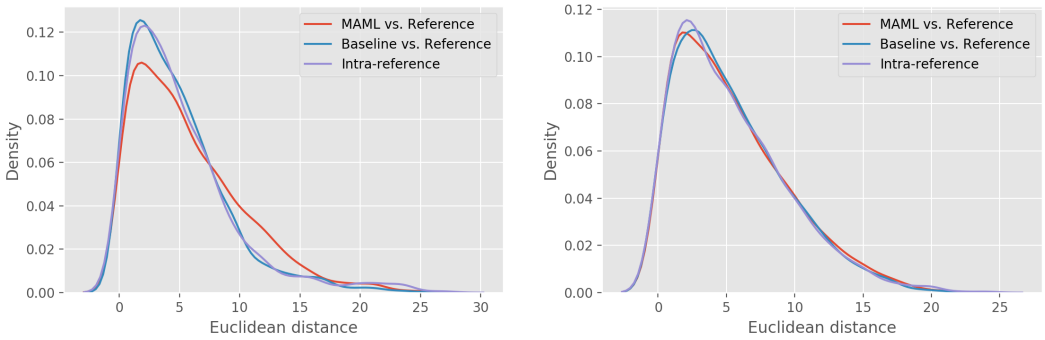
We find that our hypothesis has been falsified with both of our controls. Not only do the baselines perform identically to the MAML models, *all* of our approaches suffer in performance when given the chance to adapt to the support set.

6.2 Qualitative Results

Although the negative log likelihood results strongly indicate that the different methods perform identically, this is only one way to assess the success of our models. In addition to testing the trained models' capability to adequately generate a snippet from the query set, we can also evaluate how close *stylistically* the samples that the trained models generate are to the reference samples defining a task. Performing these tests will give us direct insight into whether the models are adapting to stylistic cues, and, importantly, whether there are enough stylistic cues for the models to work with.

Evaluating the stylistic similarity of different samples is an open problem in the literature [3]. Current approaches make use of several heuristics with inspiration from music theory. Here, we utilize the metrics Yang *et al.* [23] propose. These metrics include aggregate characterizations such as total pitch (TP), which represents the total number of distinct pitches in a piece or snippet, pitch count (PC), which represents the number of distinct pitches out of a maximum of 12 in a piece or snippet independent of octave, and pitch shift (PS), which represents the average size of the interval between two consecutive pitches in semitones. In addition to these aggregate metrics, they also propose more informative characterizations such as histograms and transition matrices for pitch classes (PCH and PCTM) and note lengths (NLH and NLTM).

We make use of these metrics in the following manner. For each task sampled during evaluation, we make use of Yang *et al*'s `mgeval` package to compute the style metrics for the reference samples and for the samples that the MAML and baseline models generate. Afterwards, we extract information regarding stylistic similarity by computing the Euclidian distances between metrics. If a metric is a scalar, then we compute the squared difference. If the metric is represented by a vector or a matrix, as is the case for the histogram and the transition matrix respectively, then we compute the relevant L2 norm. We compute Euclidian distances for three different pairings. We first compute the Euclidian distances between samples within the reference set. We collect these distances to produce a null "intra-reference" distribution that characterizes the ground-truth style of a particular genre. Afterwards, we



(a) Jazz - Average Pitch Shift (PS)

(b) Jazz - Total Used Pitch (TP)

Figure 2: Representative style metrics for the Jazz genre in the Lakh Midi Dataset. Plots display kernel density estimates for the Euclidean distances of the style metric within a single genre and between all genres. Distinct tasks, as measured by these style metrics, would display a distinct distribution from the inter-genre, null distribution.

Table 3: Style Metric Comparison of MAML vs Baseline for Test-Set Genres (Overlapping Area)

Genre	Model	PR	PS	TP	PCH	PCTM	NLH	NLTM	Avg
Country	Baseline	0.934	0.929	0.939	0.882	0.861	0.963	0.957	**0.924**
	MAML	0.832	0.913	0.943	0.931	0.794	0.956	0.951	0.903
Jazz	Baseline	0.926	0.922	0.936	0.933	0.837	0.954	0.962	**0.925**
	MAML	0.923	0.863	0.937	0.901	0.729	0.871	0.937	0.880
Reggae	Baseline	0.867	0.858	0.790	0.904	0.861	0.908	0.885	**0.868**
	MAML	0.709	0.819	0.722	0.911	0.858	0.903	0.922	0.835

