SYMBOLIC MUSIC GENERATION WITH CSCE585-MIDI

(I couldn’t think of a better name)

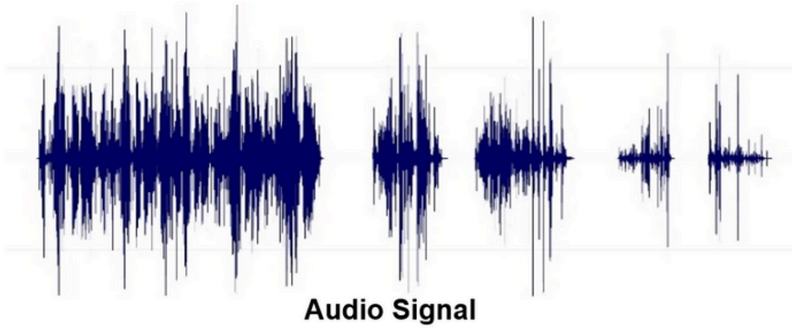
Cade Stocker

INTRODUCTION

Most generative models make music with waveform audio files. MIDI files contain symbolic representations of musical events like pitch, duration, and velocity. MIDI files can be easily edited by users after they are generated, making them more useful and convenient for music generation.

csce585-midi is a flexible experimentation platform, allowing the user to:

- Preprocess any desired MIDI dataset
- Mix and match trained models to use when generating
- Configure and customize training settings and hyperparameters
- Select preferred sampling strategies when generating music
- Automatically evaluate and log all generated music
- Use any desired MIDI file as a seed for generation



Audio Signal

MIDI Information (messages)

MOTIVATION

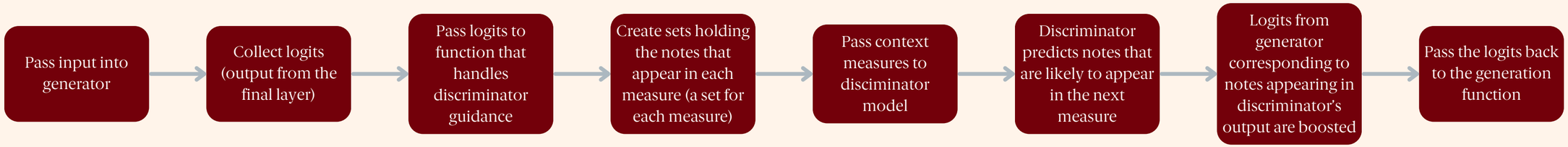
I started by attempting to replicate the architecture described in “Musenet : Music Generation using Abstractive and Generative Methods”.

I trained models on the Nottingham dataset, which is comprised of around 1,200 Irish and British instrumental folk songs. I wanted to use this dataset specifically because most of the songs are popular in the genre of Bluegrass.

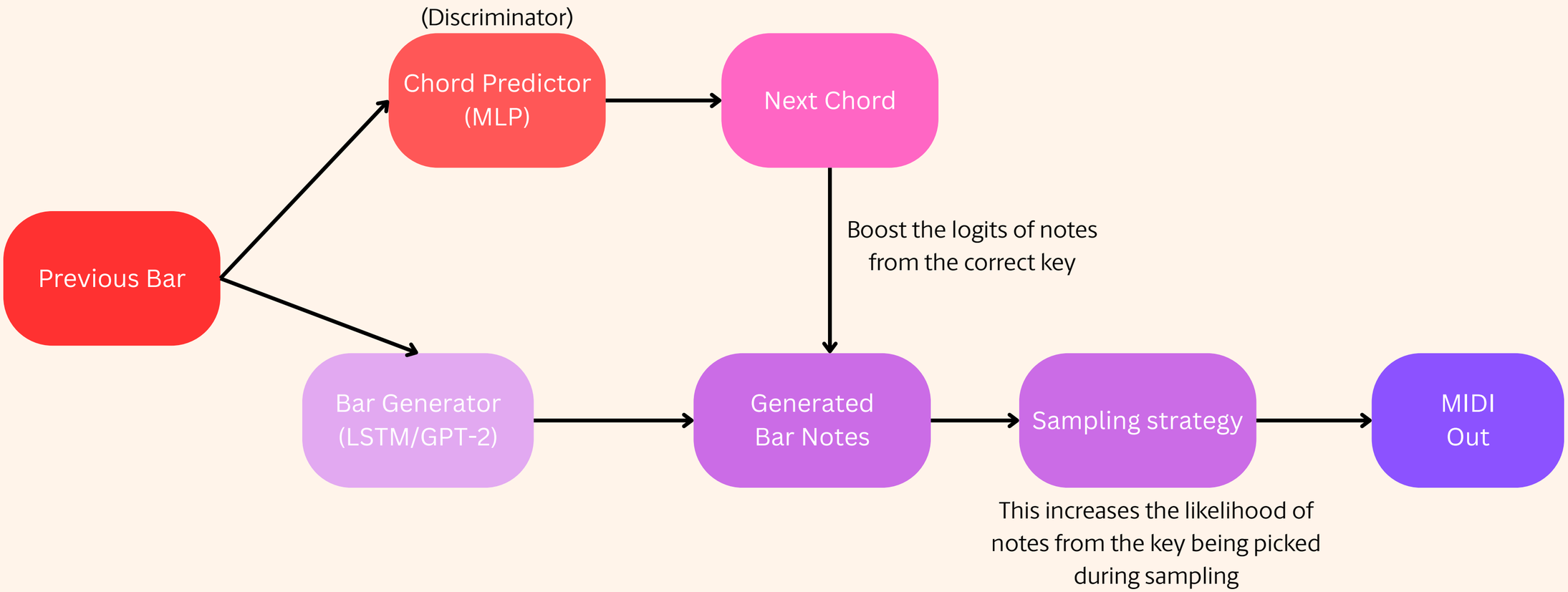
MUSENET

The architecture from Musenet revolves around two models working together to create music:

- A generator model that predicts the notes that are likely to appear next in a sequence
  - In the paper, they experiment with both LSTM (Long short term memory) and GPT-2 transformer models for the generator.
- A discriminator model trained on notes contained in measures. It generates sets of candidate notes, which are used to help guide the generator.
  - Musenet used a MLP (Multi-Layer Perceptron) model for the discriminator.



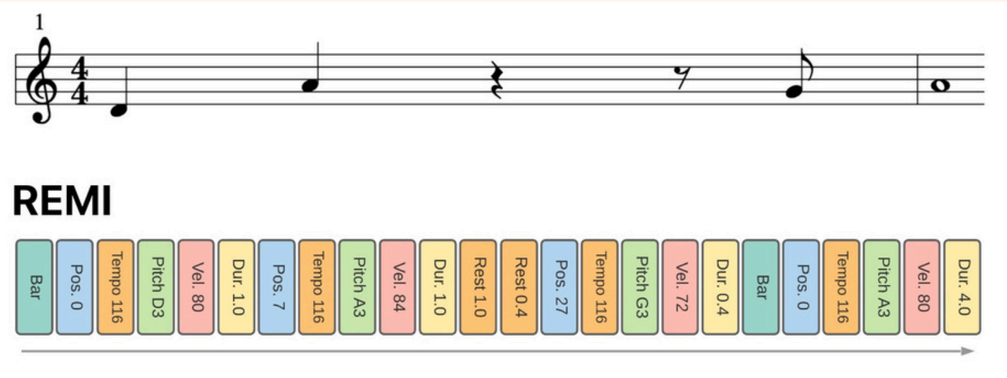
MUSENET


(Discriminator)
Chord Predictor (MLP)
Next Chord
Previous Bar
Boost the logits of notes from the correct key
Bar Generator (LSTM/GPT-2)
Generated Bar Notes
Sampling strategy
MIDI Out
This increases the likelihood of notes from the key being picked during sampling

