Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation

Hao-Wen Dong and Yi-Hsuan Yang

Research Center of IT Innovation, Academia Sinica

Outlines

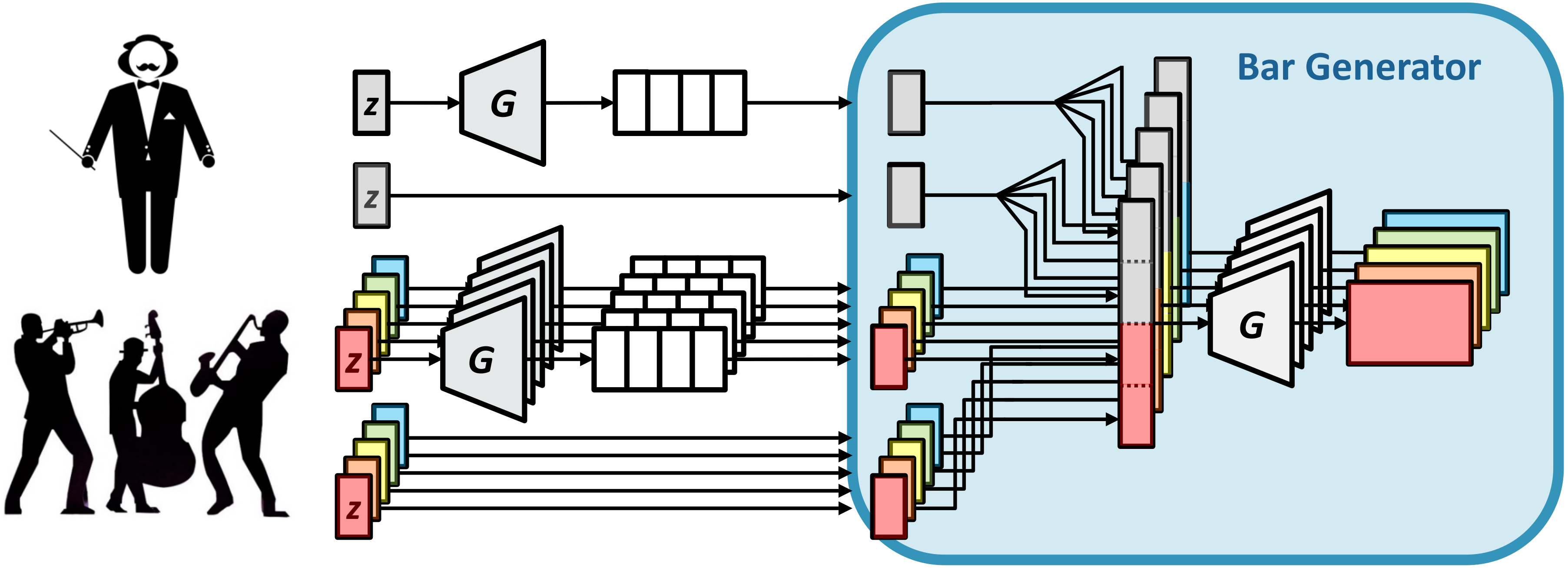
- Introduction
- Binary Neurons
- Proposed Model
- Data
- Results
- Future Works

Source Code https://github.com/salu133445/bmusegan
Demo Page https://salu133445.github.io/bmusegan/

Introduction

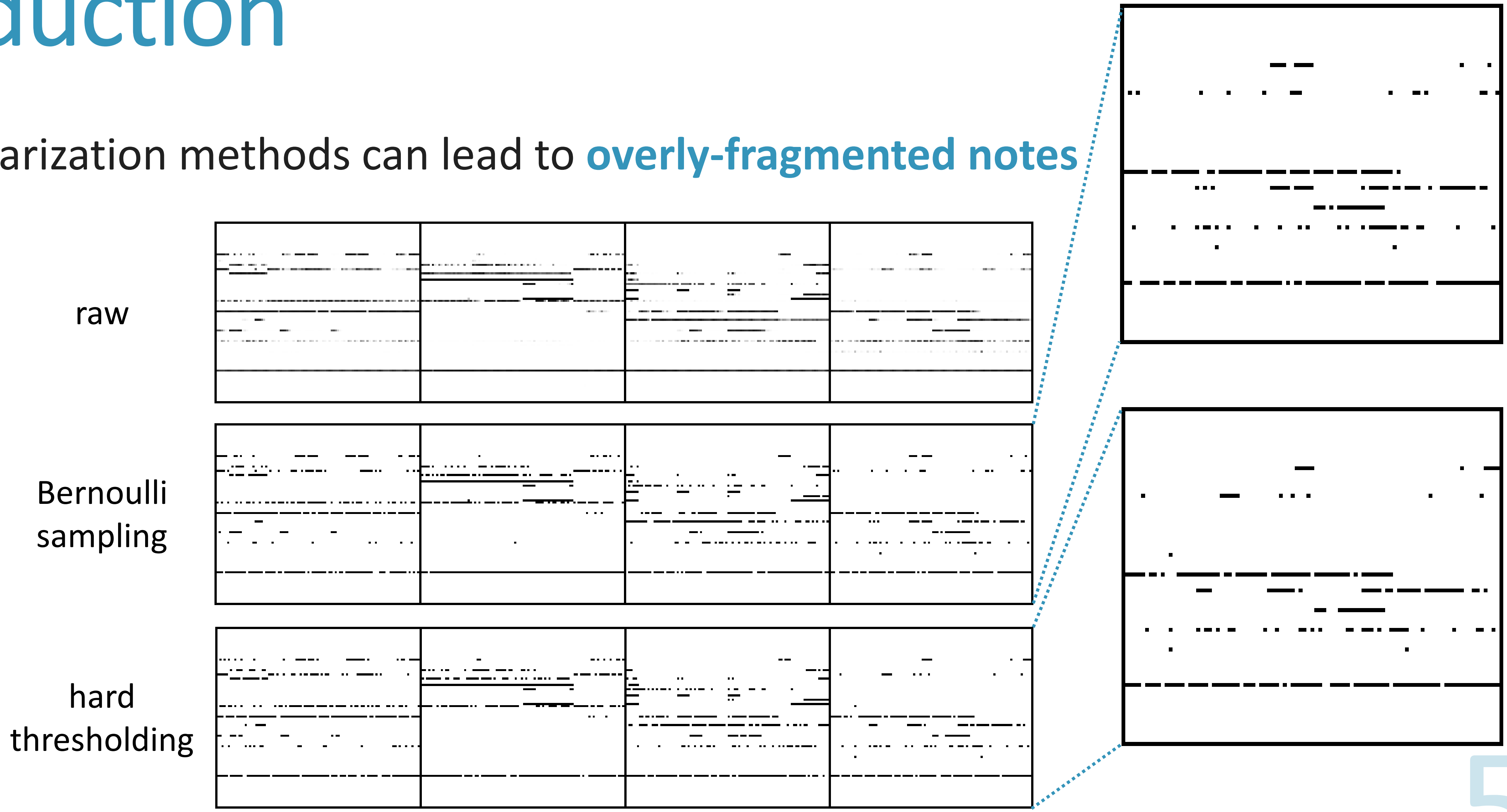
Introduction

- **MuseGAN**
  - can only generate **real-valued predictions**
  - require **postprocessing at test time**
    (e.g., hard thresholding or Bernoulli sampling)



