Adaptive Accompaniment with ReaLchords

Yusong Wu [1 2] Tim Cooijmans [2] Kyle Kastner [3] Adam Roberts [1] Ian Simon [1] Alexander Scarlatos [1 4] Chris Donahue [1 5] Cassie Tarakajian [1] Shayegan Omidshafiei [6 7] Aaron Courville [2 8] Pablo Samuel Castro [1 2] Natasha Jaques [1 9] Cheng-Zhi Anna Huang [1 2 8]

Abstract

Jamming requires coordination, anticipation, and collaborative creativity between musicians. Current generative models of music produce expressive output but are not able to generate in an *online* manner, meaning simultaneously with other musicians (human or otherwise). We propose `ReaLchords`, an online generative model for improvising chord accompaniment to user melody. We start with an online model pretrained by maximum likelihood, and use reinforcement learning to finetune the model for online use. The finetuning objective leverages both a novel reward model that provides feedback on both harmonic and temporal coherency between melody and chord, and a divergence term that implements a novel type of distillation from a teacher model that can see the future melody. Through quantitative experiments and listening tests, we demonstrate that the resulting model adapts well to unfamiliar input and produce fitting accompaniment. `ReaLchords` opens the door to live jamming, as well as simultaneous co-creation in other modalities.

1. Introduction

Deep generative models produce realistic, high-quality content, and are seeing increasing integration into the creative processes of artists. However, such models tend not to be designed for the demands of live scenarios such as interactive improvisation, which requires anticipation of others' intentions and adaptation to mistakes, stylistic choices and deliberate exploration. We focus on music in particular, which is inherently interactive and dynamic and revolves around anticipatory collaboration in ensemble settings. Most existing models in this space, while capable of creating expressive compositions or accompaniments, are not suited for simultaneous creation, where adaptation to and alignment with the ongoing musical structure are crucial.

This paper introduces `ReaLchords`, a generative model tailored for online adaptive musical accompaniment. Emulating the spontaneity of live music jamming, `ReaLchords` generates chord accompaniments in response to a stream of monophonic melody notes, adapting on-the-fly to the unfolding musical narrative. Each chord must be generated without knowing in advance which melody note it will accompany. This simultaneous interaction imposes a conditional independence assumption on the joint generative process, that an *online* model must respect. Moreover, a model must be able to gracefully handle unfamiliar situations and unexpected changes. Likelihood models, however, suffer from exposure bias due to being trained entirely on ground truth data, and transfer poorly to the online settings where mistakes, imperfections and stylistic differences are common (see Figure 1 for an example).

To address this, we use RL finetuning to improve the model with respect to reward models that consider musical coherence (§3.2). These reward models see the entire composition and evaluate its musical coherence from various perspectives (§3.3). Our setup bears similarities to RLHF (Ouyang et al., 2022; Jaques et al., 2019) and RLAIF (Bai et al., 2022; Lee et al., 2023), however our reward models are trained through self-supervision rather than human labeling. Finally, we combine RL finetuning with knowledge distillation (Agarwal et al., 2023; Zhou et al., 2023) in a novel way, distilling from a teacher that can see the future into a student that cannot, hence forcing anticipation (§3.4).

We develop key algorithmic components (Figure 2) needed to produce an online adaptive accompaniment model that is amenable to interactive use. Our contributions and findings are as follows: [1]

[1]Google DeepMind [2]Mila - Quebec AI Institute, Université de Montréal [3]Google [4]University of Massachusetts Amherst [5]Carnegie Mellon University [6]Work done while at Google [7]Field AI [8]Canada CIFAR AI Chair [9]University of Washington. Correspondence to: Yusong Wu <wu.yusong@mila.quebec>, Cheng-Zhi Anna Huang <anna.huang@mila.quebec>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Listen to audio examples here: `https://storage.googleapis.com/realchords/index.html`.

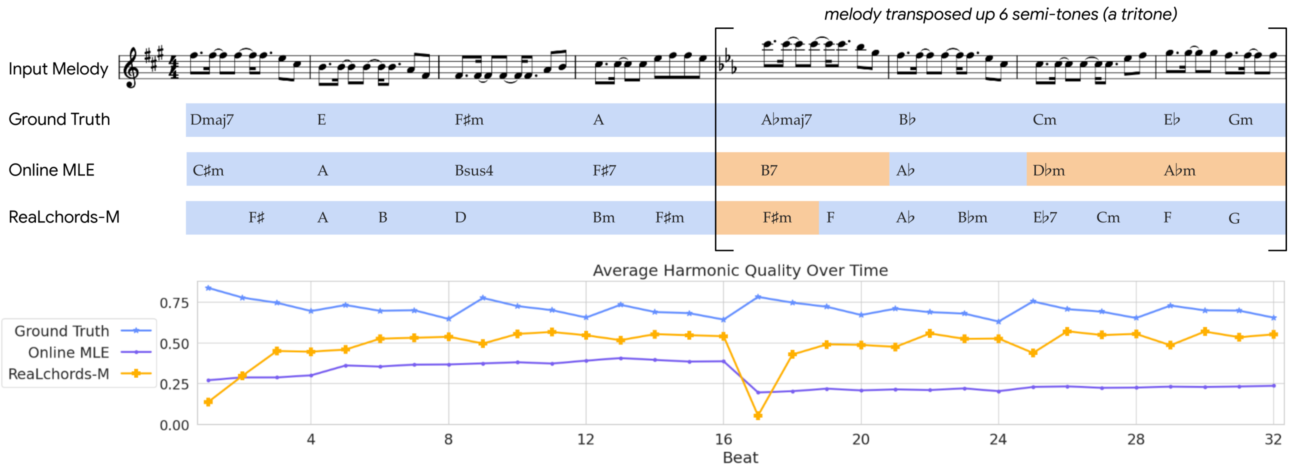


Figure 1. Online models finetuned with RL are able to recover from mistakes, while models trained with MLE alone do not. We take a melody from the test set and midway introduce an abrupt transposition designed to disrupt the accompaniment model (top row). The Online MLE model predicts a bad chord (B7) and fails to adapt. `ReaLchords` also predicts a bad chord (F♯m), but adapts quickly. Wrong chords highlighted in orange are our own judgment informed by music theory, but the overall pattern is corroborated by an objective measure of harmonic quality, averaged over many trials of this experiment (bottom row).

- ♯ We propose `ReaLchords`, an online accompaniment generation model trained by RL finetuning. Figure 1 shows how `ReaLchords` adapts to out-of-distribution input, a necessary skill for live jamming.

- ♯ We leverage knowledge distillation to learn from a non-causal teacher that can see the future (§3.4). Distillation greatly improves the quality of the model, as evidenced by the human evaluation shown in Figure 3.

- ♯ We further employ a novel set of self-supervised reward models to encourage musical coherence and perceptual quality (§3.3). Based on a human listening test, we show that our reward models align closely with human preferences (Figure 3), despite being trained without human feedback (§3.3).

- ♯ We demonstrate through a series of controlled experiments that without RL finetuning, models fail to adapt to mistakes and perturbations (Figure 4, §5.4).

- ♯ Finally, we analyze the behavior of our models in terms of domain-specific metrics (Table 1, §5.3). We find that each component in our RL finetuning methods improves the rhythmic and harmonic quality of generated accompaniments.

2. Related Work

Adaptive music accompaniment systems In contrast to automatic music generative systems, accompaniment systems often take input (such as melody) from a user, and generate output that is meant to be played in synchrony to complement what the user is playing. Some of these systems are asynchronous, where the user first provides the full melody, and the system generates an accompaniment offline. Examples include MySong (Simon et al., 2008), where a user sings a melody and the system generates chords to accompany them. Most recently, SingSong (Donahue et al., 2023) supports a very similar interaction, but generates full-band backing tracks. Both are offline systems.

In contrast, online accompaniment systems need to synchronize with user actions in real-time. Score-following is a special case where the system has the score, the full context of the content of what the musician will play, but still needs to follow along and infer *when* to play their own part. Music Plus One (Raphael, 2010) adapts its playback speed of an orchestral recording (without the soloist) to a soloist's expressive performance. Similarly, Antescofo (Cont, 2008) follows where a soloist is in a score and triggers live electronics accordingly.

Generative accompaniment systems or more generally co-creative music systems, not only have to anticipate user actions, they need to learn how to respond. Voyager (Lewis, 2003) takes a rule-based approach in how to listen, respond and generate musical material on the fly, while Omax Brothers (Assayag et al., 2006) recombines what a musician plays on-the-fly as an accompaniment but often requires another computer musician to control when it comes in and what section of material to draw from. ImproteK and later DJazz (Nika & Chemillier, 2012; Nika et al., 2017) leverages a shared predefined chord progressions (such as a Jazz Standard) to coordinate the human-machine improvisation. Instead of tight synchronization, Spire Muse (Thelle

& Pasquier, 2021) serves as a brainstorming partner which retrieves musical responses that are more or less similar depending on if the user is in a converging or diverging phase of ideation.