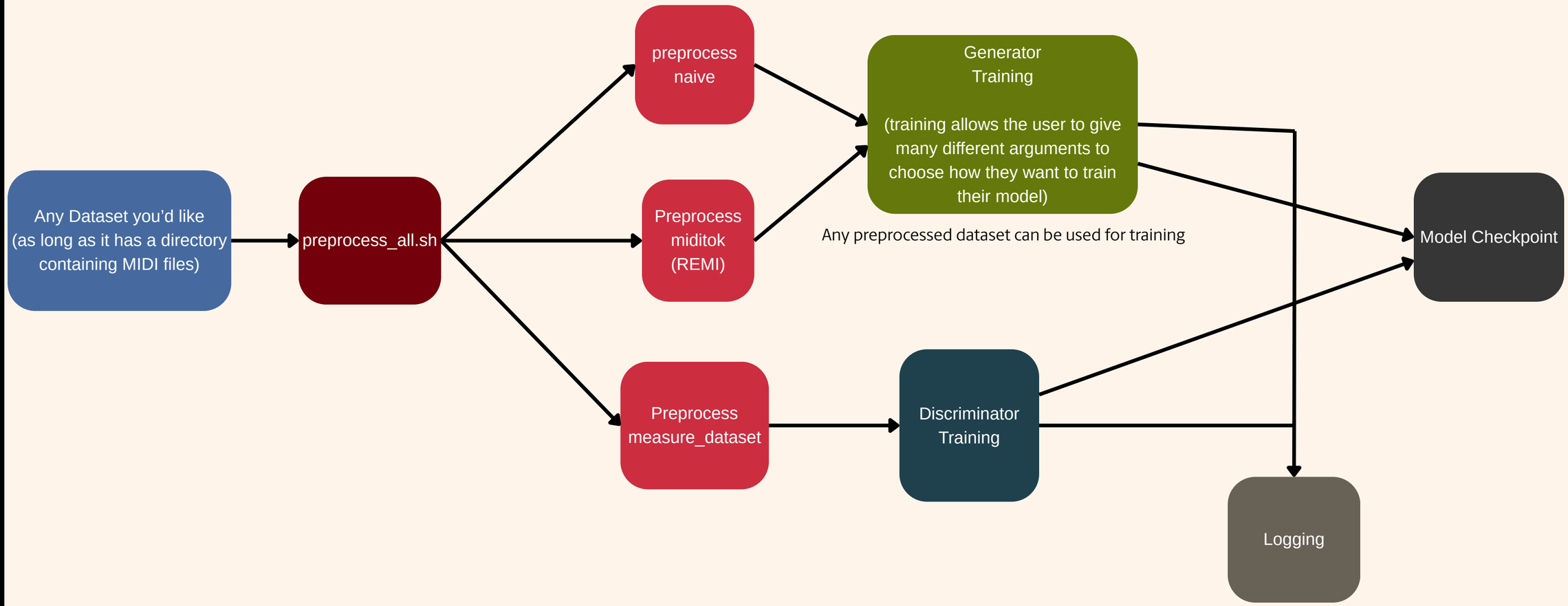
PREPROCESSING

Two types of tokens used:

- Naïve tokenization, where each note gets represented as a single token. This allows small models to learn easier, but this limits generation to constant 8th note rhythms.
  - Sequences look like [A3, C3, E4, D3, B4, A3, C3, ...]
- Miditok REMI (Revamped MIDI) tokenization converts each note into several tokens describing pitch, velocity, duration, and other musical features.



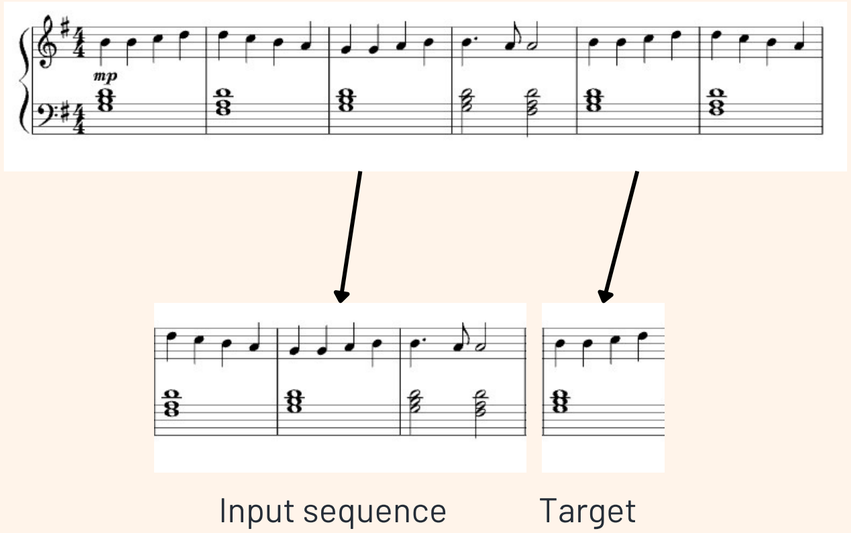
TRAINING


Any Dataset you'd like
(as long as it has a directory containing MIDI files)
preprocess_all.sh
preprocess
naive
Preprocess
miditok
(REMI)
Preprocess
measure_dataset
Generator
Training
(training allows the user to give many different arguments to choose how they want to train their model)
Any preprocessed dataset can be used for training
Discriminator
Training
Model Checkpoint
Logging

TRAINING

Users can train their desired generator and discriminator models, with commands like:

```
python  training/train_generator.py  \
--dataset data/nottingham_naive \
--model_type lstm \
--epochs  20  \
--batch_size 128 \
--lr  0.001  \
--hidden_size 512
```



EXPERIMENTS:

The following were tested to see their impact on music generation:

1. **Sampling strategy**
2. **Discriminator guidance**
3. **Tokenization type**
4. **Architecture of Generator**
5. **Architecture of Discriminator**

EXPERIMENTS:

These metrics were used to evaluate the generated MIDI files:

- Number of notes
- Duration in seconds
- Note density (notes per second)
- Pitch range (difference between lowest and highest pitch)
- Polyphony (proportion of time where more than 1 note is sounding)
- Scale consistency (fraction of notes which are contained in the major or natural minor scales of another note in the generated sequence)
- Pitch entropy (variation across pitch values)
- Pitch class entropy (variation across pitch classes)
    - Pitch class is the name of the given note regardless of its octave (C3 and C4 are the same pitch class of C)

EXPERIMENTS:

The following models were trained and used for the experiments:

Model Name	Model Architecture	Dataset	Epochs	Learning Rate	Embed Size	Hidden Size
Baseline Naïve	LSTM	nottingham_naive	10	0.001	128	256
Baseline Miditok	LSTM	nottingham_miditok	10	0.001	128	256
Augmented Miditok	LSTM	nottingham_miditok_augmented	10	0.001	128	256

Note that “augmented” means that each song in the dataset was transposed to 7 total keys during preprocessing, meaning that there were far more sequences to train off of.