Introduction

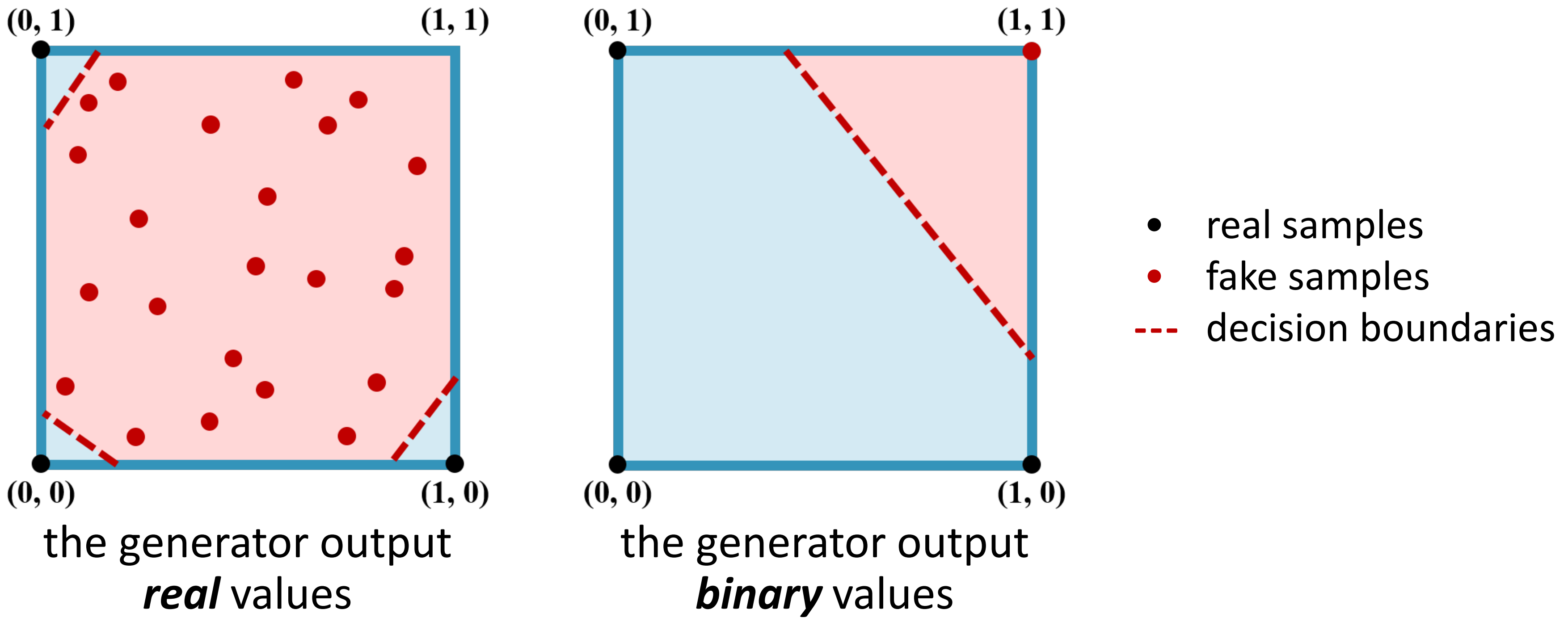
- Naïve binarization methods can lead to **overly-fragmented notes**



Introduction

- Real-valued predictions can lead to **training difficulties of the discriminator**

Decision boundaries to learn for *the discriminator*



Binary Neurons

Binary Neurons

- Neurons that output binary-valued predictions
- In this work, we consider
  - **deterministic binary neurons (DBNs)**

$$DBN(x) = \begin{cases} 1, & if\ \sigma(x) > 0.5 \\ 0, & otherwise \end{cases}$$

  - **stochastic binary neurons (SBNs)**

$$SBN(x) = \begin{cases} 1, & if\ z < \sigma(x) \\ 0, & otherwise \end{cases}, \qquad z \sim U[0, 1]$$

Gradient Estimators

- Computing the exact gradients for binary neurons is intractable
- **Straight-through (ST) estimator**
    - treat BNs as *identity functions* in the backward pass