Recent systems based on deep neural networks have emerged. BachDuet (Benetatos et al., 2020) trains an LSTM model using MLE for counterpoint (melody to bassline) accompaniment. SongDriver (Wang et al., 2022) focuses on online melody-to-chord accompaniment, similar to our work. To address exposure bias, SongDriver employs two MLE-trained models: a transformer model that predicts current output based on both current and past outputs, and a conditional random field (CRF) model that predicts current output based on previous context. The CRF model makes online predictions but does not use its own predictions for future contexts; instead, it relies on the transformer model for context.

In contrast, our system `ReaLchords` learns how to respond and in tight synchronization with user melody, by first learning interdependencies between melody and accompaniment from existing songs, and then using RL to tune the models to respond in an adaptive fashion.

RL finetuning for generative models Reinforcement learning (RL) finetuning has proven effective in aligning language models with human preferences (Ouyang et al., 2022; Jaques et al., 2019) and constraints (Jaques et al., 2017), which are often unaddressed in generative pretraining. In some cases, RL finetuning has been applied to enhance music generation models (Jaques et al., 2017; Jiang et al., 2020b). Most closely related to our work is RL-Duet (Jiang et al., 2020b), which considers a similar online generation setting, namely a duet between a user and an agent, both of them playing each note without knowing what the other will play. Our work provides several contributions over RL-Duet. First, RL-Duet is trained on Bach Chorales, a small dataset of approximately 400 songs following strict rules of counterpoint composition in the style of a particular composer. In contrast, our models are trained on the diverse Hooktheory dataset of 38,000 popular songs from a wide array of artists. To enable effective learning on this scale, we develop novel multiscale contrastive and discriminative reward models, and also propose a new knowledge distillation technique specifically geared toward the online generation setting. Finally, RL-Duet experiments are limited to the setting in which the RL model is primed with the first few ground-truth notes of the accompaniment, an unrealistic assumption for real-time collaborative jamming. As we will show in §5.4, our methods are able to begin jamming with the user's melody within a few beats, and adapt to sudden perturbations in the key.

Our work is related to the emerging literature on Reinforcement Learning from AI Feedback (RLAIF) (Saleh et al., 2020; Bai et al., 2022; Lee et al., 2023), which mitigates the need for extensive human labeling by utilizing an AI assistant for feedback generation. We use this strategy to finetune online music language models, using an MLE model to obtain a learning signal. Recently, Agarwal et al. (2023) have shown that adding a distillation objective between the policy and a larger teacher model during RL finetuning further improves performance. `ReaLchords` employs a novel knowledge distillation objective between the online policy and an offline model which can see future context, bridging the gap between online improvisational capabilities and offline musical coherence.

3. Online Musical Accompaniment

We seek a generative model that can be used for interactive music accompaniment, where a user plays a melody, and the model simultaneously plays chords to support it. Accompaniment is a special case of the general setting in which two agents generate a joint sequence $(x_1, y_1), \ldots, (x_T, y_T)$ in chronological order. At each step t, agents observe the historical material $x_{<t}, y_{<t}$, and simultaneously emit the next pair of tokens x_t, y_t. Simultaneity imposes a conditional independence on the generative process:

$$\Pr(x_t, y_t \mid x_{<t}, y_{<t}) = \Pr(x_t \mid x_{<t}, y_{<t}) \Pr(y_t \mid x_{<t}, y_{<t}).$$

In this general setting, the melody x and chords y interdepend through the conditioning on shared history $x_{<t}, y_{<t}$; this corresponds to musicians adapting to each other as they play. As a first step, we consider the specific setting where the chords do not influence the melody; now one player leads and the other follows. We call this accompaniment.

We approach this problem by constructing a model π_θ that generates accompaniment y according to a specific autoregressive process:

$$\pi_\theta(y \mid x) = \prod_t \pi_\theta(y_t \mid x_{<t}, y_{<t}). \tag{1}$$

While our goal at each timestep t is to predict a chord y_t that supports the melody token x_t about to be played, the model's prediction of y_t does not depend on x_t. This is crucial, as it allows the model to be used online as desired.

We train this model in two steps: pretraining on data (§3.1), followed by finetuning using reinforcement learning (§3.2). In the rest of this section, we first describe the general approach, and then detail on the components involved (reward models §3.3, distillation §3.4, and regularizations §3.5).

3.1. Maximum Likelihood Pretraining

The first step in training π_θ is to apply MLE, maximizing the data likelihood with respect to θ:

$$\max_\theta \mathop{\mathbb{E}}_{x,y \sim p_{\text{data}}} \log \pi_\theta(y \mid x).$$

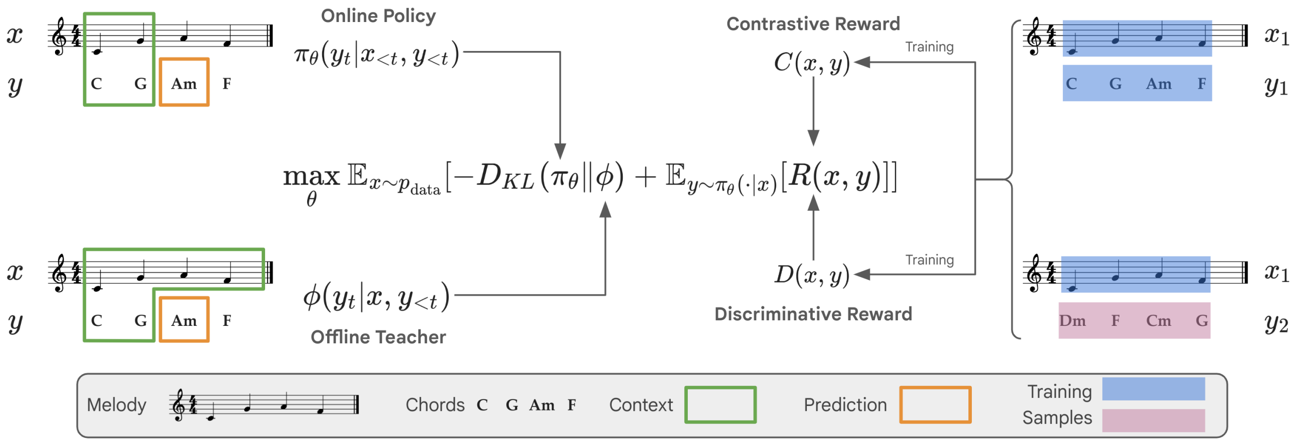


Figure 2. `ReaLchords` leverages RL finetuning to learn anticipation and adaptation for online melody-to-chord accompaniment. Initializing from a model π_θ pretrained by MLE, the policy generates a complete chord response to a melody from the dataset, each chord being predicted given only previous melody and chords (top left). In contrast, the offline model ϕ_ω (also trained by MLE) predicts each chord given the complete melody (bottom left). A KL-divergence penalty distills the predictions of the offline model into the online model, improving its ability to anticipate the future. (Right) The reward stems from an ensemble of multi-scale contrastive and discriminative models that evaluate the musical coherence between melody and chord. The final training objective in `ReaLchords` is a sum of the reward and the distillation loss (center).

The data distribution p_{data} can be interpreted as standing in for p_{user}: we simulate user play by sampling fixed melodies from the dataset. This limits our ability to encourage and assess the model's ability to adapt to out-of-distribution melodies. Nevertheless, the model will still encounter out-of-distribution combinations of melodies and chords during inference.

Unfortunately, applying only MLE training to **online accompaniment model** suffers from exposure bias (Arora et al., 2022): during training, the model is always conditioned on ground-truth context, but this does not occur during inference. Consequently, MLE models struggle to learn two skills required in online accompaniment (Jiang et al., 2020b;a). First, the model must *anticipate* what the user is going to play, in order to ensure that its own output agrees with that of the user. Second, the model must be able to *adapt* to and recover from unknown input, due to its own mistakes or those of the user, due to misanticipation, or due to user idiosyncrasies.

As a concrete example, Figure 1 shows a failure mode of the online MLE model. The model fails to adequately anticipate future inputs, leading to exposure bias and error accumulation due to a distribution mismatch between training and inference. Whenever the first few time-steps of output do not fit with the melody input stream the model will continue its chord progression, *ignoring the input*.