EXPERIMENTS:

These metrics were used to evaluate the generated MIDI files:

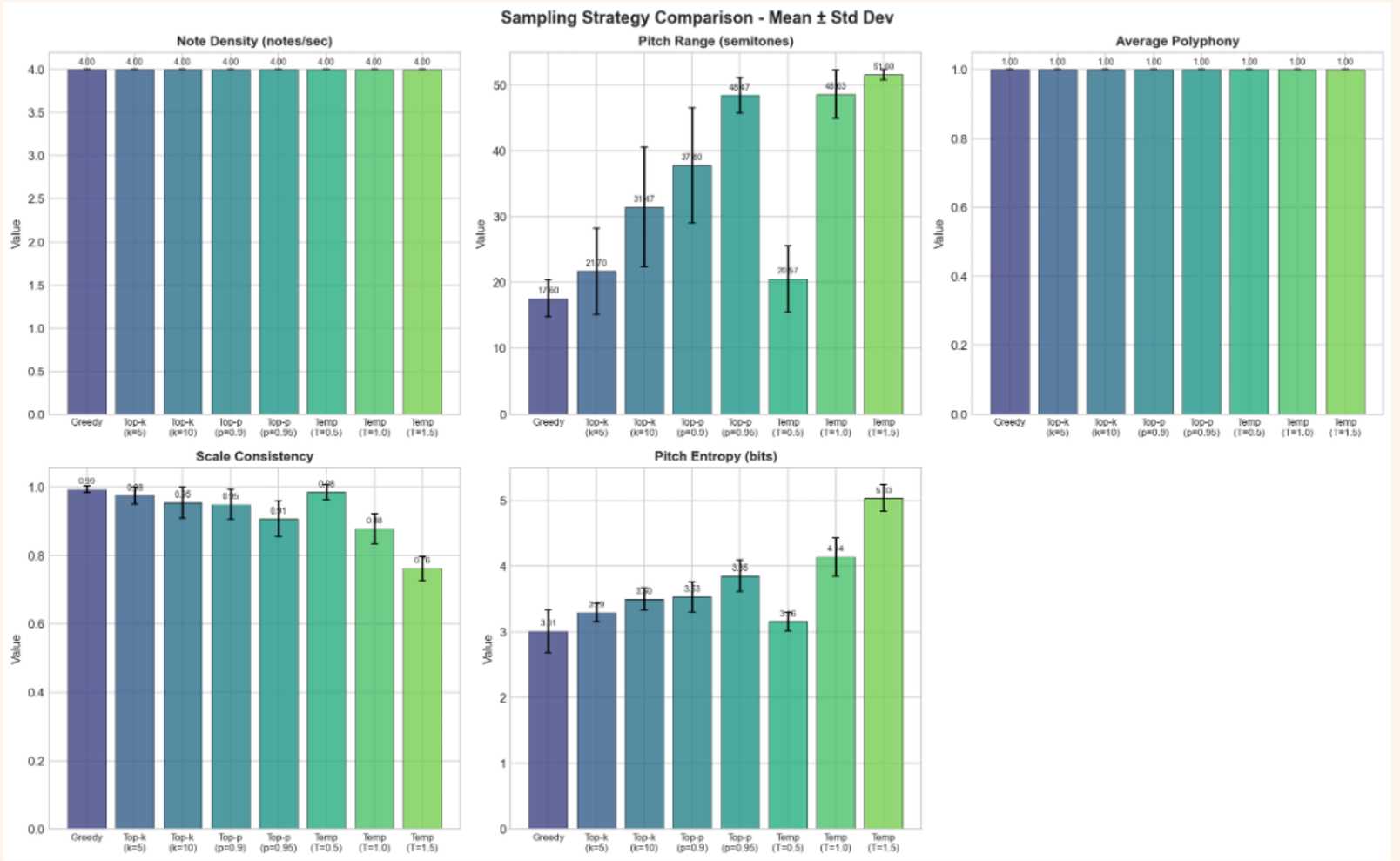
- Number of notes
- Duration in seconds
- Note density (notes per second)
- Pitch range (difference between lowest and highest pitch)
- Polyphony (proportion of time where more than 1 note is sounding)
- Scale consistency (fraction of notes which are contained in the major or natural minor scales of another note in the generated sequence)
- Pitch entropy (variation across pitch values)
- Pitch class entropy (variation across pitch classes)
    - Pitch class is the name of the given note regardless of its octave (C3 and C4 are the same pitch class of C)

SAMPLING STRATEGY

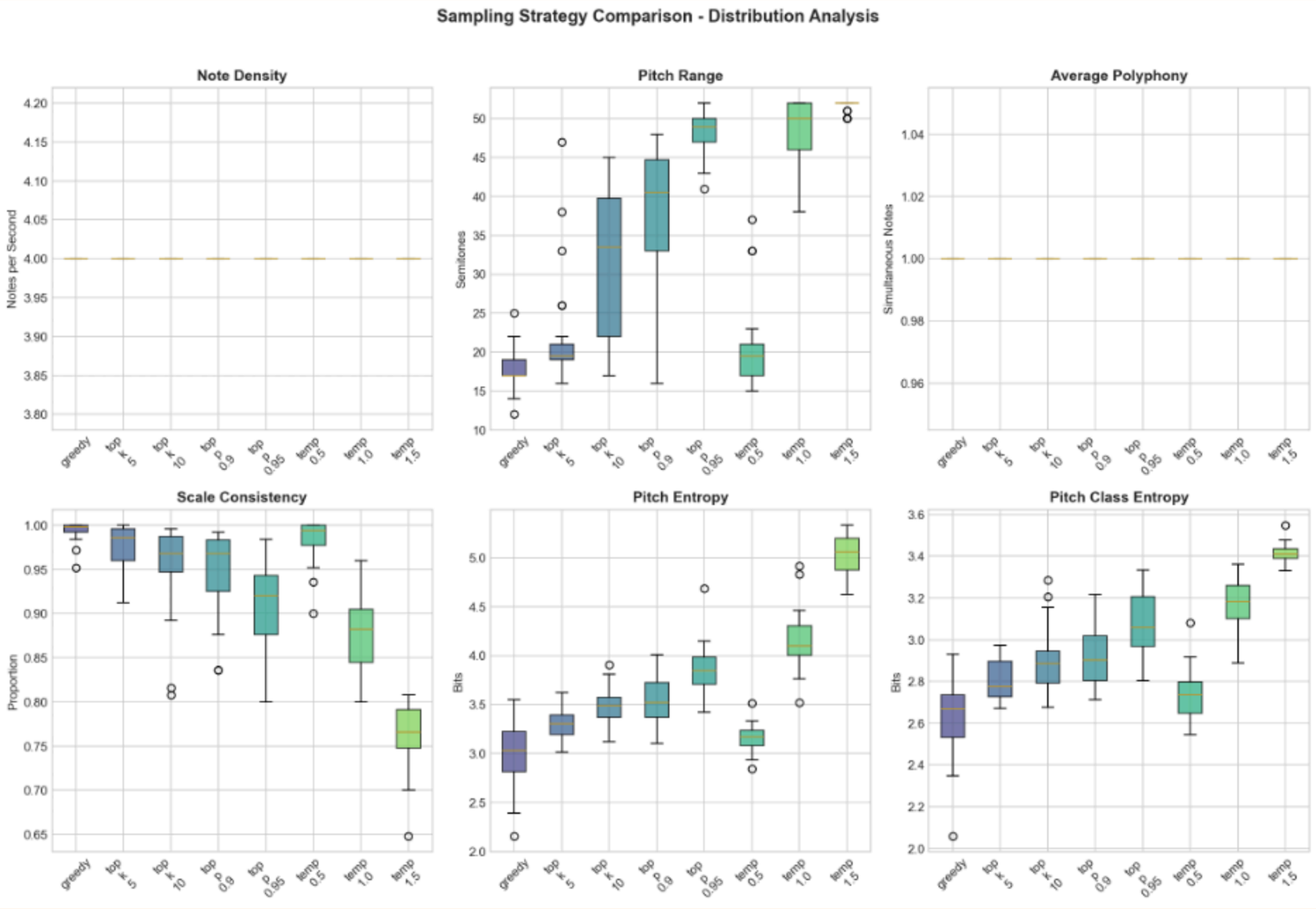
Using the baseline naive LSTM model, 30 samples were generated for each of the following sampling methods:

- Temperature: 0.5, 1.0, 1.5
- Top-k: k = 5, 10
- Top-p: p = 0.9, 0.95

SAMPLING STRATEGY

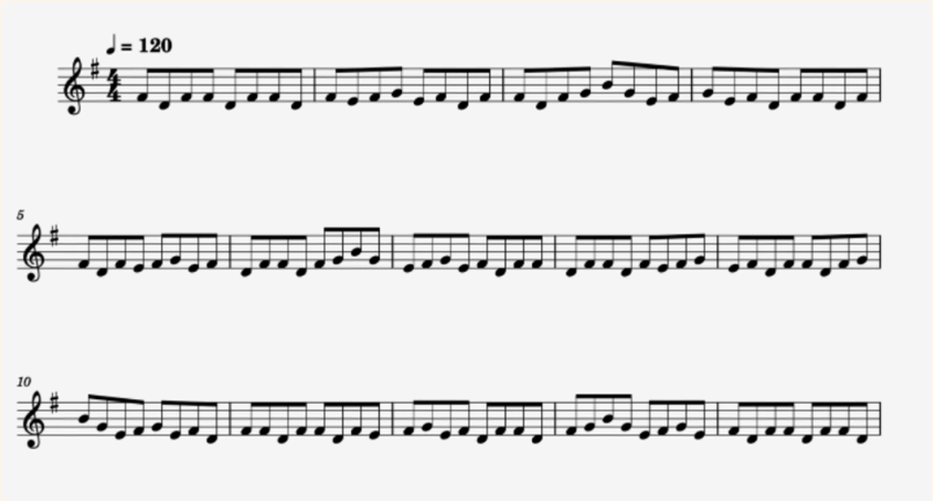


SAMPLING STRATEGY

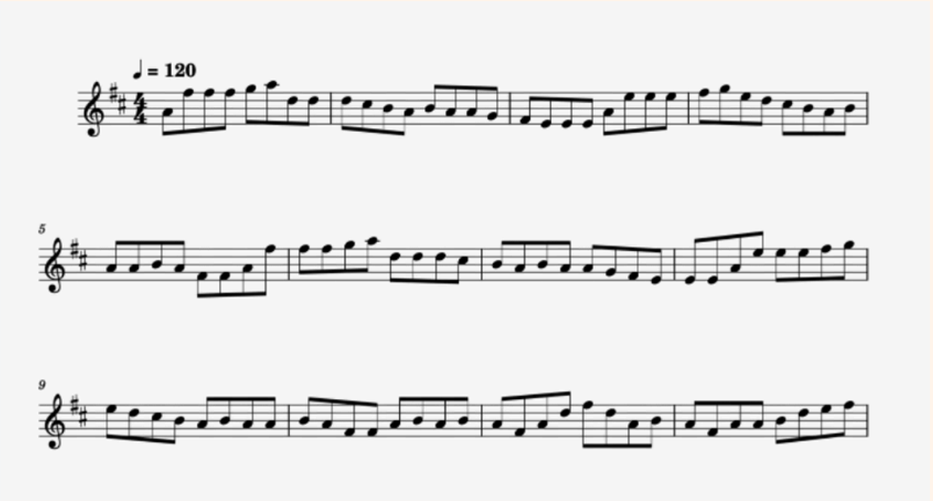


SAMPLING STRATEGY

MIDI generated with temperature = 1.0



MIDI generated with temperature = 2.0



SAMPLING STRATEGY

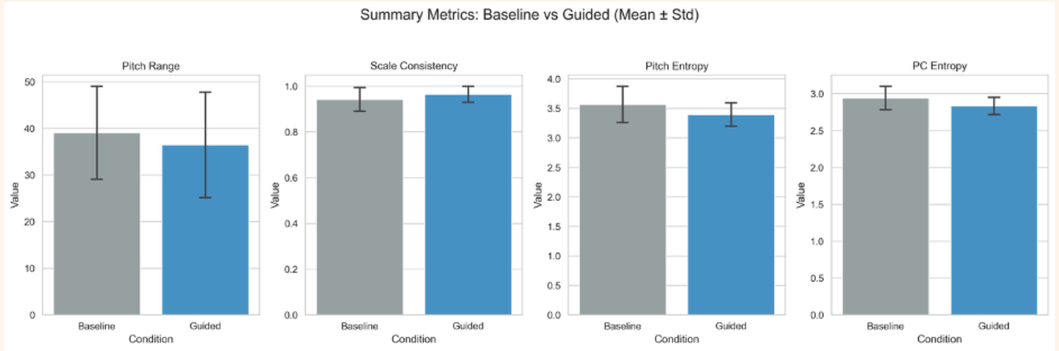
Findings:

- **Higher temperature, k, and p values result in higher pitch variety. (5.3, 5.4)**
- **Low temperature and greedy sampling frequently collapsed into the same sequence of notes getting repeated constantly.**
- **Because naïve models only output constant 8^{th} notes, density and polyphony are unchanged across all sampling settings.**

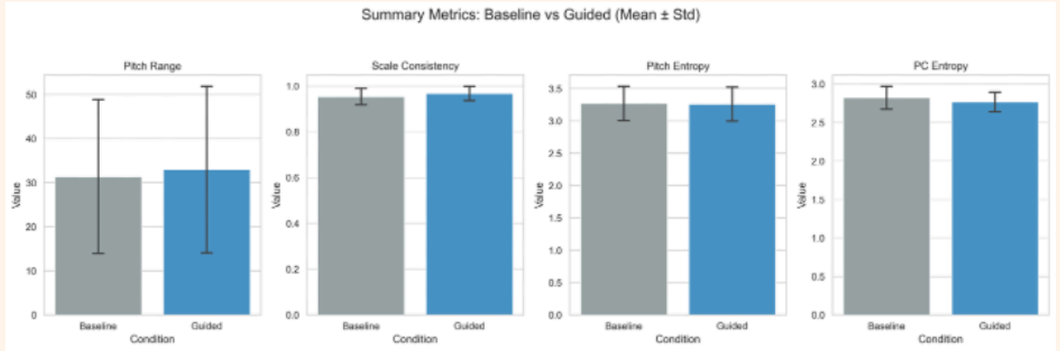
DISCRIMINATOR GUIDANCE

To analyze how discriminator guidance alters musical characteristics, 50 samples were generated with and without a trained MLP discriminator (the same model architecture used in Musenet). Experiments were repeated for the naïve and miditok baseline generators.