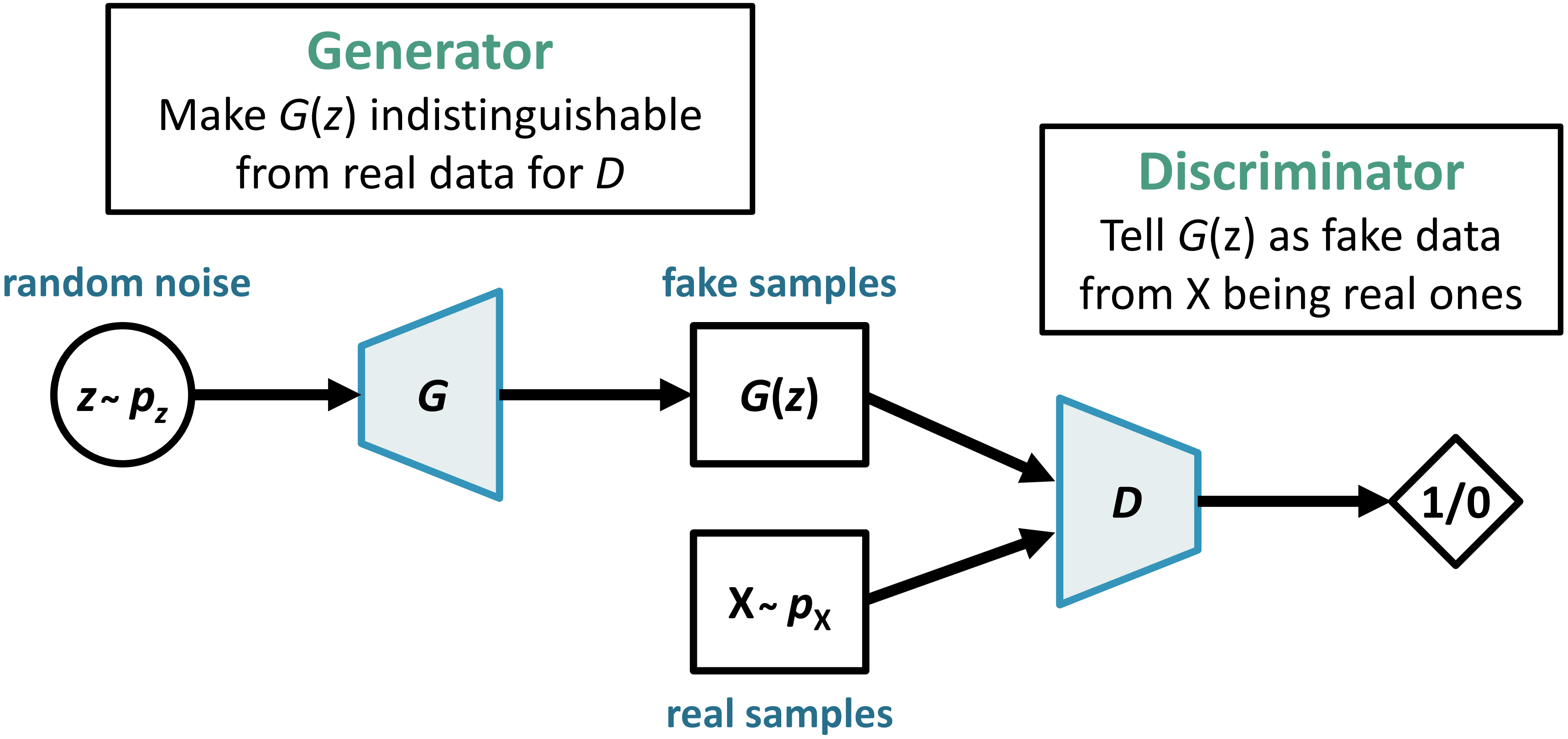
$$\frac{\partial BN(x)}{\partial x} = 1$$

- **Sigmoid-adjusted ST estimator**
    - treat BNs as *identity functions multiplied by the derivative of the sigmoid function* in the backward pass

$$\frac{\partial BN(x)}{\partial x} = \sigma(x)$$

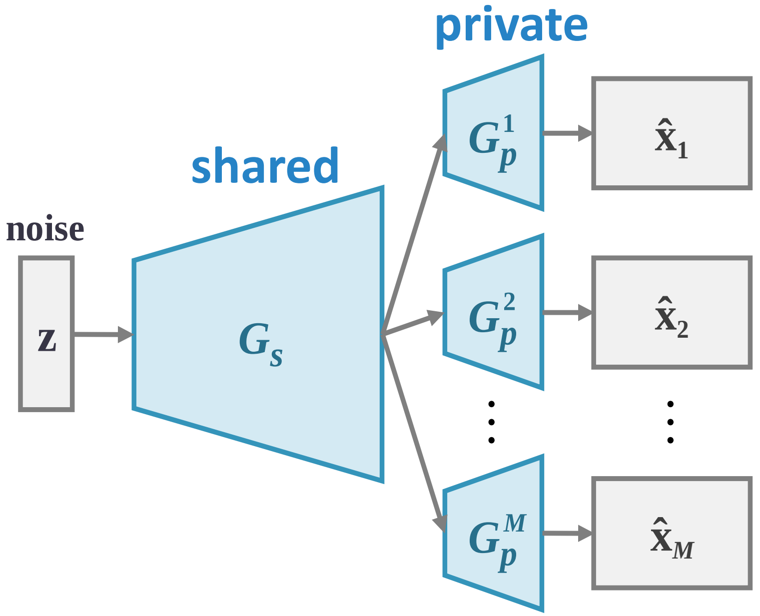
Proposed Model

Generative Adversarial Networks



Generator

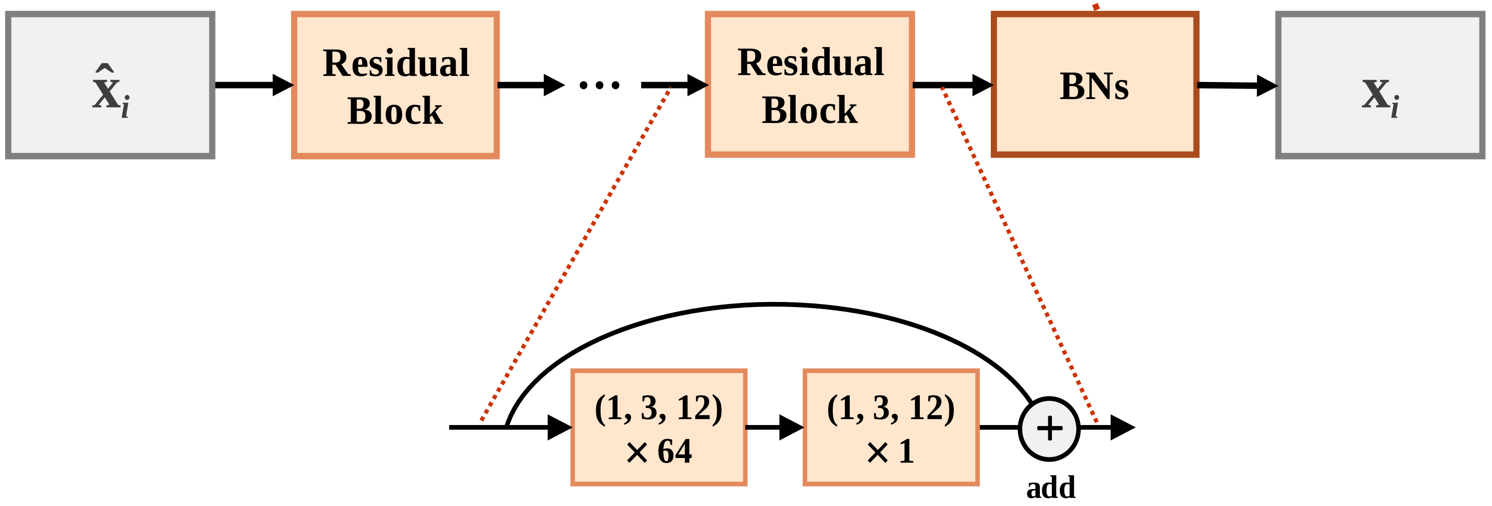
- One single input random vector
- ***Shared/private design***
  - Different tracks have their own musical properties (e.g. textures, patterns, techniques)
  - Jointly all tracks follow a common, high-level musical idea