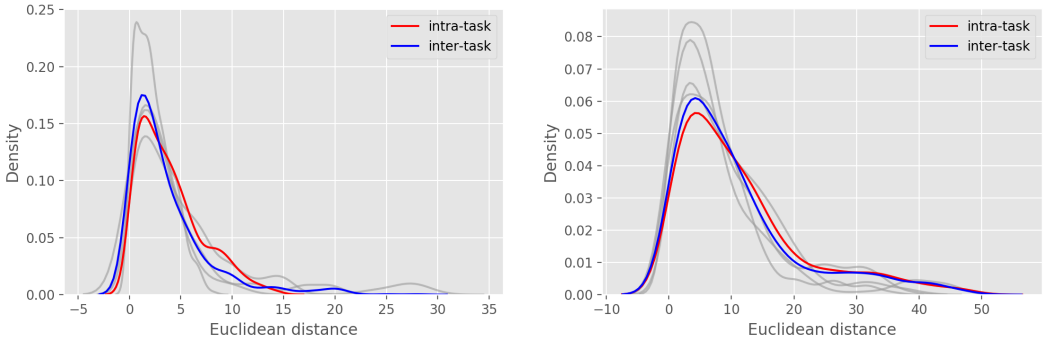
compute the Euclidian distances between the samples generated by the baseline and MAML models and the reference examples to obtain our second and third distributions respectively. If a model's distance distribution with the reference set is similar to the null "intra-reference" distribution, then the samples it generates are within the style of the reference. For a particular task, we compute the similarity of two distributions using their overlapping area. This metric is motivated by the literature: Choi *et al* and Yang *et al* note that the overlapping area metric was found to be less sensitive to song-specific features and more general to the underlying style as compared with other divergences such as the Kullback-Leibler (KL) divergence and the symmetrerized KL [3]. The overlapping area is then averaged across tasks to give the score for each metric.

Figure 2 provides a specific example of the style distance distributions for a particular genre task: Jazz. Here, average pitch shift (PS) and total used pitch (TP) are shown as representative metrics. Similarity to the intra-reference distribution suggests stylistic similarity to reference style. The average overlapping area for each metric can be seen in Table 3. While it seems that the baseline performs slightly better than the meta-learner on these metrics (as indicated by marginally greater average overlapping area), the differences are in general quite minor. We find that the produced samples from both models have similar quality.

6.3 Measuring Stylistic Differences Across Genres

The success of the baseline models, relative to the meta-learners, led us to hypothesize that our genre-based task definition was unsatisfactory. In particular, this split appeared to lack stylistic coherence *within* a task and did not exhibit enough stylistic differences *across* tasks. These properties would enable a baseline model to make use of general musical competencies to generate quality samples–the value of genre-specific training degrades when genres themselves are stylistically ambiguous. To test this hypothesis, we performed another experiment assessing differences between genres using the same metrics.

Specifically, using code provided by Yang *et al.* in their `mgeval` package, we compute representative style metric distributions for each genre and compute a reference null distribution for songs across the different genre sets. The representative distributions are computed by sampling $N = 32$ songs from a single genre, computing the stylistic metrics for each, and then computing the pair-wise Euclidean distance between the stylistic metrics in an exhaustive, leave-one-out style fashion. The distributions over these intra-genre Euclidean distances describe the intra-genre consistency for a given style. These distributions are compared to an *inter*-genre null distribution, which is computed in a similar way agnostic to genre distinction. We observed that the distributions *within genres* are quite similar to the distributions *between genres*, which indicates that the genres identified by the Lakh dataset do not carve out unique areas of style. Figure 3 depicts the average *intra-* genre style distribution plotted against the null distribution for all genres. The marked similarity between the style distributions within a given task and between all tasks supports our hypothesis that songs across an individual genre (as denoted by the Million Song's Dataset) do not consistently display a unique style signature (as measured by our stylistic metrics).



(a) Average Pitch Shift (PS)

(b) Total Used Pitch (TP)

Figure 3: Representative style metrics comparing the style consistency for the genre tasks in the Lakh Midi Dataset. The red trace describes the average *intra*-genre style distribution, while the blue trace describes the *inter*-genre style distribution. Grey traces are the intra-genre style distributions for individual genres. Note that the music style distribution across genres is markedly similar to the style distribution within genres. This suggests there are similar amounts of style variation within a genre as compared to across all genres.