Naive

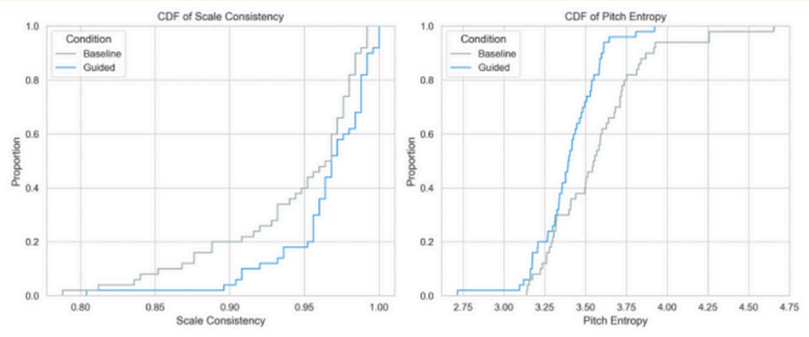


Miditok

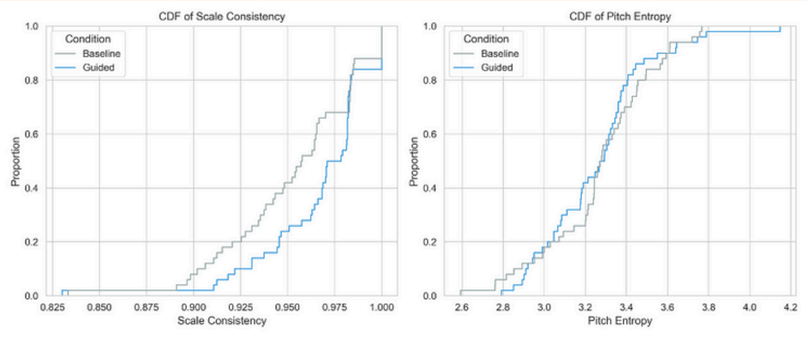


DISCRIMINATOR GUIDANCE

Naive



Miditok



Key results:

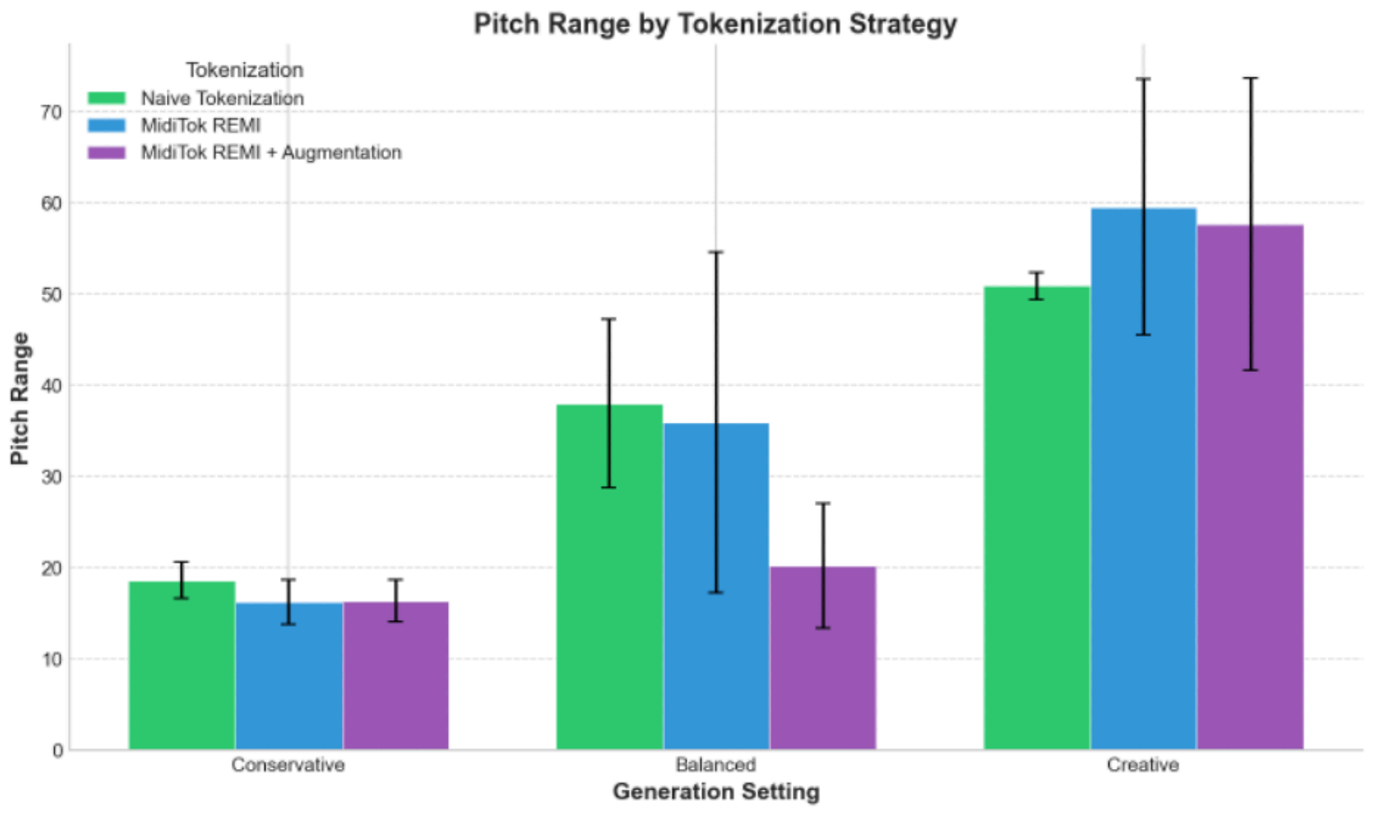
- **Scale consistency increased for both generators when given discriminator guidance.**
- **Pitch class entropy increased, which indicates more unique notes being used.**
- **Miditok models surprisingly increased in pitch range when given guidance. This is likely because guidance prevented it from collapsing into repeated patterns.**
- **Guided samples sounded more correct but less expressive. They avoided any kind of interesting chromatic note choices.**
- **The CDF plots confirm that having discriminator guidance causes higher scale consistency and also reduces pitch entropy.**