Refiner

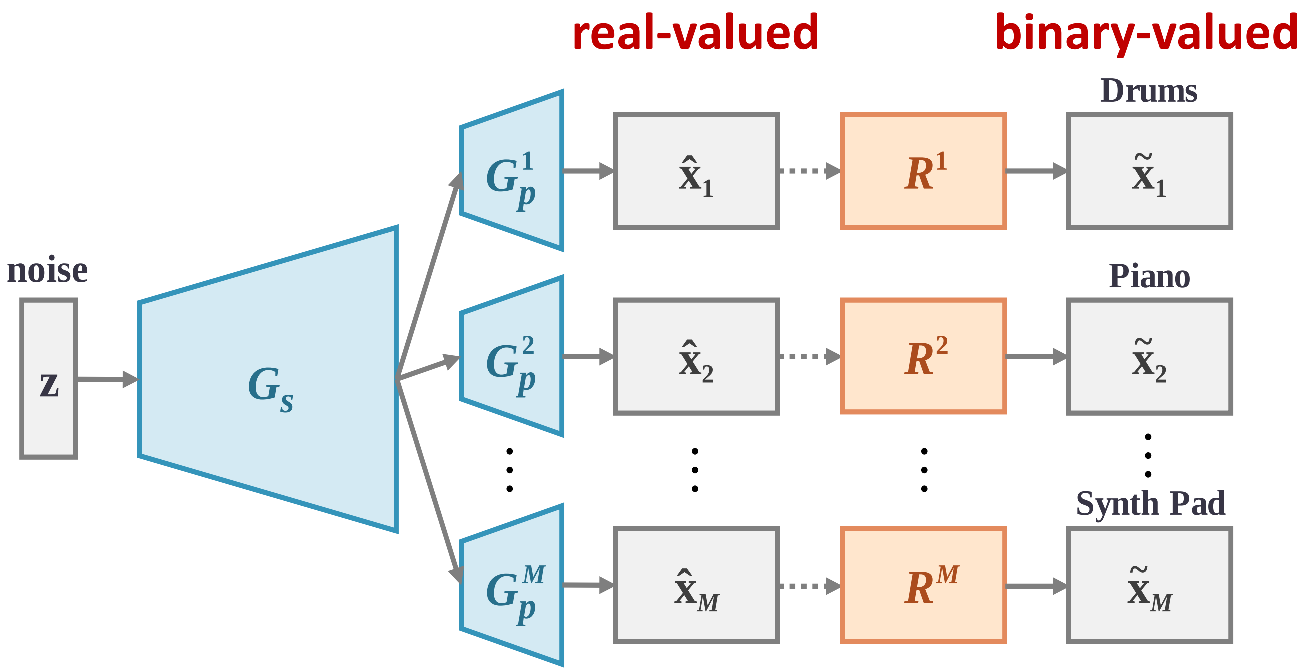
- Refine the real-valued outputs of the generator into binary ones
- Composed of a number of ***residual units***

DBNs — *deterministic* binary neurons
SBNs — *stochastic* binary neurons



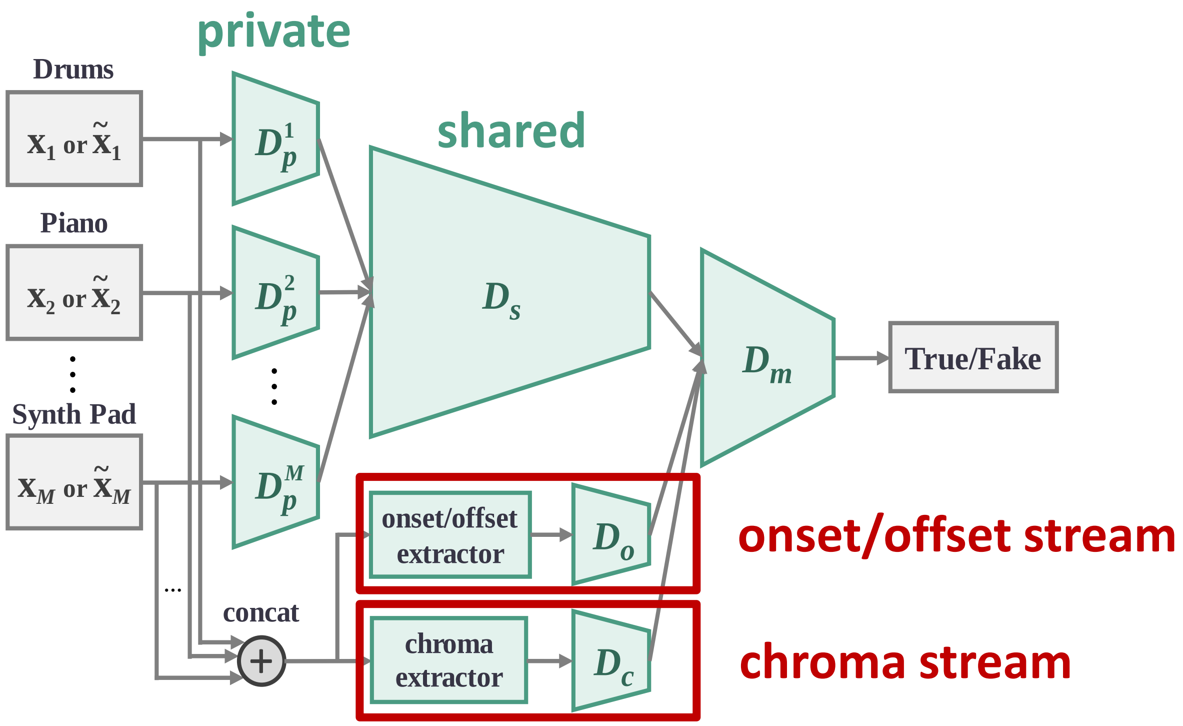
Refiner

- Refine the real-valued outputs of the generator into binary ones
- Composed of a number of ***residual units***



