3. Evaluation

Context:

The goal for our task is to generate melodies in the style of Nintendo video game soundtracks, given an input melody.

While the evaluation of music generating models can not be 100% quantitative, there are certain properties we can look at to see how “good” our output is. Let’s walk through these:

1. **Perplexity/Cross Entropy**
   a. **Definition:** CE measures how well the model can predict a sequence of notes. Lower CE means the model assigns higher likelihood to the ground-truth. Perplexity is just the exponential of CE.
   b. **Impact:** Lower perplexity means that the model assigns higher probability to plausible note choices. This means that lower the perplexity the more the generated melody will resemble that of NES soundtracks.
2. **Chord-Tone Accuracy**
   a. **Definition:** The percentage of generated notes whose pitch class matches any pitch class of the conditioning chord at the same time-step.
   b. **Impact:** A higher value for this means that the melody stays relatively close to the intended harmony.
3. **Global Tonal Similarity**
   a. **Definition:** We are looking at the pitch content, which notes are used, how often, and how often they relate to each other, also known as pitch distance. This is done with cosine similarity.
   b. **Impact:** We can take a look at the key/scale distribution and see if it matches the dataset. If it does, that means the melodies stay in the right tonality
4. **Interval Entropy**
   a. **Definition:** This is a measure of melodic variety, meaning that the generated output is neither too repetitive or too random.

    b. **Impact:** If Interval entropy is 0 bits, we get a monotone sound. If its 3.6 bits(log(12)), then its max variety, usually too random. This means we want the interval entropy to be somewhere in the middle.

5. **The Human Experience**
    a. **Definition:** This is just how it sounds to our ears.
    b. **Impact:** While this is the least quantitative method of evaluation, it is probably the most important. We are able to pick up many properties in a musical piece that are very hard to quantify and formally record.

Discussion:

Baselines Models to Compare

Repeat Input: This naive approach just trains a model to play the chord root. This is a simple case where the model just learns model beats but does not generate anything new. Basically just memorizing.

Bigram: This is a next-note bigram baseline scenario where the next note is sampled unconditionally given a note. This is a classic example in most music-gen papers. This ignores chords and is short–range only.

Chord-aware Bigram: This model creates a separate bigram for each chord symbol. This adds harmonic context with minimal complexity. However there is no long-range memory for this model.

As we can see in the data below, the LSTM performs much better than the baseline cases. This is because the LSTM's hidden state carries a summary of all previous notes *and* the current chord embedding, so the model can learn longer-range patterns. A bigram, by definition, forgets everything beyond the last note and therefore can't model those dependencies.

System	Perplexity	Chord-Tone %	Pitch Distance	Interval Entropy
Repeat Input	NA	100.0	0	0 bits
Unconditional	7.1	11.0	1.0	NA

Bigram				
Chords Bigram	**7.7**	**11.1**	**0.36**	**2.6 bits**
Our Bi-LSTM	**3.3**	**14.1**	**.071**	**2.4 bits**

Perplexity(lower is better):
If we look at the results, we can see that the perplexity(model confidence) is less than 2x of the bigram models. This means that our Bi-LSTM guesses the next note way better than a simple bigram probability table.

Pitch-class Histogram Distance(lower is better):
For each melody we build a 12-bin histogram that counts how often each pitch-class (C…B, ignoring octaves) occurs, then average those histograms over the whole test set. We can see in these results that our Bi-LSTM performs the best as it almost nears 0.
0 = perfect match - The model stays in the same keys/scales as the dataset
2 = completely different - all pitches are different

Interval Entropy(variety):
Anything near 2.5–3 bits is usually considered a nice mix of small steps and bigger leaps that allows for variety. The chord-aware bigram and the Bi-LSTM both land at around 2.5 bits, while the repeat-root baseline (0 bits) is flat. The unconditional bigram does not take into account chords and therefore can not be determined.

Chord-Tone %(higher is better)
All the models, including our LSTM, are stuck near 11 – 15%. So even though the LSTM is great at predicting *something* reasonable, it still ignores the chords most of the time. Cross-entropy is not a predictor of a harmonic melody.