TOKEN TYPE

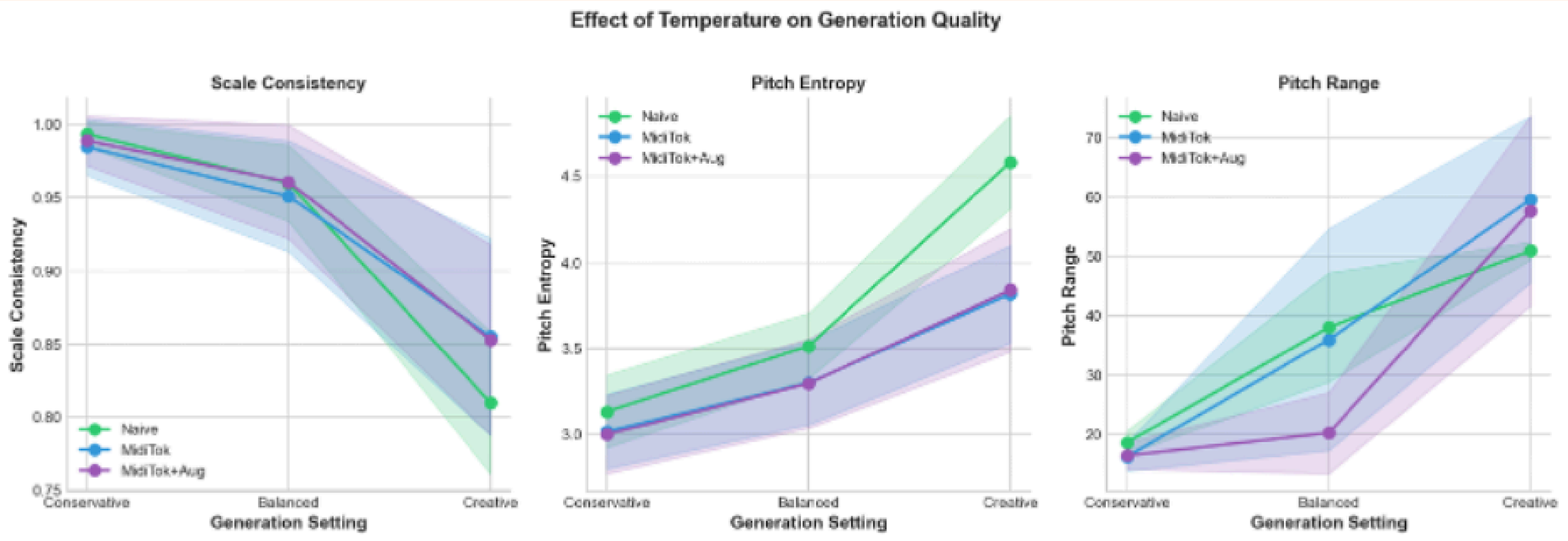
This experiment compared naïve, miditok, and augmented miditok LSTM models. 90 MIDI files were generated by each model and are evaluated. I expected naïve models to perform better than the standard miditok model, but I assumed that by augmenting the miditok model's dataset, it could generalize better and become comparable to the naïve model in performance.

Each model generated 30 MIDI files with 3 types of generation arguments:
·Conservative: p = 0.85, Temp = 0.8
·Balanced: p = 0.9, Temp = 1.2
·Creative: p = 0.95, Temp = 1.2

TOKEN TYPE


Pitch Range by Tokenization Strategy
Tokenization
Naive Tokenization
MidiTok REMI
MidiTok REMI + Augmentation
Pitch Range
0
10
20
30
40
50
60
70
Conservative
Balanced
Creative
Generation Setting

TOKEN TYPE


Effect of Temperature on Generation Quality
Scale Consistency
Scale Consistency
1.00
0.95
0.90
0.85
0.80
0.75
Naive
MidiTok
MidiTok+Aug
Conservative
Balanced
Creative
Generation Setting
Pitch Entropy
Pitch Entropy
4.5
4.0
3.5
3.0
Naive
MidiTok
MidiTok+Aug
Conservative
Balanced
Creative
Generation Setting
Pitch Range
Pitch Range
70
60
50
40
30
20
Naive
MidiTok
MidiTok+Aug
Conservative
Balanced
Creative
Generation Setting

TOKEN TYPE

Findings:

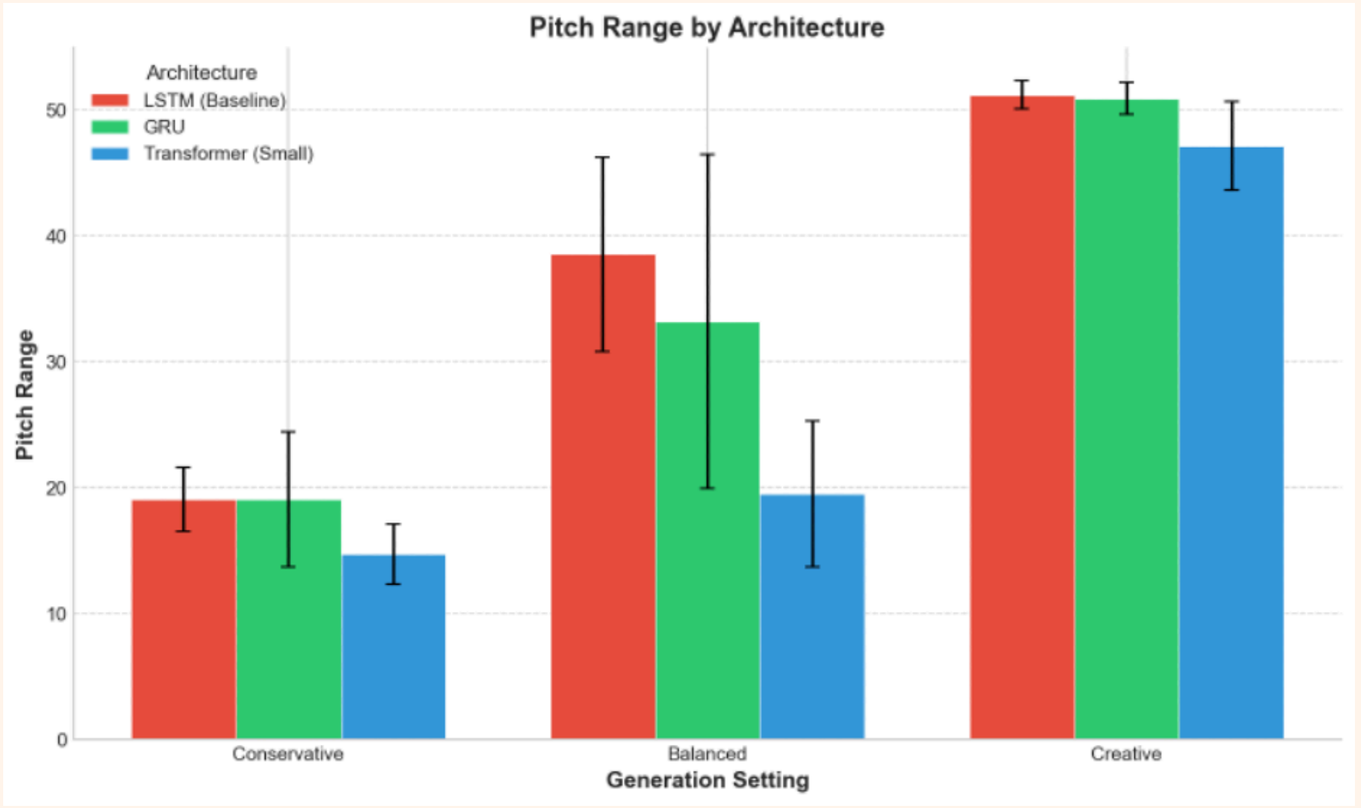
- Naïve models had the highest pitch range in conservative and balanced settings.
- With creative sampling, both miditok models surpassed the naïve model in pitch range.
- The augmented miditok model consistently outperformed the standard miditok model across most metrics, showing that a larger dataset is necessary for more complicated tokenization methods.