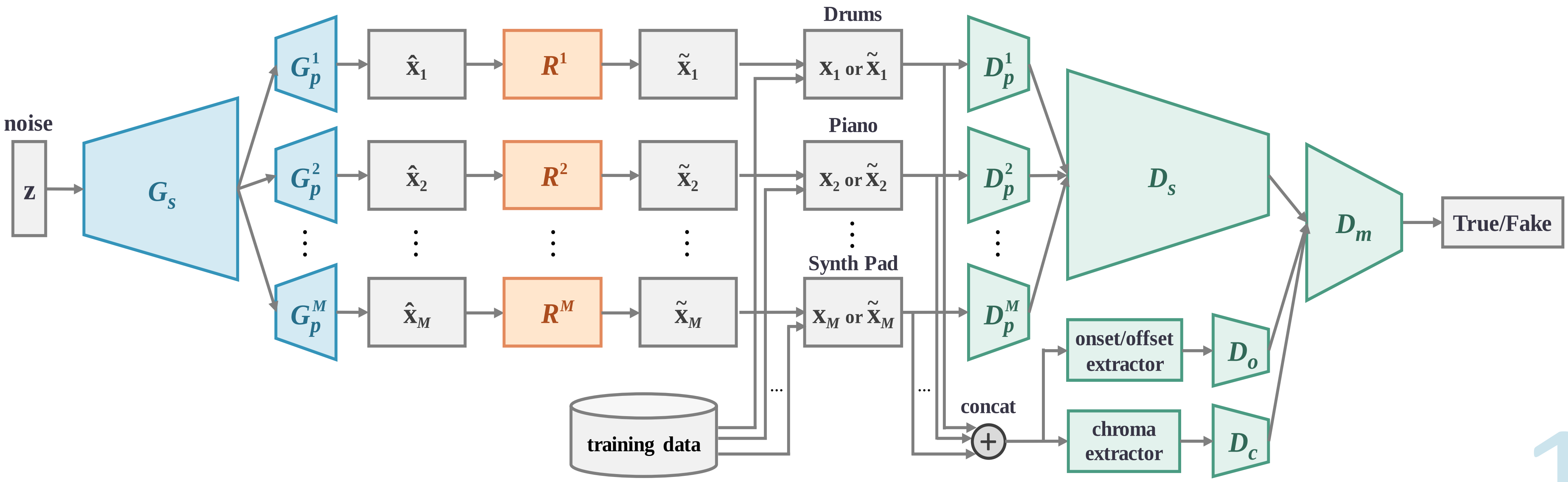
Discriminator

- ***Shared/private design*** (similar to the generator)
- Additional ***onset/offset stream*** and ***chroma stream***



Two-stage Training

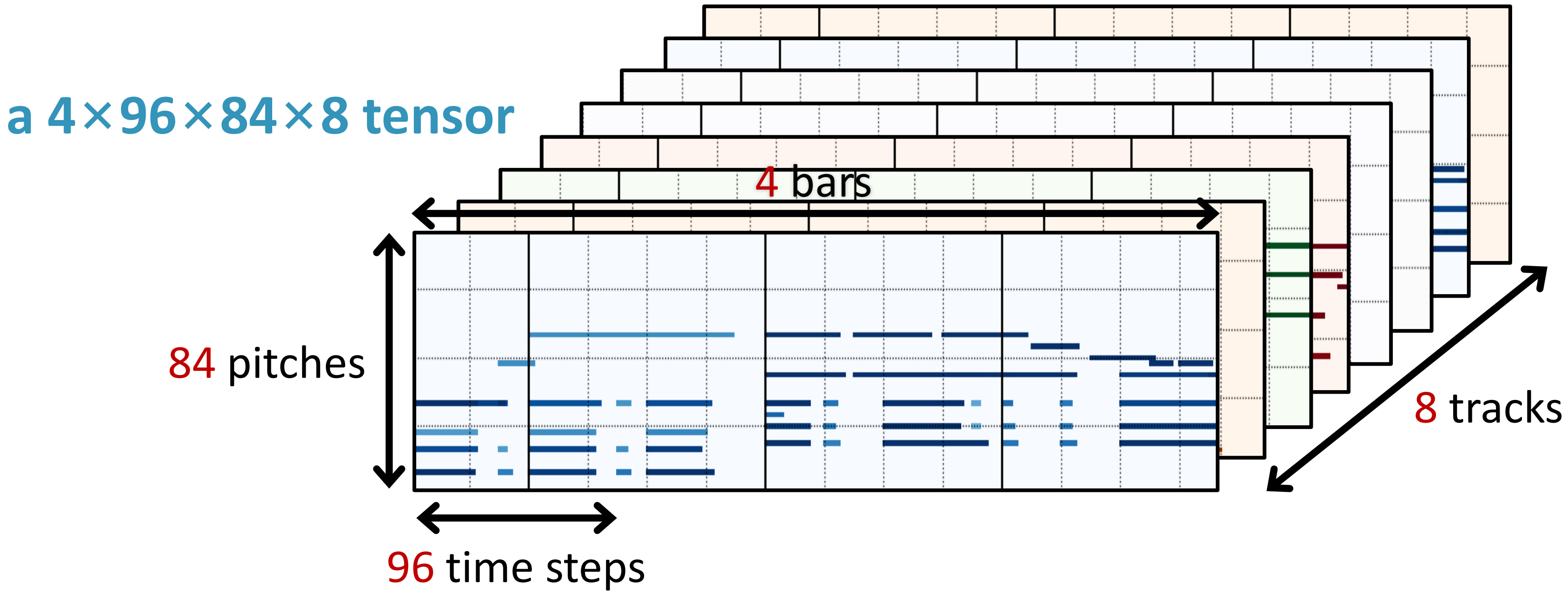
- First stage — pretrain the ***generator*** and ***discriminator***
- Second stage — train *the **refiner*** and ***discriminator*** (*with G fixed*)