6.4 Discussion

When we consider the prior sections together, a clear picture emerges. Using meta-learning to adapt to example snippets sampled from songs belonging to a particular genre does not lead to improved performance in generating music from that genre. Our intuition and stylistic analysis suggest that this is because the genres themselves–and therefore the tasks that we define–do not contain appropriate structure for the meta-learner to constructively adapt to. What we instead find is that songs within genres are diverse and differences between genres are not sharp enough. As such, a baseline model with access to samples from all genres during training can achieve a better understanding of musical structure. As a result, it can produce samples that adhere to the reference styles in a manner equal, if not marginally better than, the meta-learning model. We note that the negative log likelihood scores of the MAML models are likely similar because they have also captured this general structural information during training.

This realization further reveals an obstacle to the goals we outlined in the introduction. Although meta-learning has the flexibility to adapt to varied inputs, if the meta-training tasks do not possess an identifiable structure, then the model optimizes a vague objective. Model updates over a vague objective could hurt performance, which may explain the decreased performance of our models when compared to their "Zero-Shot" equivalents. In order to determine whether meta-learning is truly an appropriate approach for customized music generation, we are motivated to explore a different task definition with a finer structure.

7 Maestro Dataset Experimental Setup

7.1 Task Definition

The particular dataset that we have chosen to explore in light of our findings with the Lakh Midi dataset is the Maestro dataset. For the Maestro dataset, we investigate a different task definition. In particular, we assign each *song* in the Maestro dataset to its own task, splitting tasks using the original training / validation / testing splits provided by Google Magenta. The intuition behind this task definition is that, while the broad genre of "classical music" likely contains a multitude of different styles, each song is much more likely to be internally consistent with respect to a single musical style. Thus, we wanted to investigate the extent to which meta-learning could be used to adapt to the style of a specific performance of an individual piece, with the hypothesis that doing so would improve a meta-learning model's ability to predict other sequences from the same piece.

More specifically, for each training step, we select a single song from the training split. We then sample $K_{train} + K_{test}$ non-overlapping windows from that song, and split them into a *support* and *query* set, just as with the Lakh dataset. From there, training proceeds in an identical fashion as with the Lakh dataset.

7.2 Baseline

The baseline for the Maestro dataset closely mirrors that for the Lakh dataset. The baseline model once again uses the same neural architecture as the meta-learners and is trained on batches of tokens from the training split, with no special distinction made between different songs. Due to the success and greater speed of Transformer models, we restrict our attention to experiments using Transformer architectures.

7.3 Details

As before, models were trained for 15000 iterations with an embedding and hidden dimension of 128. A window size of 120 tokens was used, and the models were optimized using `Adam` with an initial learning rate of 0.001. In the case of MAML, simple SGD is used as the inner optimizer with a learning rate of 0.003.

8 Maestro Dataset Results

8.1 Quantitative Results

The quantitative results to the experiments described in the section prior are provided in Table 4 below. As with the Lakh dataset, we present results from two settings: the "Standard" setting using MAML updates or fine-tuning given the context of a support set, and the "Zero-Shot" setting which measures the model's pure transfer performance to the new song. Note that evaluation is performed over the test-split of songs and is measured in terms of negative log-likelihood.

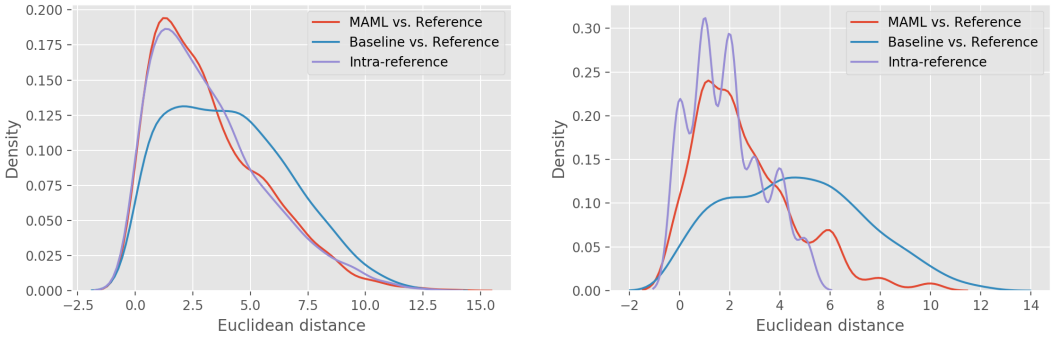
Table 4: Negative Log Likelihood Performance on the Maestro Dataset

Model Name	Standard	Zero-Shot
MAML (Transformer)	$\mathbf{2.00 \pm 0.045}$	$\mathbf{1.95 \pm 0.044}$
Baseline (Transformer)	2.040 ± 0.046	1.99 ± 0.045

In contrast to the results of the Lakh datset, we find here that the meta-learning approach *does* outperform the baseline, though only barely. Given the standard deviations on the reported Negative Log Likelihoods, it is most appropriate to say that the models perform comparably. We find once again that performance degrades slightly for both models when they are asked to adapt to the support set.