3.2. Finetuning using Reinforcement Learning

Similar challenges are encountered in imitation learning (Ross & Bagnell, 2010), where policies trained by MLE to reproduce expert demonstrations are brittle, and fail to transfer to the real environment (see e.g. Reddy et al. (2019)). A rich history of work has demonstrated Reinforcement Learning (RL) finetuning to be an effective remedy.

We begin by initializing the weights of our RL policy π_θ with those of the pretrained online MLE model. As in eq. 1, at timestep t, the policy predicts action probability distribution $a_t = y_t$ given state $s_t = (x_{<t}, y_{<t})$. Then, we adopt an RL finetuning methodology similar to the popular RLHF (RL from Human Feedback) framework used for language models (Ouyang et al., 2022; Jaques et al., 2019). Namely, in addition to maximizing RL rewards $R(x, y)$, we minimize KL-divergence from a pretrained MLE anchor model $\phi_\omega(y|x)$ parameterized by ω, as proposed in Jaques et al. (2017). Let x and y represent the full melody and chord sequence, each consisting of several tokens (i.e. the full *trajectory*). This gives us the KL-regularized RL objective:

$$\max_\theta \mathop{\mathbb{E}}_{\substack{x\sim p_{\text{data}}\\ y\sim\pi_\theta(\cdot|x)}} R(x,y) - \beta D_{KL}(\pi_\theta(\,\cdot\mid x) \parallel \phi_\omega(\,\cdot\mid x)). \tag{2}$$

To evaluate (2), we sample a batch of melodies x from the training set, then use the current policy π_θ according to (1) to generate a batch of corresponding harmonies y (Figure 2, top left). We then evaluate the resulting batch of compositions (x, y) according to reward models (§3.3) and regularizers (§3.5) to obtain the reward $R(x, y)$ (Figure 2, top and bottom right). Additionally, we measure $\phi_\omega(y \mid x)$ under the offline model ϕ_ω (§3.4) in order to compute the KL term (Figure 2, bottom left). Finally, we update the model according to (2), using REINFORCE with a separate value model serves as baseline estimation for improved stability (Lee et al., 2023; Agarwal et al., 2023). The separate value model is also initialized from pretrained online MLE model, and is trained to estimate the total return. We use mean square error between the estimated return and total return as objective to train the value model.

Unlike in RLHF (Ouyang et al., 2022) and RLAIF (Bai et al., 2022), our reward models are not trained on preference labels from either human or machine labelers. Instead, they are trained using positive and negative melody-chord pairs constructed from a dataset (see Figure 2, §3.3). Nevertheless, a listening test (§5.1) shows that our reward models align well with human preferences, as shown in Figure 3.

3.3. Reward Models

We develop a novel ensemble of reward models that evaluates the coherence between input (melody) and output (chord) tracks. We implement two types of coherence evaluation reward models, contrastive and discriminative, each with different inductive biases. Reward model training and architectural details can be found in Appendix §F and §G.

The **contrastive reward model** consists of a melody encoder and a chord encoder, which respectively map the melody x and chord y to embedding vectors E_x, E_y. The encoders are trained in an unsupervised manner using InfoNCE loss (Oord et al., 2018; Radford et al., 2021) applied to positive and negative samples created from the dataset. As shown in Figure 2, the positive pairs are defined as the melody-chord pairs from the same song, and the negative pairs are created by randomly pairing melody and chord from different songs. The InfoNCE loss essentially maximizes the cosine similarity for positive pairs, and minimizes the cosine similarity for negative pairs. The reward for a given pair x, y is the cosine similarity of E_x and E_y.

The **discriminative reward model** looks at the entire generated pair (x, y) as a whole. This model is trained in an unsupervised manner to discriminate between "real" melody-chord pairs and randomly paired melodies and chords. Each training batch case provides a set of positive pair and, by combining its melody with the chords from another randomly chosen batch case, a negative pair. Once trained, the model produces a probability of (x, y) being "real" that directly serves as the reward.

Due to the bottleneck on the embedding vectors E_x, E_y, the contrastive models focus on global coherence. The discriminative models on the other hand are able to evaluate temporally local coherence. Indeed, our experiments in §5.3 show that contrastive reward models promote mainly harmonic quality whereas discriminative reward models encourage mainly synchronization.

While these reward models are effective, we find that they can be overly harsh on temporally localized incompatibilities, such as short-lived mistakes that are quickly resolved. To mitigate this and improve temporal credit assignment, we further propose to use an ensemble of *multi-scale* variants that evaluate isolated fragments without being influenced by distant mistakes. We train multiple contrastive and discriminative reward models to judge fragments of reduced lengths ($\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ of the maximum sequence length 256). During finetuning, we apply these models to sliding windows (50% overlap) of the example.

3.4. Distillation from Offline Teacher Model

As stated in §3.2, during RL finetuning we penalize KL-divergence from a model pretrained on the data distribution to ensure the model maintains realistic outputs while maximizing rewards (Jaques et al., 2017). However, unlike in typical RL finetuning, the online MLE model with which our policy is initialized suffers from a lack of robustness to out-of-distribution data, and as such is not an ideal anchor for use with the KL-regularization term. Agarwal et al. (2023) demonstrated how the KL penalty can be used not just to avoid diverging from a checkpoint, but also to distill knowledge from a larger teacher model. We take this idea one step further and distill knowledge from an *offline* model that can see the future of the melody.

The offline model ϕ is trained with MLE to autoregressively predict chords given the full melody x:

$$\phi_\omega(y \mid x) = \prod_t \phi_\omega(y_t \mid x, y_{<t}). \tag{3}$$

In traditional knowledge distillation, ground truth data is used to obtain the predictions of both the teacher and student models, and a KL loss is then applied to bring the student's predictions closer to the teacher's. Here, it is instead evaluated on samples generated by the current policy. This is a special case of *on-policy knowledge distillation* (Agarwal et al., 2023; Zhou et al., 2023), which in general allows any mixture of ground truth data, student samples and teacher samples. We tested various on-policy knowledge distillation schedules and found it works best when driven by the student (§5.3). Thus, during RL finetuning we only train on outputs from the student.

3.5. Regularization Penalties

RL finetuning can lead to pathological behavior, such as repetitions and mode collapse (Jaques et al., 2017; Jiang et al., 2020b). We introduce three regularization penalties

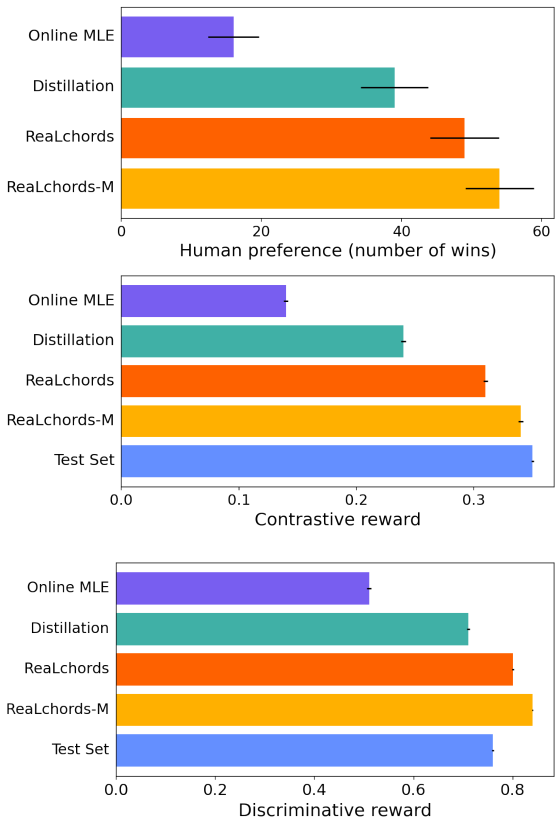


Figure 3. Our reward models are aligned with human preferences. We carried out a listening test (§5.1) to evaluate the quality of our models. The online MLE model performs poorly, but is greatly improved by distillation from the offline MLE model. Our proposed systems `ReaLchords` and `ReaLchords`-M improve further thanks to RL finetuning. The rewards given by both the contrastive and discriminative reward models are strongly correlated with human evaluations.

to discourage specific failure modes: **Repetition**: Inspired by repetition penalties used for training language models (as in Saleh et al. (2020); Jaques et al. (2019)), we impose a penalty for chords that are held for too long. **Silences**: We impose a penalty for silences beyond the beginning of a phrase. **Ending early**: A penalty is imposed for early end-of-sequence (EOS) tokens. See §D for an ablation that shows the need of these penalties.