Data

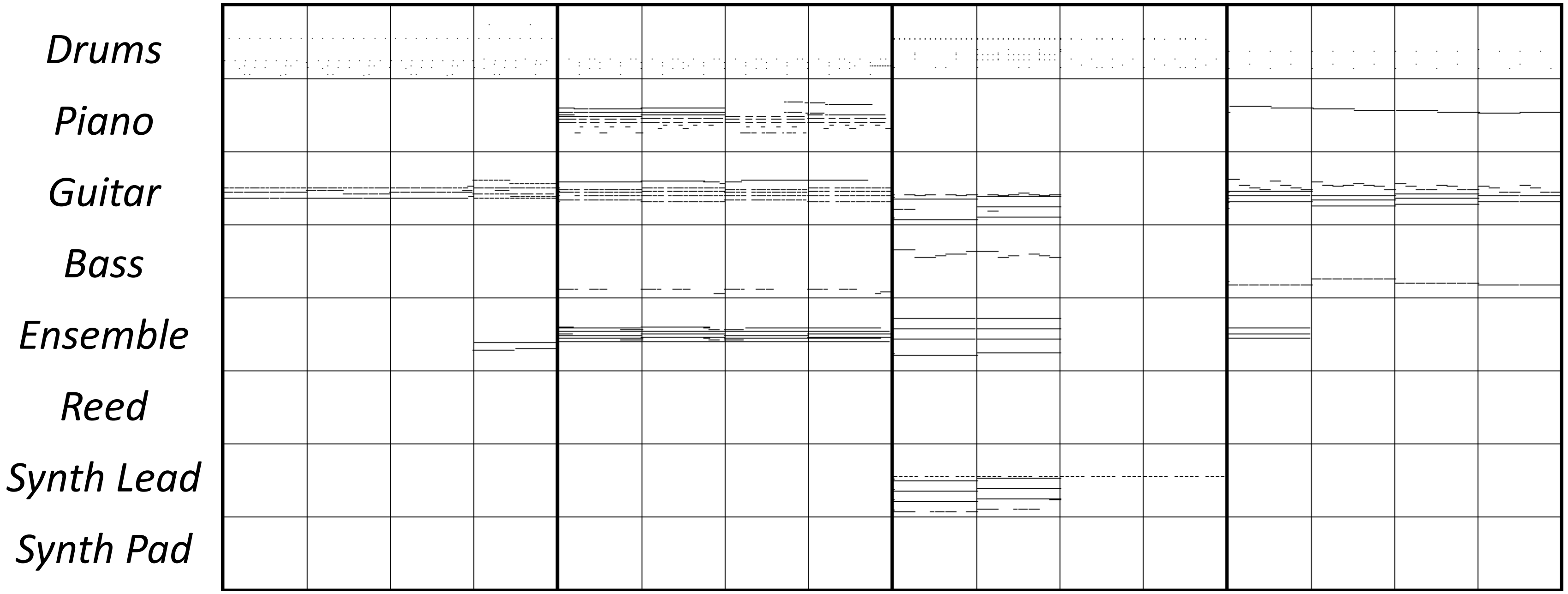
Data Representation

- **Multi-track piano-roll**
- **8 tracks**
  - *Drums*, *Piano*, *Guitar*, *Bass*, *Ensemble*, *Reed*, *Synth Lead* and *Synth Pad*