Code:

Evaluation Protocol:

1. For each test song:
    a. Feed chord tokens through model -> logits
    b. Cross-entropy vs ground-truth notes -> Per-song perplexity
    c. Greedy arg-max -> generated melody
    d. Compare generated notes to chord sets -> chord-tone accuracy

2. After 300 songs:
    a. Get the mean of each metric
    b. Compute interval entropy and pitch hist distance
3. Visualize
    a. Histogram of per-song perplexity
    b. Bar chart of mean metrics

Many of the evaluation functions are made with the help of two crucial python libraries **muspy and music21**.

Helper Functions:
chord_tone_match(chords, notes, id2pc=None)
- Fraction of notes whose pitch-class matches the current chord.

seq_to_music(seq)
- Convert a list[int] note-tokens to a MusPy Music object.

pitch_hist_distance(generated, reference)
- L1 distance between the average pitch-class histograms of two lists of sequences

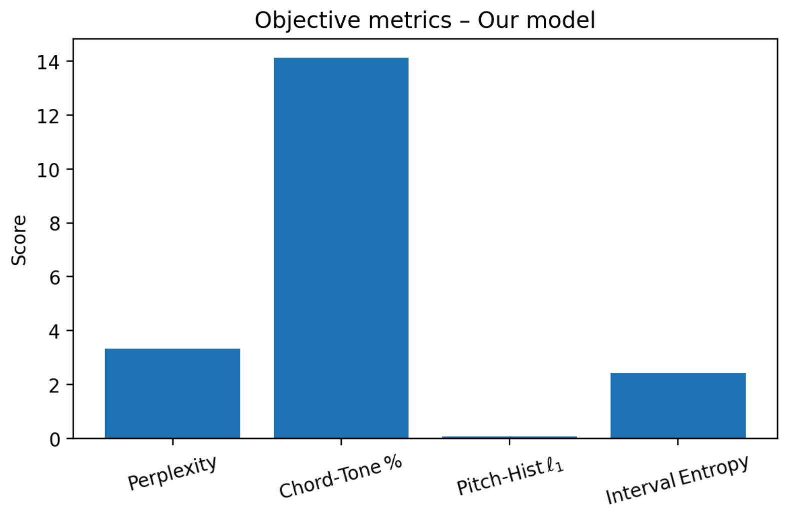
We first created the unconditional bigram and chord-aware bigram model. We made sure this can also go through our evaluation pipeline.

Bigrams Output:
```
Unconditional bigram -
perplexity_mean:  7.729547514594387
chord_tone_acc:  11.746323529411764
pitch_hist_dist:  1.0
pitch_interval_entropy: nan
-------------------------------------------------------
```

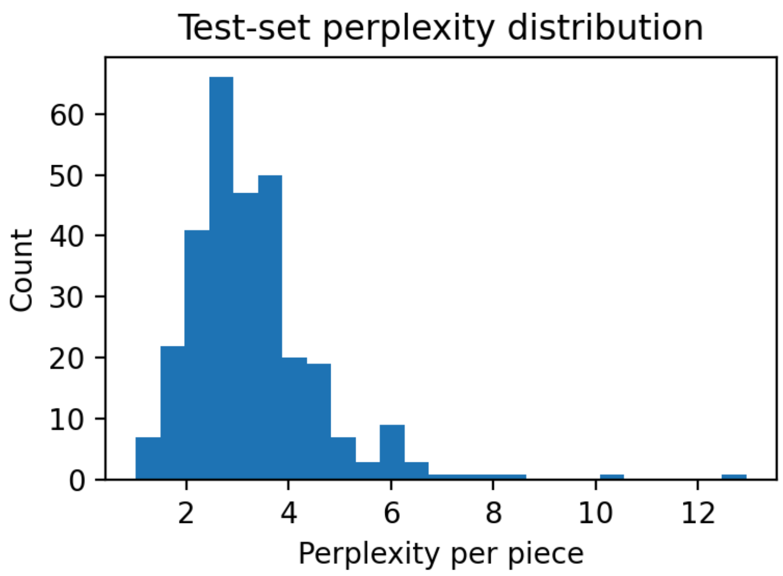
Chord-aware bigram -
perplexity_mean: 7.132908656239655
chord_tone_acc: 11.580882352941178
pitch_hist_dist: 0.36294568436833324
pitch_interval_entropy: 2.636203893101653