4. Dataset

We train our models on an updated version of the Hooktheory dataset (Donahue et al., 2022), which comprises crowdsourced analyses of monophonic melodies and chords from recordings and now contains 38K melody-chord pairs. We adopt a frame-based representation where time is quantized to sixteenth notes, where each frame is a discrete index. We set a maximum sequence length of 256 for x and y. We augment the data by randomly transposing up or down by up to 6 semitones. 20% of the data is held out and divided equally into validation and test sets. We develop on the validation set and report the test set results in the paper. Please refer to §L for details on the dataset and data representation.

5. Experiments

Is the system capable of producing accompaniments of high musical quality? How swiftly can the system adjust to unfamiliar situations? We address these questions from three directions.

To directly assess musical quality, we conduct a human listening test using samples generated from the models (§5.1). We demonstrate adaptation through several controlled generation experiments, tracking the quality of the accompaniment over time (explained in §5.4). Finally, we evaluate the system using heuristic metrics to assess the quality of compositions generated in response to melodies in the test set (detailed in §5.3).

The following systems are compared in our experiments:

MLE baselines The **Online MLE** model trained to predict $y_t \mid x_{<t}, y_{<t}$ without seeing x_t (§3.1). The **Offline MLE** model that sees the full input x and is used as a teacher for knowledge distillation (§3.4).

Our proposals These models are trained with both contrastive and discriminative rewards, as well as regularization and knowledge distillation. `ReaLchords` incorporates the global reward models, whereas `ReaLchords`-M incorporates the multi-scale variants of both reward models.

Ablations The model **KD**, trained with only knowledge distillation and regularization. Two models trained by MLE and then finetuned using either only **Contrastive (C)** reward or only **Discriminative (D)** reward, with regularization and KL divergence to the MLE checkpoint. A model **C+D** using both contrastive and discriminative reward, with regularization and KL divergence to the online MLE checkpoint.

5.1. Human and Machine Evaluation on Musicality

Any measure of the quality of a musical piece must ultimately be grounded in human preferences. We carry out a listening test to evaluate four systems: the Online MLE baseline, KD, `ReaLchords` and `ReaLchords`-M. In the listening test, participants are presented with 8-second audio clips from two different systems, and asked to rate which one sounded more musical, on a 5-point Likert scale. We recruited ten musicians, and collected 192 ratings with each system involved in 96 pairwise comparisons (see §J for more details).

Figure 3 (top) shows the number of wins for each sys-

Table 1. Effect of RL finetuning with our reward models and knowledge distillation on harmonic, synchronization and rhythmic diversity metrics. Each number is an average over a large number of accompaniments to test set melodies. In row 2-7, we report confidence interval (95%) of metric values over 3 RL finetunings, each with different random seeds.

Models	Harmony Note in Chord ↑, %	Synchronization Δ Chord-Note Onset Interval ↓, $\times 10^{-3}$	Rhythm Diversity Chord Length Entropy ↑
Online MLE	36.99	14.23	2.21
Knowledge Distillation (KD)	46.38 ± 0.45	14.39 ± 1.25	**1.80** ± 0.04
Contrastive (C)	47.22 ± 1.22	16.61 ± 4.78	1.80 ± 0.29
Discriminative (D)	44.29 ± 0.99	**12.34** ± 7.95	1.70 ± 0.06
C + D	46.12 ± 0.95	12.86 ± 6.17	1.56 ± 0.05
`ReaLchords`	48.17 ± 0.27	16.09 ± 3.63	1.35 ± 0.30
`ReaLchords`-M	**54.29** ± 1.55	17.17 ± 4.86	1.66 ± 0.20
Offline MLE	63.73	9.85	1.90
Test set	70.94	0.00	2.19

tem. We ran a Kruskal-Wallis H test and confirmed that there are statistically significant pairs among the permutations. According to a post-hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction (with $p<0.05/6$ as there are 6 pairs of systems), we found the following statistically significant results: All systems outperformed the Online MLE baseline. Also, the fully-fledged systems `ReaLchords` and `ReaLchords`-M outperformed distillation alone (KD). While `ReaLchords`-M appears to outperform `ReaLchords`, this comparison is not significant.

Overall, the results from the listening test show that distillation alone (KD) accounts for a large improvement in perceptual quality. The reward models agree with this assessment, even though KD does not directly optimize for these rewards. In general, we find that the rewards given by our self-supervised reward models (Figure 3, middle and bottom) correlate strongly with human preferences, which justifies their use in lieu of human feedback.

5.2. Quantitative Metrics

In line with prior research (Jiang et al., 2020b; Yang & Lerch, 2020; Fang et al., 2020), we introduce quantitative metrics to evaluate the quality of accompaniments:

Harmonic quality We measure harmonic quality by the note-in-chord ratio, which is the amount of time that the melody's pitch class occurs in the chord. For example, if the melody token x_t is a C, and the chord y_t is F minor, then the note-in-chord ratio as time t equals 1. We average this metric across time t and across all compositions x, y generated, to obtain the overall note-in-chord ratio for the model in question.

Synchronization To gauge temporal synchronization between melody and chord progression, we look at *chord-to-note onset interval*, which is the length of time between the onset of a chord and the onset of the nearest preceding melody note. The synchronization of a model can be judged by comparing this quantity's distribution on the test set versus on the output of the model. Whereas Jiang et al. (2020b) compare averages of this quantity, we propose to compare the full distributions using Earth Mover's Distance (EMD) on histograms of chord-to-note onset intervals.

Rhythmic diversity We examine the distribution of durations of generated chords to assess overall rhythmic behavior. The entropy of this distribution measures rhythmic diversity.

5.3. Quantitative Evaluation Results

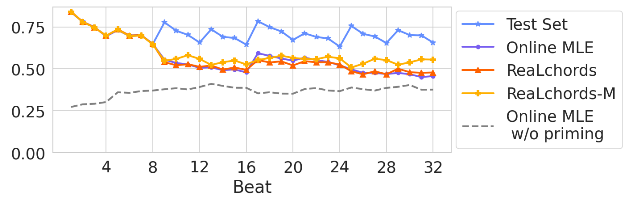
We evaluate each model based on a large number of accompaniments to test set melodies. The average metrics are reported in Table 1.

MLE baselines The behavior of Offline MLE is closest to that of the test set, as is expected due to its ability to see the future input. Online MLE exhibits poor harmonic and temporal coordination with the melody, which suggests that it produces chords without paying attention to the melody.

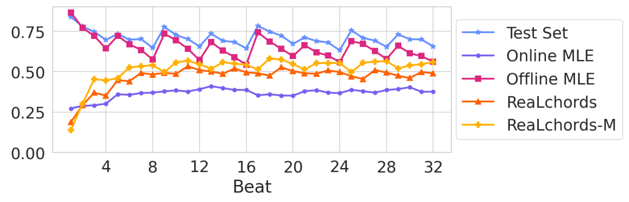
Distillation On-policy knowledge distillation (KD in Table 1) significantly enhances online generation, particularly with regard to harmony and synchronization. Distillation from the offline teacher will suppress the probability of chords that match poorly with the future melody, forcing the student to anticipate the immediate future. The student (KD) learns to produce outputs that align better with the input context while retaining a causal conditioning structure.

Reward models Training with contrastive and discriminative reward models, individually (C, D) and combined (C + D), shows distinct improvements in harmony and synchronization. The use of the contrastive reward model (C) improves more on harmony, presumably because it compresses the entire melody and the entire accompaniment separately

(a) Accompaniment quality when primed with ground truth.



(b) Accompaniment quality after a cold start.



(c) Accompaniment quality when perturbed midway.

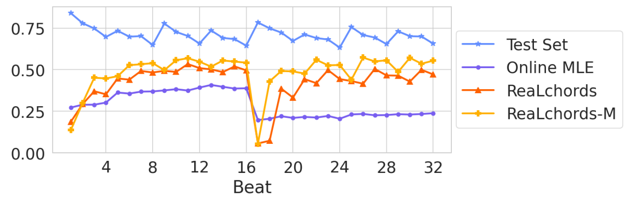


Figure 4. Comparing the quality of overall accompaniment as a function of the number of beats generated, in three scenarios of increasing difficulty (§5.4). Quality is measured by note-in-chord ratio. (a) Priming the online model with ground-truth context (8 beats in this case) results in comparable performance between models. (b-c) `ReaLchords` and `ReaLchords`-M recover from cold starts and perturbation, while the online model does not.

and merges them only in the final cosine similarity. The use of the discriminative reward model (D) improves more on synchronization, while having worse harmony. This is expected, as the classification is biased towards direct comparison. We further examine the bias of different reward models by plotting reward values against harmonic perturbations, where varying portions of chords in the test set are replaced with random alternatives. As shown in Figure 6 in §G, the contrastive model is more sensitive to harmonic perturbations.