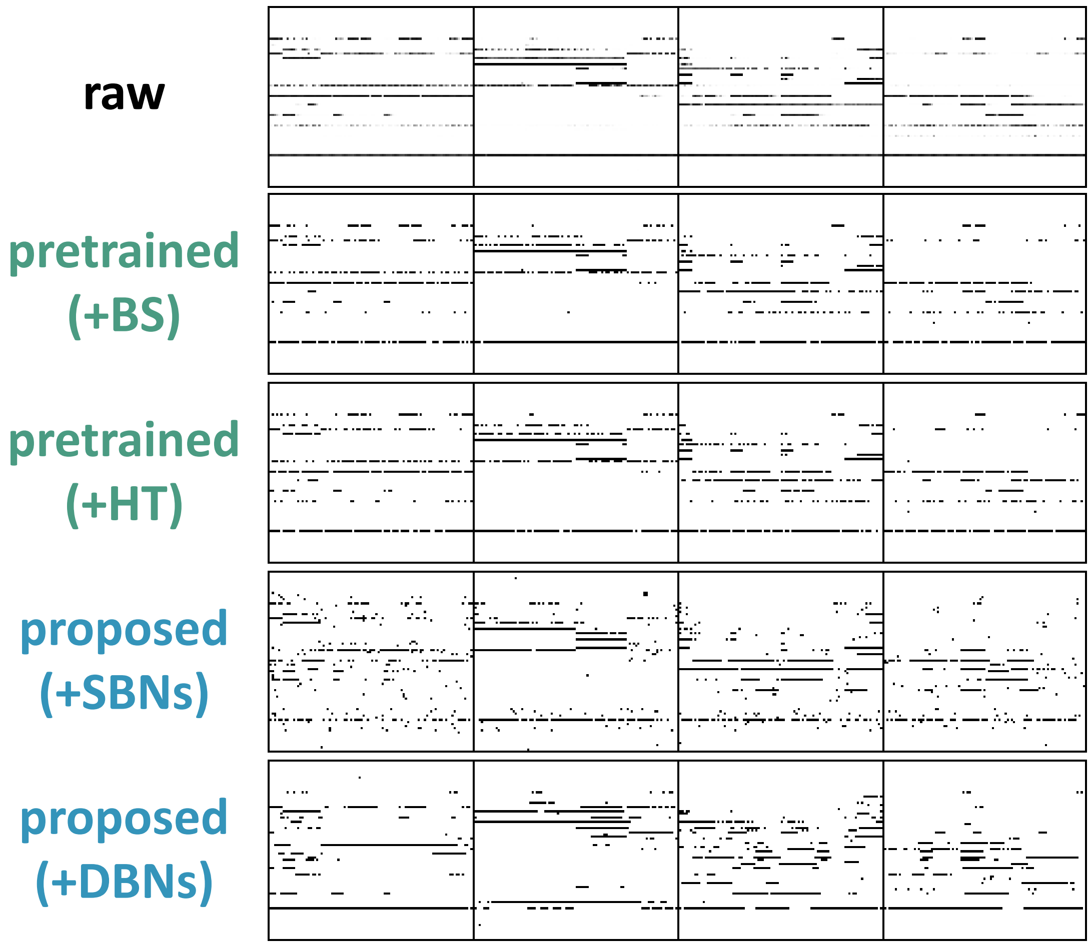
Training Data

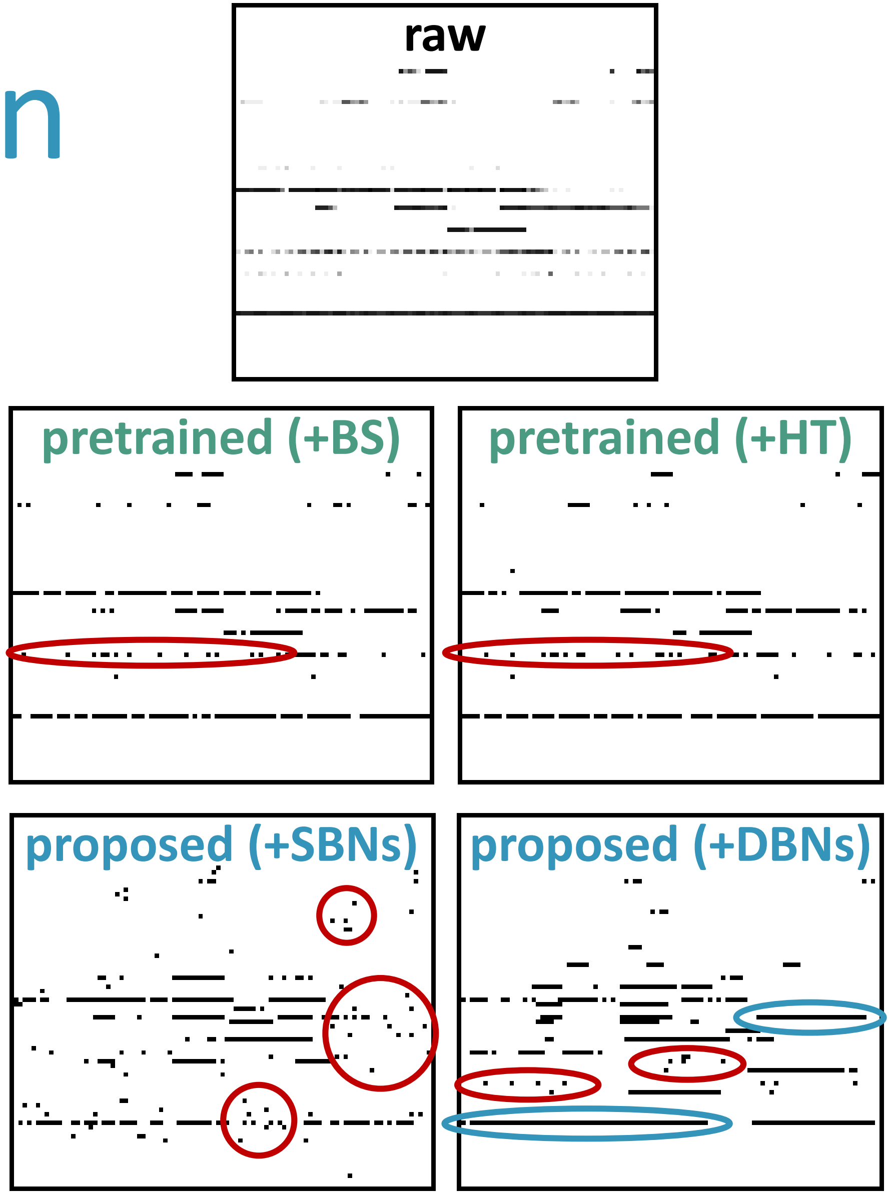
- ***Lakh Pianoroll Dataset*** (LPD)
- **13746 *four-bar phrases*** from 2291 songs (six for each)
  - Pick only songs in 4/4 time and with an *alternative* tag



Results

Qualitative Comparison





Audio Samples

- **proposed (+DBNs)** — fewer overly-fragmented notes; more out-of-scale notes
- **proposed (+SBNs)** — more overly-fragmented notes ; lots of artifacts

	Sample 1	Sample 2
pretrained (+BS)		
pretrained (+HT)		
proposed (+DBNs)		

	Sample 1	Sample 2
proposed (+SBNs)		

More samples available on demo page
https://salu133445.github.io/bmusegan/

Evaluation Metrics

- **Qualified note rate (QN)**

$$\text{QN} = \frac{\text{\# of notes no shorter than 3 time steps (i.e., a 32th note)}}{\text{\# of notes}}$$

- **Polyphonicity (PP)**

$$\text{PP} = \frac{\text{\# of time steps where more than two pitches are played}}{\text{\# of time steps}}$$

- **Tonal distance (TD)**
    - measure the distance between two chroma features in a tonal space

Comparisons of Training Strategies

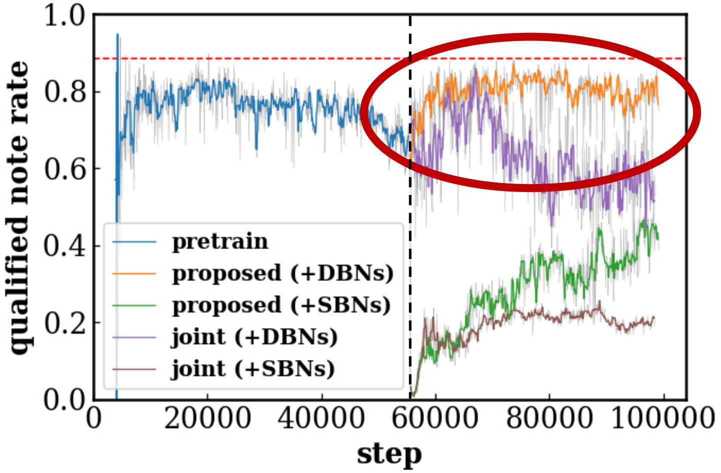
- **Two-stage training** (*proposed*) — *[stage 1] pretrain G and D [stage 2] train R and D*
- **Joint training** (*joint*) — *[stage 1] pretrain G and D [stage 2] train G, R and D*
- **End-to-end training** (*end-to-end*) — *train G, R and D in one stage*

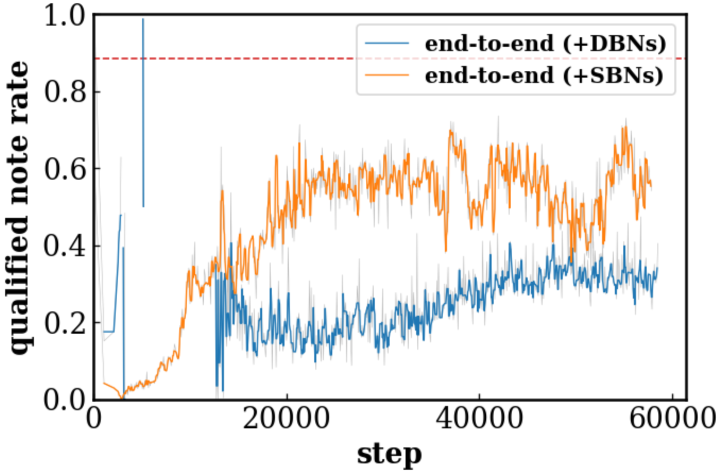
	training data	pretrained		proposed		joint		end-to-end	
		BS	HT	SBNs	DBNs	SBNs	DBNs	SBNs	DBNs
QN	0.88	**0.67**	**0.72**	0.42	**0.78**	0.18	0.55	**0.67**	0.28
PP	0.48	0.20	0.22	**0.26**	**0.45**	0.19	0.19	0.16	**0.29**
TD	0.96	**0.98**	1.00	**0.99**	0.87	**0.95**	1.00	1.40	1.10