8.2 Qualitative Results

We present the same set of qualitative metrics for the Maestro dataset below. Note that comparisons between *genres* have now been replaced with comparisons between *songs*, as that is our task definition for the Maestro dataset. Mirroring the results above, Figure 4 shows the distribution of values for two style metrics, comparing between models. Table 5 broadens these results to all of the metrics presented in Yang *et al.*



(a) Average Pitch Shift (PS)

(b) Total Used Pitch (PT)

Figure 4: Representative style metrics for a test-set task: *Sonata in A major K. 208* by Domenico Scarlatti. Plots display kernel density estimates for the Euclidean distances of the style metric within a single genre and between all genres. Distinct tasks, as measured by these style metrics, would display a distinct distribution from the inter-genre, null distribution.

Table 5: Style Metric Comparison of MAML vs Baseline for Test-Set Genres

Model	NLTM	PCH	TP	PS	PR	PCTM	NLH	Avg
MAML	0.890	0.876	0.860	0.866	0.844	0.875	0.890	0.872
baseline	0.804	0.753	0.80	0.829	0.697	0.860	0.789	0.790

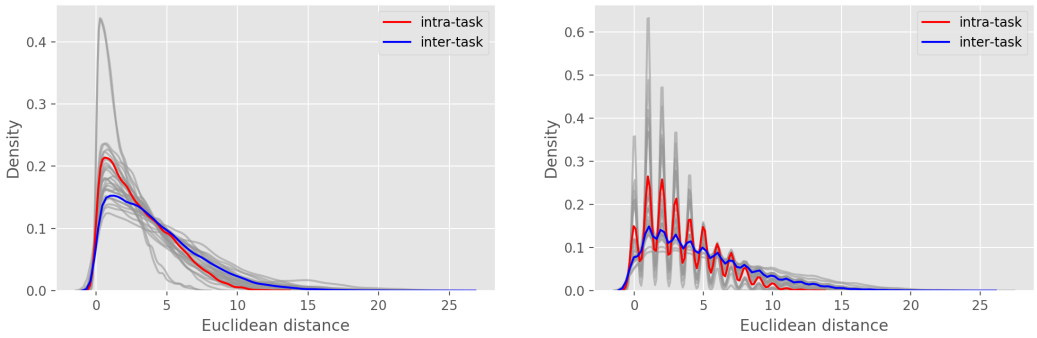
In contrast to the quantitative results, here, we see a more significant difference between the MAML and baseline approaches. In particular, we find that outputs from the meta-learning model much more closely match the underlying target style, as measured by the overlapping area with the intra-reference style distribution. Notably, the MAML model outperforms the baseline model for all style metrics. This indicates that the meta-learning model has, in some sense, acquired a better understanding of the underlying features that guide musical style for a particular song.

8.3 Measuring stylistic differences across songs

As mentioned previously, we were motivated to explore the Maestro dataset using individual songs as meta-learning tasks in order to ensure greater consistency within a given task. Qualitatively, we observed the genre-based task definition used with the Lakh dataset to be stylistically broad, and Figure 3 revealed a quantifiable stylistic ambiguity between genres. Here, we perform the same analysis with the new song-specific task definition. Specifically, we compute a distribution for style metric consistency within each song/task in the test set (*intra*-task distribution), as well as a null distribution of style across all songs in the test set. Representative results are displayed in Figure 5. Encouragingly, in contrast to the nearly identical distributions observed in the genre-based task definition, here we see a distinction between the intra-task and inter-task style consistency. These results appear to confirm our intuition that defining tasks as individual songs confers greater stylistic consistency within that task.