GENERATOR ARCHITECTURE

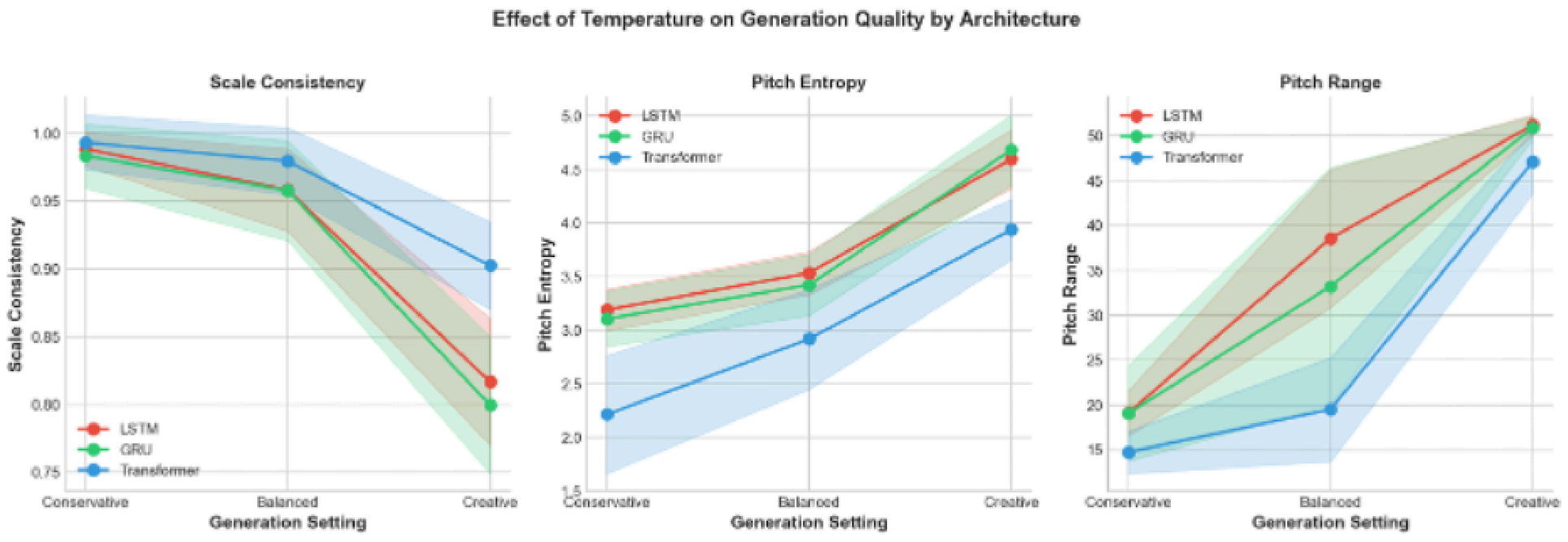
The baseline LSTM model got compared to:

- A GRU model
- A small Transformer model

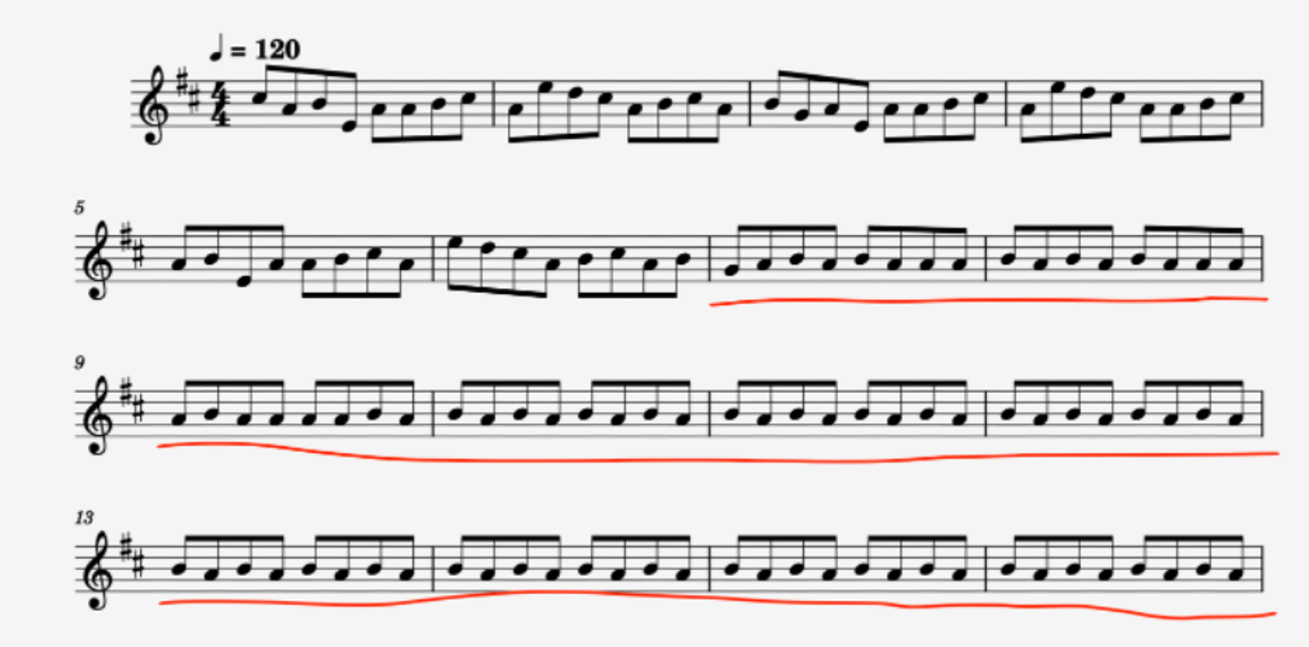
All models were trained under the same hyperparameters and used naive tokens.



GENERATOR ARCHITECTURE



GENERATOR ARCHITECTURE



Transformer model output repetitive sequences even with top-p sampling.

GENERATOR ARCHITECTURE

Findings:

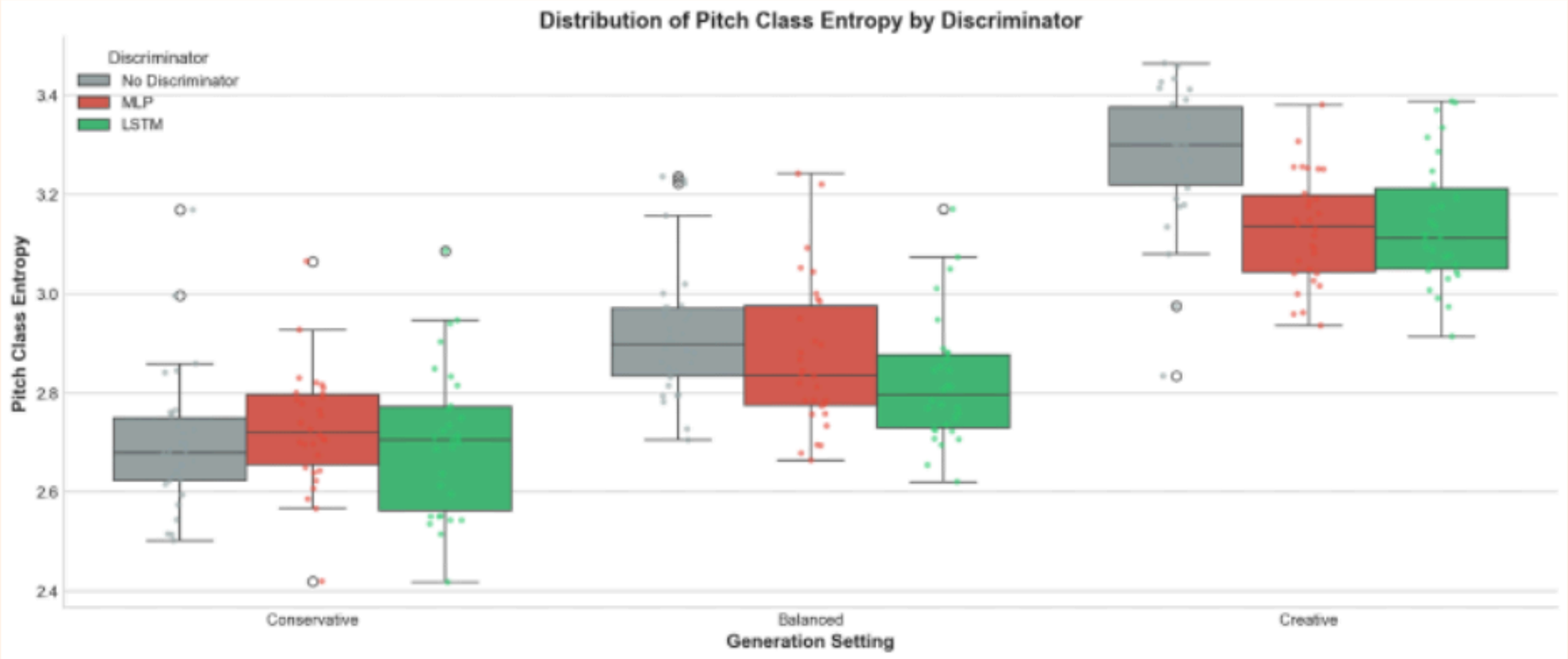
- **GRU performance was close to the LSTM but slightly worse.**
- **Transformer performance was consistently the worst. Even with top-p sampling, it produced repetitive sequences with low pitch variety.**