The blend of both rewards in `ReaLchords` offers enhancement in each metrics. Additional experiments applying RL fine-tuning with ensembles of the same type of reward models show similar metric improvements, as presented in Table 6 in §I. This suggests that the observed metric enhancements may result from both the combined biases and the ensemble of reward models.

Combining rewards with distillation Integrating both reward models with knowledge distillation yields better harmony but less rhythmic diversity (`ReaLchords` in Table 1). This indicates a tendency of the model to opt for ‘safer’ chord progressions, presumably resulting from satisfying both reward maximization and knowledge distillation. This is further validated in Figure 8 in Appendix §M where we visualize the chord length histograms and find that this model tends to hold chords for 2 or 4 beats.

Multi-scale reward models `ReaLchords`-M further improves harmonic quality thanks to the locally isolated rewards from the multi-scale variants of our reward models. This aligns well with the findings in Figure 3.

5.4. Adaptation Dynamics

To study the temporal dynamics of model adaptation to unknown input, we measure accompaniment quality as a function of number of beats generated. A beat is 4 frames with a total of a quarter note length. We compare among the dataset and four models: Online MLE, Offline MLE, `ReaLchords` and `ReaLchords`-M. We report harmonic quality in terms of note-in-chord ratio. In all experiments, we draw melodies from the test set, and let the models generate accompaniment. However, we consider three different scenarios with different interventions: priming, cold-start, and perturbation.

Priming (Figure 4a): We start with the setting of RL-Duet (Jiang et al., 2020b), where the models are primed with several beats of ground truth chords before generating their own chords. This avoids the cold-start problem of predicting a chord without knowing anything about the melody, and gives an indication of the model’s ability to anticipate what happens next without having to first adapt to what happened previously. Behavior is similar across the models. For reference, we also plot the cold-start behavior of Online MLE (without priming), which is significantly worse. We argue that, as a benchmark for online accompaniment, primed generation is unnatural and too facile.

Cold start (Figure 4b): We now proceed to the cold-start setting, which is more natural and more difficult. Here, models predict chords immediately and have to adapt to the resulting melody-chord combinations, which are usually wrong and outside of the data distribution. The Online MLEstruggles to adapt to its own past chords, and never gets close to the primed behavior. `ReaLchords` and `ReaLchords`-M quickly overcome their own mistakes and play as well as if they were primed.

Perturbation (Figure 4c): Finally, we introduce a deliberate perturbation in the middle of the generation process, to demonstrate the ability of our systems to recover from serious errors. We transpose the melody up by a tritone (6 semitones) at beat 17, resulting in both an out-of-distribution melody and almost guaranteeing that the next chord is a poor fit. This is similar to the push test in legged robotics. The Online MLE fails the test: it exhibits a drop in harmonic quality and never recovers. `ReaLchords` and `ReaLchords`-M quickly adapt to the new key and recover their previous performance.

Overall, these results confirm that Online MLE suffers from exposure bias due to only being trained on ground-truth data. This brittleness, or inability to produce reasonable output given out-of-distribution (OOD) input not covered by the training data, is similar to the failures exhibited by imitation learning or behavior cloning methods in traditional RL contexts (Reddy et al., 2019), which also rely purely on supervised learning. Our systems `ReaLchords` and `ReaLchords`-M quickly recover from both cold-start situations and mid-song disturbances, which highlights their ability to follow along with a user as they explore ideas. Thus, `ReaLchords` solves a critical requirement for an online accompaniment system, which will have to accompany a diverse range of human users who are likely to play novel melodies not covered by the training data and change what they play midway.

Finally, we find an interesting emergent behavior due to RL finetuning, where models hold off on playing chords initially, preferring instead to wait for more information about the melody. This *wait and see* behavior is also visible in Figure 1, and examined further in Appendix §A. Similar behavior occurs in human performers, who often wait for several bars when improvising with an unfamiliar player.

6. Conclusion

We proposed `ReaLchords`, an online generative model that improvise simultaneous chord accompaniment in response to melody input. `ReaLchords` leverages RL finetuning with multi-scale contrastive and discriminative reward models and employs a novel offline-to-online knowledge distillation technique. Throughout our experiments, we show that `ReaLchords` accompanies with good harmony and synchronization, while effectively adapting to mistakes and perturbations. We also show that listeners preferred `ReaLchords` over the online MLE baseline and distillation-only models, while validating the proposed reward models align with human judgement. `ReaLchords` enables an exciting path towards an effective and engaging real-time, interactive music accompaniment system.

Acknowledgement

We would like to express our gratitude to the following individuals for their insightful discussions and valuable advice on the project: Jesse Engel, Ethan Manilow, Antoine Caillon, Laura Graesser, Athul Jacob, Kory Mathewson, Max Schwarzer, Evgenii Nikishin, Zhixuan Lin, Dinghuai Zhang, Michael Noukhovitch, Ke Chen, Yuxuan Wu, Yi Deng. We also like to thank the individuals who designed and built the RL training infrastructure used in this paper: Léonard Hussenot, Johan Ferret, Robert Dadashi, Geoffrey Cideron, Alexis Jacq, Sabela Ramos, Piotr Stanczyk, Sertan Girgin, Danila Sinopalnikov, Amélie Héliou, Bobak Shahriari, Bilal Piot, Matt Hoffmann, Nikola Momchev, and Olivier Bachem.

Impact Statement

While this work is developed in the domain of music, the interactive dynamic of simultaneous musical accompaniment has broader societal implications beyond the realm of music generation. This research could also influence the way AI is perceived in creative fields, reinforcing the potential of AI as a collaborative tool rather than a replacement for human creativity.

References

Agarwal, R., Vieillard, N., Stanczyk, P., Ramos, S., Geist, M., and Bachem, O. GKD: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 2023.

Arora, K., Asri, L. E., Bahuleyan, H., and Cheung, J. C. K. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. *arXiv preprint arXiv:2204.01171*, 2022.

Assayag, G., Bloch, G., Chemillier, M., Cont, A., and Dubnov, S. Omax brothers: a dynamic yopology of agents for improvization learning. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Benetatos, C., VanderStel, J., and Duan, Z. Bachduet: A deep learning system for human-machine counterpoint improvisation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2020.

Cont, A. Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *International Computer Music Conference (ICMC)*, 2008.

Donahue, C., Thickstun, J., and Liang, P. Melody transcription via generative pre-training. In *Proceedings of ISMIR 2022*, 2022.

Donahue, C., Caillon, A., Roberts, A., Manilow, E., Esling, P., Agostinelli, A., Verzetti, M., Simon, I., Pietquin, O., Zeghidour, N., et al. Singsong: Generating musical accompaniments from singing. *arXiv preprint arXiv:2301.12662*, 2023.

Fang, A., Liu, A., Seetharaman, P., and Pardo, B. Bach or mock? a grading function for chorales in the style of js bach. *arXiv preprint arXiv:2006.13329*, 2020.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pp. 1645–1654. PMLR, 2017.

Jaques, N., Shen, J. H., Ghandeharioun, A., Ferguson, C., Lapedriza, A., Jones, N., Gu, S. S., and Picard, R. Human-centric dialog training via offline reinforcement learning. *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Jiang, N., Jin, S., Duan, Z., and Zhang, C. When counterpoint meets chinese folk melodies. *Advances in neural information processing systems*, 33:16258–16270, 2020a.

Jiang, N., Jin, S., Duan, Z., and Zhang, C. Rl-duet: Online music accompaniment generation using deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 710–718, 2020b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Lewis, G. E. Too many notes: Computers, complexity, and culture in voyager. In *New Media*, pp. 93–106. Routledge, 2003.

Nika, J. and Chemillier, M. Improtek: integrating harmonic controls into improvisation in the filiation of omax. In *International Computer Music Conference (ICMC)*, pp. 180–187, 2012.

Nika, J., Chemillier, M., and Assayag, G. Improtek: introducing scenarios into human-computer music improvisation. *Computers in Entertainment (CIE)*, 14(2), 2017.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Raphael, C. Music plus one and machine learning. In *ICML*, 2010.

Reddy, S., Dragan, A. D., and Levine, S. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Roberts, A., Chung, H. W., Mishra, G., Levskaya, A., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A., et al. Scaling up models and data with t5x and seqio. *Journal of Machine Learning Research*, 24(377):1–8, 2023.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

Saleh, A., Jaques, N., Ghandeharioun, A., Shen, J., and Picard, R. Hierarchical reinforcement learning for open-domain dialog. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8741–8748, 2020.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Simon, I., Morris, D., and Basu, S. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008.