8.4 Discussion

To summarize, we find that when using the finer task definition of individual songs in the Maestro dataset, our musical meta-learner performs better than the baseline when evaluated using qualitative



(a) Average Pitch Shift (PS)

(b) Total Used Pitch (TP)

Figure 5: Representative style metrics comparing the style consistency for the song-based tasks in the Maestro dataset. As before, the red trace describes the average intra-song style distribution, while the blue trace describes the *inter*-song style distribution. Grey traces are distributions for individual songs that have been averaged into the red trace. In contrast to the results from the genre-based task definition, here we see a distinction between the style *within* an individual song compared to styles across songs, suggesting that the tasks represent a more unique stylistic character.

metrics. Using the average overlapping area, we find that the meta-learning model better matches the underlying style of the provided task. However, when evaluated with negative log-likelihood, there is no substantial difference between the MAML and baseline models. Furthermore, adapting to the support set in the "Standard" setting does not provide a quantitative benefit to the models when compared against the "Zero-Shot" setting.

What might be the reason for this apparent discrepancy between a qualitative improvement and a quantitative stagnation? Perhaps the simplest explanation is that the negative log likelihood is not effectively capturing musical style. That is, despite a realized improvement in generating music of a target style, the negative log likelihood remains unaffected. An example with our specific encoding is particularly illustrative in this case. Suppose a model predicts a note with duration of 1 quarter note, but the ground truth has a duration of 1.125 quarter notes. In this case, the model would receive a substantial penalty from the negative log likelihood. Meanwhile, the actual musical difference would be minor and would be unlikely to affect any measures of underlying musical style. Given this potential discrepancy in the negative log likelihood objective, it is perhaps not too concerning that adaptation (in the "Standard" setting relative to the "Zero-Shot" setting) leads to worse performance in terms of negative log likelihood, considering the fact the MAML model performs better than the baseline in terms of style factors.

Even so, we can also entertain the possibility that both models truly perform worse following adaptation. In this case, why does the MAML model still perform qualitatively better than the baseline if it is not positively adapting to task information? One likely explanation is that the meta-learning objective forces the MAML model to place additional emphasis on larger structural features such as melodic and rhythmic motifs shared across reference snippets and that this increased structural competence is what enables it to produce samples that sound similar to those belonging to a particular piece. The baseline model, on the other hand, cannot adequately capture the necessary musical information in the few-shot training regime with a small model possessing limited computational capacity. Nonetheless, regardless of how we interpret the poorer Negative Log Likelihood following adaptation, we find that MAML has resulted in samples that are more aligned stylistically with the references.

9 Conclusion

In this report, we presented an initial exploration of the application of meta-learning to the production of music belonging to a style defined by a collection of samples. We evaluated a baseline pre-train and finetune model and MAML on two sets of experiments, one focused on the Lakh Midi dataset and the other focused on the Maestro dataset. Our results hint that meta-learning can provide one

approach to stylistic adaptation when the given reference samples have enough structural similarities between them.

Regardless, our report remains an initial investigation of a problem space that is mostly unexplored. Since most of our experiments were constrained by time and compute concerns, we would, if given the chance, repeat the experiments for the Maestro dataset using larger models and longer context sizes. Longer context sizes could provide the model with important stylistic information, such as the musical structure across phrases or the self-similarity exhibited by musical motifs. In addition to an exploration of more expressive models and hyperparameters, investigating different objective functions and/or encoding styles would be a fruitful extension. Our results suggest that negative log likelihood does not perfectly align with the objective of stylistic similarity, so there may exist more appropriate representations that would provide a more productive learning signal to the model.

10 Code Availability

All code for this project can be found at the following GitHub repository:
`https://github.com/schlagercollin/meta-learning-music`.

References

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel. Encoding musical style with transformer autoencoders. *arXiv preprint arXiv:1912.05537*, 2019.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Monica Dinculescu, Jesse Engel, and Adam Roberts. Midime: Personalizing a musicvae model with user data. 2019.

[6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. *arXiv preprint arXiv:1709.06298*, 2017.

[7] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. Technical report, 2002.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[9] Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. *arXiv preprint arXiv:1905.06118*, 2019.

[10] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[11] Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*, 2018.

[12] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. *arXiv preprint arXiv:1810.12247*, 2018.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*, 2019.

[15] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[17] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, 2020.

[18] Christine Payne. Musenet. *OpenAI Blog*, 2019.

[19] Colin Raffel. The lakh midi dataset v0. 1, 2016.

[20] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[22] Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, Peng Xu, and Pascale Fung. Meta-transfer learning for code-switched speech recognition. *arXiv preprint arXiv:2004.14228*, 2020.

[23] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, May 2020.