We then ran our LSTM model through the evaluation pipeline

LSTM Output:
"perplexity_mean": 3.3254613267455237,\n'
"chord_tone_acc": 0.14130208333333333,\n'
"pitch_hist_dist": 0.07090021652442936,\n'
"pitch_interval_entropy": 2.4272686870566353\n'



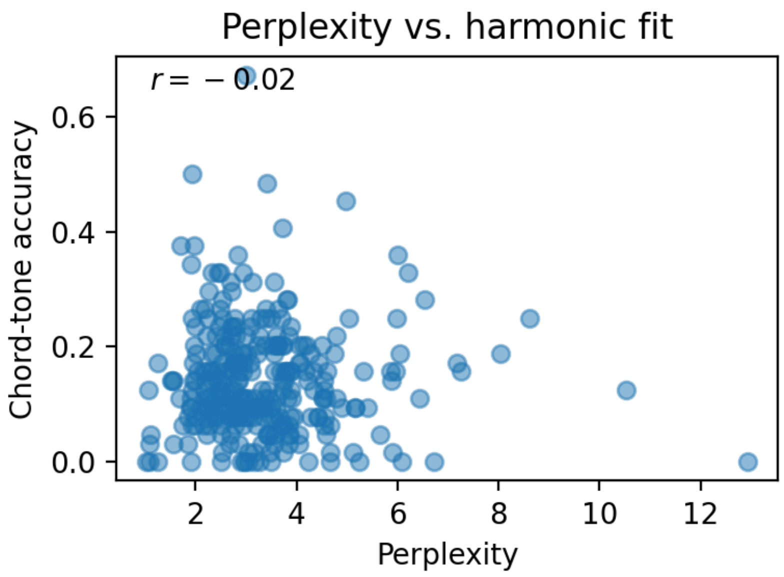
Simple bar graph plotting the different properties evaluated.



X-axis: Perplexity per piece.
For each of the 300 test melodies in the evaluation protocol we plotted exp(cross-entropy), lower means the model assigned higher probability to the ground-truth notes.

Y-axis: Count.
How many test pieces fell into each perplexity bin.

So roughly 80 % of the pieces have perplexity < 4, but a handful are much harder for the model.



This figure reveals virtually no correlation between perplexity and chord-tone accuracy ($r = 0.02$). This confirms that the cross-entropy objective alone is insufficient to guarantee harmonic correctness: the model often predicts statistically plausible, yet harmonically inappropriate, notes. We therefore propose adding an auxiliary cord-alignment loss in future work.

Discussion:

1. The Nin Video-Game MIDIs dataset we got is only recently available on Kaggle and, to our knowledge, has not yet appeared in a peer-reviewed music-generation study. It differs from most symbolic game-music datasets in two ways: every file is a solo-piano reduction, so the texture is cleaner, and the dataset spans several console generations and franchises, giving broader coverage than 8-bit-only sets.

   Previous work on video-game music has mostly used the NES-MDB family of datasets, including the original multi-voice NES-MDB and its video-aligned extension NES-VMDB, to study music generation, and performance rendering.
2. Similar Works and Comparisons

Year	Paper	Implementation
2002	Eck & Schmidhuber – "Finding temporal structure in music"	Most relied on listening examples instead of evaluation metrics, Early uni-directional LSTM improvised a blues solo conditioned on a fixed 12-bar chord loop. Proof-of-concept that LSTMs can map harmony→melody.
2019	ChordAL (Tan et al., ICCC demo)	Perplexity(NLL) -> cross entropy loss is around 2.9 per token and if you exponentiate it you get around 18, but this is not validated. Two parallel LSTMs (one for pitch, one for duration) receive a one-hot chord label at every time-step; improves consonance vs. un-conditioned baselines on folk tunes.

2021	CM-HRNN (Ren et al.) – “Hierarchical RNN for Conditional Melody Generation”	Perplexity(NLL) = 2.93 The model uses **two LSTM layers** that work at different time-scales: **Measure planner (top layer).** Every bar, this layer decides the overall shape of the melody. **Beat-by-beat writer (lower layer).** It turns those bar-level decisions into the actual notes for each beat. The chord you’re on is fed into **both** layers so each note stays harmonically appropriate.