Thelle, N. J. and Pasquier, P. Spire muse: A virtual musical partner for creative brainstorming. In *NIME 2021*. PubPub, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Z., Zhang, K., Wang, Y., Zhang, C., Liang, Q., Yu, P., Feng, Y., Liu, W., Wang, Y., Bao, Y., et al. Songdriver:

Real-time music accompaniment generation without logical latency nor exposure bias. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 1057–1067, 2022.

Yang, L.-C. and Lerch, A. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

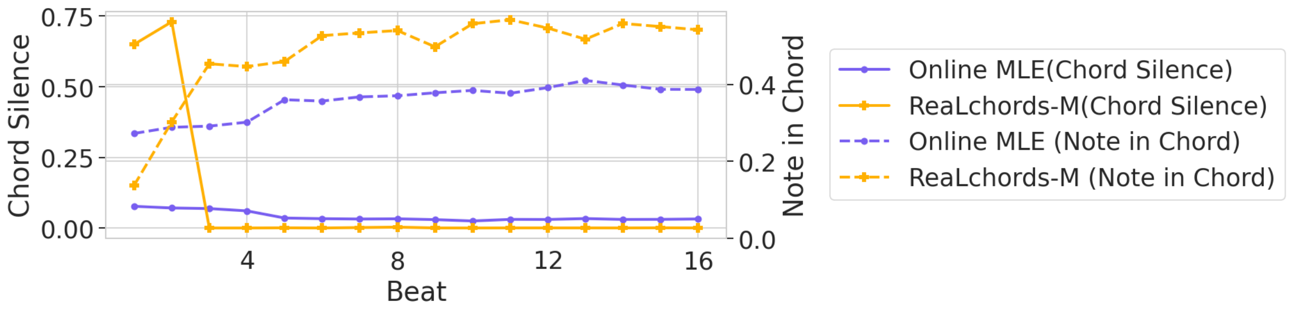


Figure 5. Chord silence ratio within each beat of the generation.

A. Wait and See Behavior

To further investigate the *wait and see* behavior noted at the end of §5.4, we measure ratio of chord silence during a non-silence note across each beat. Figure 5 shows that the finetuned models `ReaLchords` and `ReaLchords`-M, are often silent during the first few beats. We believe this behavior results from trading off the penalty for silence and the low reward for bad guesses.

B. Knowledge Distillation Schedules

On-policy knowledge distillation (Agarwal et al., 2023; Zhou et al., 2023) allows the KL to be evaluated with respect to any mixture of policy, teacher and data distribution. We compare these options and report results in Table 2. Sampling from the policy alone results in good performance, and we use this choice in all of our experiments.

C. Details of Regularization Penalties

RL finetuning can lead to pathological behavior, such as repetitions and mode collapse (Jaques et al., 2017; Jiang et al., 2020b). We introduce several regularization penalties to discourage specific failure modes.

Repetition The model has a tendency to repeat itself, generating long-held chords. Inspired by repetition penalties used for training language models (as in Saleh et al. (2020); Jaques et al. (2019)), we provide a penalty of -1 for each token in a chord that is held for longer than 32 frames (8 beats), because chords longer than 8 beats rarely occur in the training set.

Silence To avoid behavior where no chords are generated, we provide a penalty of -1 for each silence token if more than 4% of frames are silent accompaniment to a non-silent input. This is the rate at which silent accompaniment occurs in the training set. We omit this penalty for the first 8 frames (first half-note) to allow early adaptation.

Ending early To discourage early end-of-sequence (EOS) tokens, we provide a penalty of -1 for each token after model outputs EOS but before the input does. The penalties are given to the whole sequence at the last token of output before EOS. An ablation experiment in §D demonstrates the need for these penalties.

D. Ablation on Regularizers

To show the need for regularization penalties, we conduct an ablation test. As shown in the Table 3, removing the repetition penalty results in significantly more long chords. Removing the silence penalty would result in a long chunk of silence at the beginning of the generation, while removing the early-finish penalty would result in high ratio of samples stop early.

E. Training and Architecture Details of Online and Offline Models

The online model π_θ is implemented as an 8-layer autoregressive decoder-only transformer (Vaswani et al., 2017) with 6 heads and hidden dimension of 512. It is trained on interleaved input and output tokens $y_1, x_1, \ldots, y_T, x_T$, providing conditional distributions of the form $\pi_\theta(y_t \mid x_{<t}, y_{<t})$ and $\pi_\theta(x_t \mid x_{<t}, y_{\leqslant t})$. To use this model *online* as per (1), we alternate drawing y_ts from $\pi_\theta(y_t \mid x_{<t}, y_{<t})$ and x_ts from the given melody, foregoing the use of the conditionals

Table 2. Comparing different knowledge distillation schedule, generating with policy data achieves best balance between harmonic, synchronization and rhythmic diversity metrics.

Models	Melody Note in Chord Ratio (harmony) $\uparrow$	Δ Chord-Note Onset Interval $\times 10^{-3}$ (synchronization) $\downarrow$	Chord Length Entropy (rhythm diversity) $\uparrow$
Online MLE	36.99	14.23	2.21
Policy (KD)	46.54	12.81	1.79
Dataset	35.25	19.58	2.05
Dataset + Policy	46.28	13.65	1.67
Dataset + Policy + Teacher	46.64	13.39	1.63
Offline MLE	63.73	9.85	1.90
Test set	70.94	0	2.19

Table 3. Comparing quantitative metrics of ablation experiments on regularization penalties, the proposed penalties effectively prevent RL training to exploit reward which result in sub-optimal solution. Details in §D.

Models	Note-in-Chord Ratio (%)	Chord-Note Onset Interval	Chord Length Entropy	Chord Silence Ratio (%)	Long Chords Ratio (%)	Early Stop Ratio (%)
Online MLE	36.99	14.23	2.21	7.31	1.67	21.66
Offline MLE	63.73	9.85	1.90	2.91	0.73	2.62
Test set	70.94	0	2.19	2.83	0.71	0
KD	46.54	12.8	11.79	4.55	1.51	0
ReaLchords	48.17	13.01	1.25	4.90	0.02	0
KD w/o all penalties	45.96	14.98	1.66	32.67	0	0.38
ReaLchords w/o repetition penalty	48.19	13.17	1.31	4.81	0.78	0
ReaLchords w/o repetition penalty w/o silence penalty	46.38	11.12	1.43	26.50	0.32	0
ReaLchords w/o all penalties	46.92	11.55	1.46	27.85	0.29	0

$\pi_\theta(x_t \mid x_{<t}, y_{\leqslant t})$. The online model is trained using Adafactor optimizer (Shazeer & Stern, 2018) and learning rate of 10^{-3} with a batch size of 256. The online model is trained for $50,000$ steps with 1000 steps of warmup. We apply a dropout with rate 0.1 to the online model during training.

The offline model ϕ_ω is implemented as an 8-layer encoder-decoder transformer in the style of T5 (Raffel et al., 2020) with 6 heads and hidden dimension of 512. This model is trained on pairs of sequences x, y without interleaving, by first encoding the entire melody x and then generating the entire chord progression $y \mid x$ conditioned on the encoding of x. The offline model is trained using Adafactor optimizer (Shazeer & Stern, 2018) and learning rate of 10^{-3} with a batch size of 256. The offline model is trained for $50,000$ steps with 1000 steps of warmup. We apply a dropout with rate 0.1 to the offline model during training. We train our online and offline transformers using the T5X framework (Roberts et al., 2023).

F. Training and Architecture Details of Reward Models

The training specification is identical for reward models at all scale, with the only difference lies in their input length. Our contrastive reward model consists of two identical 6-layer, 6-head transformer encoders with 512 hidden dimension, one for encoding the entire melody x and one for encoding the entire chord progression y. The contrastive reward model is trained using Adam optimizer (Kingma & Ba, 2014) and learning rate of 10^{-4} with a batch size of 128. The contrastive reward model is trained for 35 epochs with a dropout rate of 0.1.

The discriminative reward model consists of a 6-layer, 6-head transformer encoder with 512 hidden dimension that encodes the entire composition x, y.The discriminative reward model is trained using Adam optimizer (Kingma & Ba, 2014) and learning rate of 10^{-4} with a batch size of 64. The discriminative reward model is trained for 5 epochs which we find after which point the model starts to overfit. We also applied a dropout rate of 0.1 during training.