This highlights that Transformer models require much larger datasets to produce useful and coherent music.

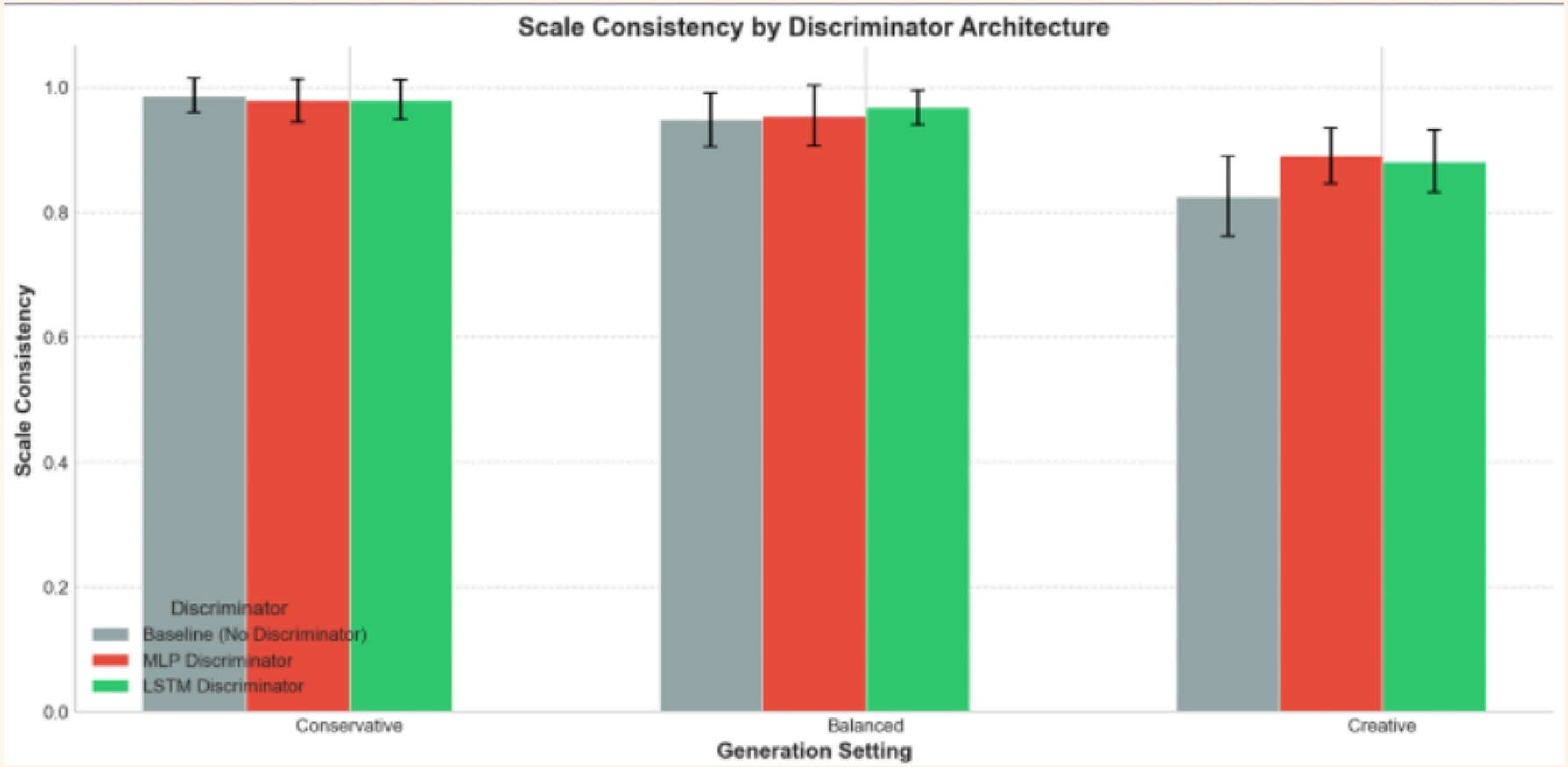
DISCRIMINATOR ARCHITECTURE

The baseline naïve LSTM model generated music with:

- No discriminator guidance
- A MLP discriminator model
- A LSTM discriminator model



DISCRIMINATOR ARCHITECTURE



DISCRIMINATOR ARCHITECTURE

Findings:

- **Scale consistency decreased with higher temperature, regardless of guidance.**
- **For conservative generation, having no guidance resulted in the lowest mean pitch class entropy (the guided models had more variation).**
- **Discriminator guidance during balanced and creative settings led to lower note variation than unguided.**


DEMO
VIDEO:


-MIDI

WRAPPING UP

Across all the experiments the following seemed to be consistent:

1. Tokenization strongly influences generalization. Naïve tokens work much better than miditok for small datasets because they require less context.
2. Augmentation improves the performance of midtok models.
3. Higher creativity settings like temperature expand pitch range but can also lead to incoherent music.
4. Discriminator guidance improves correctness but reduces expressiveness. It filters out note selections that aren't harmonically "correct", leaving boring note selections at times.
5. Model architecture seems to matter less than tokenization and dataset size. Transformer performed the worst on the small dataset.

WRAPPING UP

- **Naïve tokenization with LSTM models offer the best performance for small datasets (compared to the other tested configurations).**
- **Miditok models require larger datasets to generalize effectively.**
- **Discriminator guidance improves theoretical musical correctness, but may sound worse or less interesting.**
- **Transformers aren’t suitable for small symbolic music datasets.**

SOURCES

- https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2
- Zheng Jiang (2019) Automatic Analysis of Music in Standard MIDI Files
- Zhu, Y., Baca, J., Rekabdar, B., Rawassizadeh, R. (2023). A Survey of AI Music Generation Tools and Models
- Briot, J., Hadjeres, G., Pachet, F. (2017). Deep Learning Techniques for Music Generation -- A Survey
- Bhandari, K., Roy, A., Wang, K., Puri, G., Colton, S., Herremans, D. (2024). Text2midi: Generating Symbolic Music from Captions
- Yang, L., Chou, S., Yang, Y. (2017). MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation
- Tian, S., Zhang, C., Yuan, W., Tan, W., Zhu, W. (2025). XMusic: Towards a Generalized and Controllable Symbolic Music Generation Framework
- Colin Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching". _PhD Thesis_, 2016