G. Test Set performance of Reward Models

We evaluate the contrastive and discriminative reward models on test set, and report performance in Table 4 for contrastive model, and Table 5 for discriminative model. For contrastive model, we evaluate it on retrieval metrics, namely recall (R@1, R@5, R@10) and mean average precision (mAP@10). For discriminative model, we use test set pair as positive label, and generate same number of negative samples by taking randomly pairing across test set. While performance of discriminative model remains similar across all scale, for contrastive model, the retrieval performance degrades at smaller scale. Given the focus of contrastive models on harmony evaluation, we posit that the observed decline in performance at smaller scales is attributed to the increased chance of harmonic coherence between unmatched samples in shorter segments.

We examine the bias of different reward models by plotting reward values against harmonic perturbations, where varying portions of chords in the test set are replaced with random alternatives. As shown in Figure 6, the contrastive model is more sensitive to harmonic perturbations.

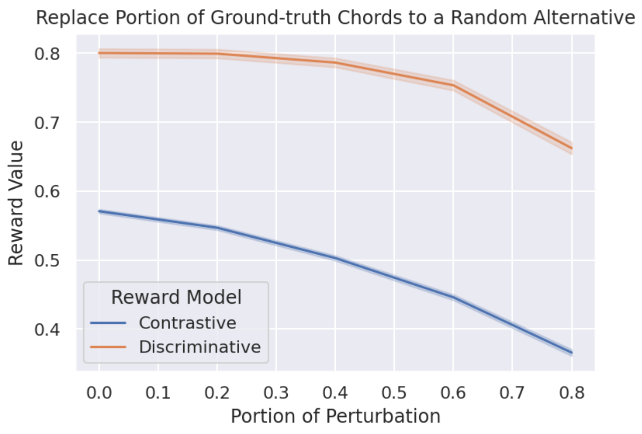


Figure 6. We take ground-truth samples from the test set, and replace a portion of chords with random chord names but keep the chord boundary. We then run the perturbed sample through contrastive and discriminative reward models. The points in line show the average reward value while shaded area shows the confidence interval (95%).The results show that the contrastive model is more attentive to harmonic perturbation or “mistakes”. Both reward models lower their scores as perturbation or “mistakes” increases.

H. Training Details of RL Finetuning

In RL training, we use a learning rate of 10^{-4} for both policy model and value model. In the initialization of RL finetuning, the policy model and the value model are both initialized from online MLE model. The coefficient β between reward maximization and KL loss in Equation 2 is fixed as 0.5 for all the experiments. We apply a coefficient of 50 to the reward produced by reward models. We apply a coefficient of 20 to the ending early penalty in all experiments used this penalty. In experiment using only knowledge distillation objective or using only one reward model, we apply a coefficient of 1 to both the repetition penalty and silence penalty. To train `ReaLchords`, we apply a coefficient of 2 to both the repetition penalty and silence penalty. To train `ReaLchords`-M, we apply a coefficient of 10 to both the repetition penalty and silence penalty.

We do not backpropagate gradients through the sampling process of the policy. All rewards, including reward models and penalties are summed and applied to the last token policy generated. For knowledge distillation objective, we only backpropagate gradients through the tokens policy generated, excluding the input tokens.

Table 4. The retrieval performance of contrastive reward model on test set.

Contrastive Reward Models	Note to Chord				Chord to Note			
	R@1	R@5	R@10	mAP@10	R@1	R@5	R@10	mAP@10
Full context (256 frames)	0.17	0.39	0.49	0.26	0.17	0.39	0.51	0.27
1/2 context (128 frames)	0.05	0.14	0.21	0.09	0.05	0.15	0.21	0.09
1/4 context (64 frames)	0.02	0.08	0.13	0.05	0.02	0.07	0.12	0.05
1/8 context (32 frames)	0.02	0.06	0.10	0.04	0.02	0.05	0.09	0.03
1/16 context (16 frames)	0.01	0.04	0.07	0.03	0.01	0.03	0.06	0.02

Table 5. The classification performance of discriminative reward model on test set.

Discriminative Reward Models	Precision	Recall	F1
Full context (256 frames)	0.69	0.91	0.79
1/2 context (128 frames)	0.69	0.92	0.79
1/4 context (64 frames)	0.71	0.84	0.77
1/8 context (32 frames)	0.69	0.88	0.77
1/16 context (16 frames)	0.68	0.79	0.73

For systems using only reward models (C, D, C+D in Table 1), in addition to regularization penalties, we also include the knowledge distillation loss between policy and online MLE model. Similar to Jaques et al. (2017), we found without such KL regularization, the training will be unstable and the models will generate invalid token sequences (e.g. a hold token before any onset token).

We also attempt to reproduce similar methods used in previous work RL-Duet (Jiang et al., 2020b) but received little success. We explored using offline MLE model as reward model, or using both offline MLE model and online MLE model as reward models, alongside with all regularization penalties we use. In all cases, the trained policy only generates repeated same chord, failing to generate adequate accompaniment.

I. Ensemble Same Type of Reward Models

We conduct experiments on RL finetuning with ensemble of same type of reward models, with results shown in Table 6. For both contrastive and discriminative model, ensemble with same kind helps improving better metric value.

Table 6. Effect of RL finetuning with our reward models and knowledge distillation on harmonic, synchronization and rhythmic diversity metrics. In row 2-7, we report confidence interval (95%) of metric values over 3 trainings, each with different random seeds. We additionally include results of RL finetuning with C+C and D+D (two contrastive or discriminative models trained with different initial seeds). From the metrics value, the ensemble of reward models helps achieving better metric performances.

Models	Harmony Note in Chord $\uparrow$, %	Synchronization Δ Chord-Note Onset Interval $\downarrow$, $\times 10^{-3}$	Rhythm Diversity Chord Length Entropy $\uparrow$
Online MLE	36.99	14.23	2.21
Knowledge Distillation (KD)	46.38 ± 0.45	14.39 ± 1.25	1.80 ± 0.04
Contrastive (C)	47.22 ± 1.22	16.61 ± 4.78	1.80 ± 0.29
Discriminative (D)	44.29 ± 0.99	12.34 ± 7.95	1.70 ± 0.06
C + D	46.12 ± 0.95	12.86 ± 6.17	1.56 ± 0.05
`ReaLchords`	48.17 ± 0.27	16.09 ± 3.63	1.35 ± 0.30
`ReaLchords`-M	54.29 ± 1.55	17.17 ± 4.86	1.66 ± 0.20
C + C	47.30 ± 1.63	10.13 ± 0.44	1.55 ± 0.05
D + D	46.15 ± 0.92	10.56 ± 1.33	1.63 ± 0.18
Offline MLE	63.73	9.85	1.90
Test set	70.94	0.00	2.19

J. Sample preparation for human evaluations

The listening test consisted of pairwise comparisons between different systems. The pairwise comparisons are prepared as following: We randomly sample 32 melodies from the test set, and have each system generate an accompaniment to the first eight bars of a song. For each melody, the accompaniments from each system is paired with all other systems, yielding $\binom{4}{2} = 6$ pairs. Going through all 32 melodies, we obtain $6 * 32 = 192$ total pairs. We recruited ten musicians, who each rated 15 to 20 pairs.

K. Quantitative Metrics Details

The entropy of chord length uses a base of e result in unit of nats. The EMD of chord-to-note onset interval reported in Table 1 is multiplied by 10^3 for better comparison. To determine the distribution of chord-note onset intervals, we categorized these intervals into bins defined by the number of frames, using bin boundaries set at $[0, 1, 2, ..., 16, 17, \infty]$. Similarly, for the distribution of chord lengths, we organized the data into bins based on the number of frames, with bin boundaries established at $[0, 1, 2, ..., 32, 33, \infty]$. Following this, we generated histograms for both the chord-note onset intervals and chord lengths. Finally, we computed the Earth Mover's Distance (EMD) between these histograms.

For note-in-chord ratio of a song, we average across the binary value at each frame representing whether the note is in a chord. For calculating whether the note is in a chord, we exclude the frames where either input melody or output chord is silence. Then, to report the note-in-chord ratio of a system, we average across the note-in-chord ratio of each song. To report the note-in-chord ratio at certain beat reported in §5.4 and §A, for each song we only consider the frames at that beat, and when averaging across songs, we exclude the song where the whole beat is silence.

L. Dataset and Data Representation Details

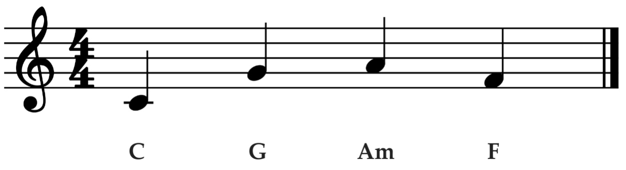


Figure 7. A visualization of example data samples.

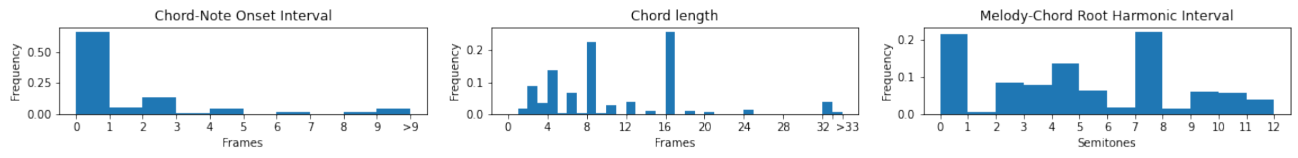
In `ReaLchords`, we represent each chord by a unique name, assigning each distinct chord name a unique index. Melody tokens x_t indicate both pitch and whether they mark the onset of a new note or the continuation of the previous token. Similarly, chord tokens y_t denote the chord symbol and its onset or continuation status. The Hooktheory dataset we use comprises 5041 distinct chords. For the output y, each frame y_t can be one of three possibilities: a chord selected from the 5041 available options, a chord hold (indicating the continuation of the previous chord) from the same set of options, or a silence, which signifies the generation of silence in the current frame. Similarly, there are 128 possible melody pitch values, ranging from 0 to 127 according to the MIDI pitch standard. For the input x, each frame x_t also falls into one of three categories: a note-on event chosen from 128 possible values, a note hold (indicating the continuation of the previous note) among the same 128 values, or a silence, indicating no input for the current frame.

As an example shown in Figure 7, the input melody x for that example at each frame would be: [C4_on, C4_hold, C4_hold, C4_hold, G4_on, G4_hold, G4_hold, G4_hold, A4_on, A4_hold, A4_hold, A4_hold, F4_on, F4_hold, F4_hold, F4_hold], and the output chords y for that example at each frame would be: [C_on, C_hold, C_hold, C_hold, G_on, G_hold, G_hold, G_hold, Am_on, Am_hold, Am_hold, Am_hold, F_on, F_hold, F_hold, F_hold].

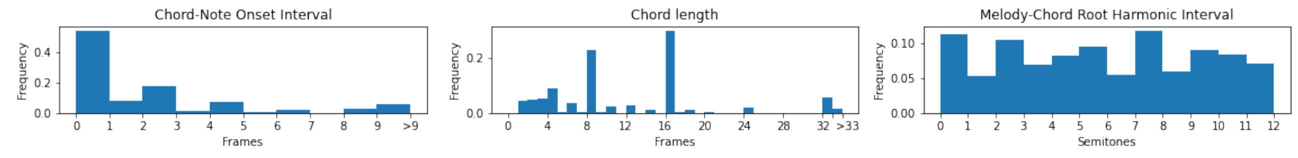
The maximum length of the examples used for training is set to 256 frames, equivalent to 64 beats, with any samples exceeding this duration being randomly cropped during training.

M. Histogram Statistics of Compared Systems

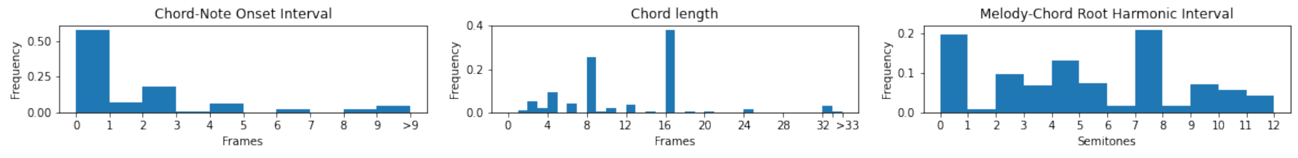
In Figure 8 and Figure 9, we present the histogram of chord-to-note onset interval, chord length and note-to-chord harmonic interval for one of each system compared in Table 1. Those histograms are used to calculate statistics report in Table 1.



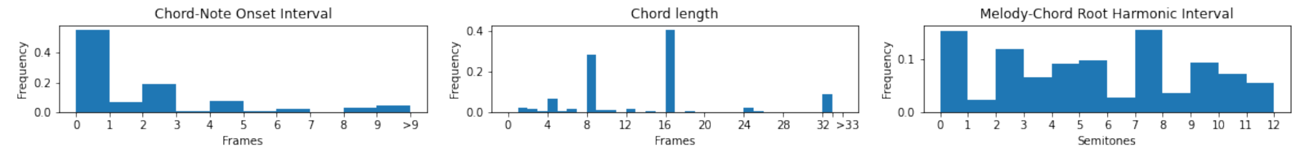
(a) Test Set



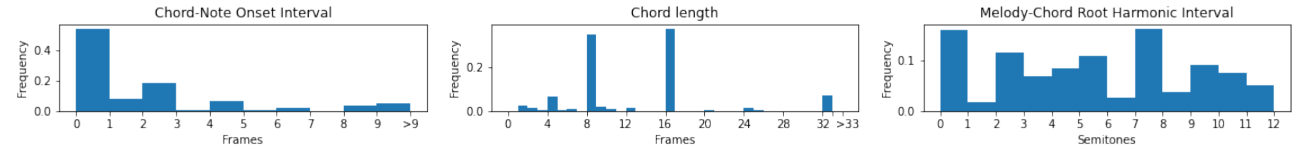
(b) Online MLE



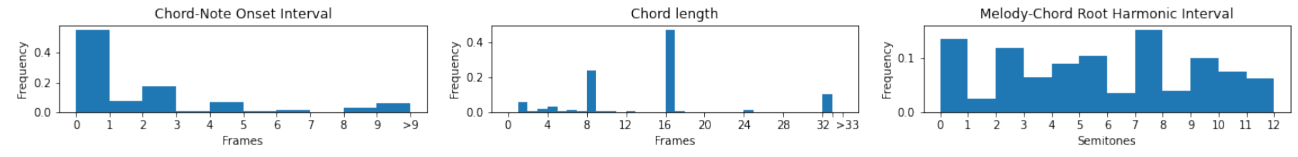
(c) Offline MLE



(d) Knowledge Distillation (KD)

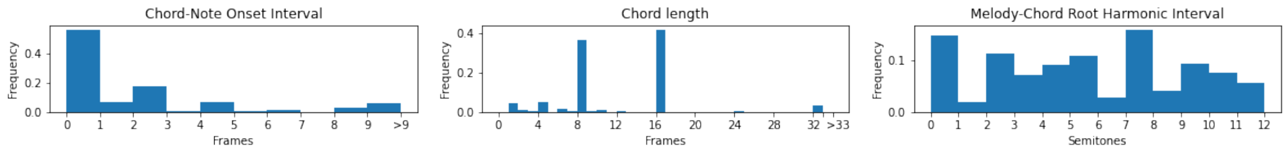


(e) Contrastive Reward Model (C)

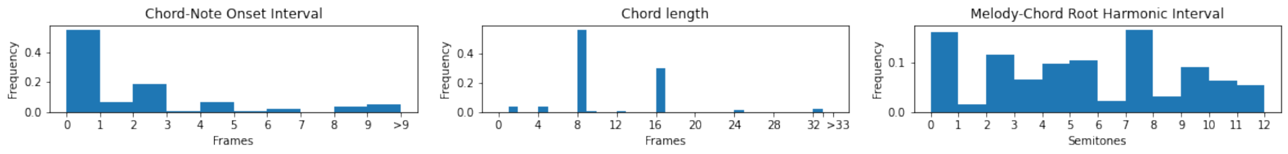


(f) Discriminative Reward Model (D)

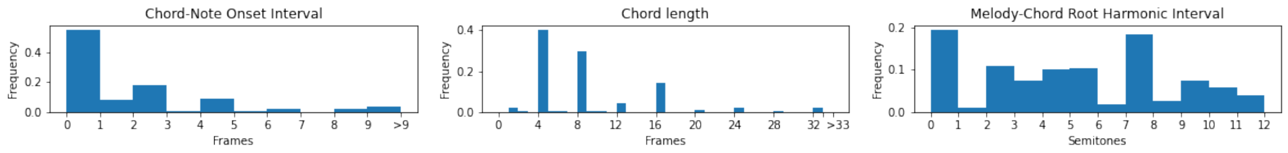
Figure 8. The histogram of chord onset interval, chord length and harmonic interval for each system compared in §5.3.



(a) Contrastive + Discriminative Reward Model (C+D)



(b) `ReaLchords`



(c) `ReaLchords`-M

Figure 9. The histogram of chord onset interval, chord length and harmonic interval for each system compared in §5.3.