(values closer to that of the training data is better; **underline**: closest; **bold**: top 3 closest)

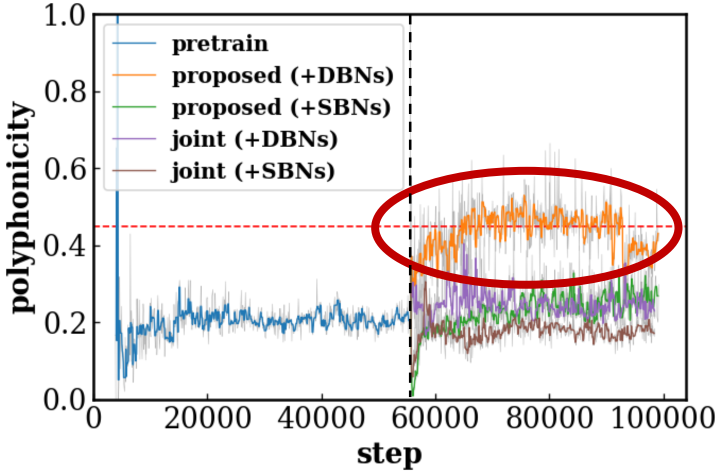
Comparisons of Training Strategies

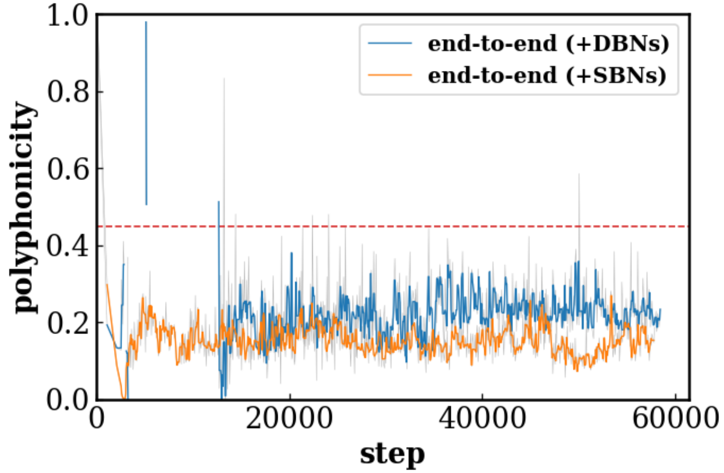
QN





PP

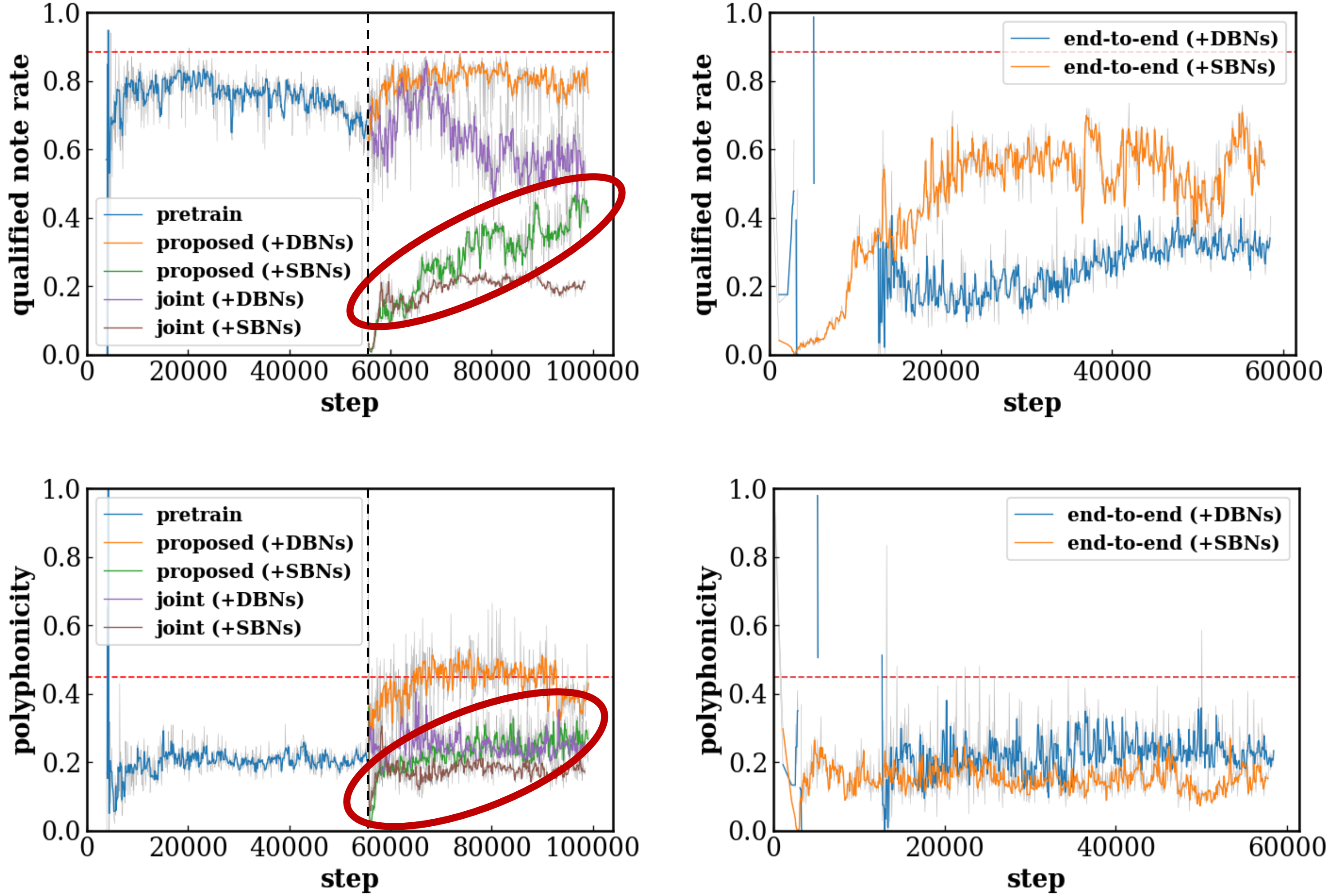




Comparisons of Training Strategies

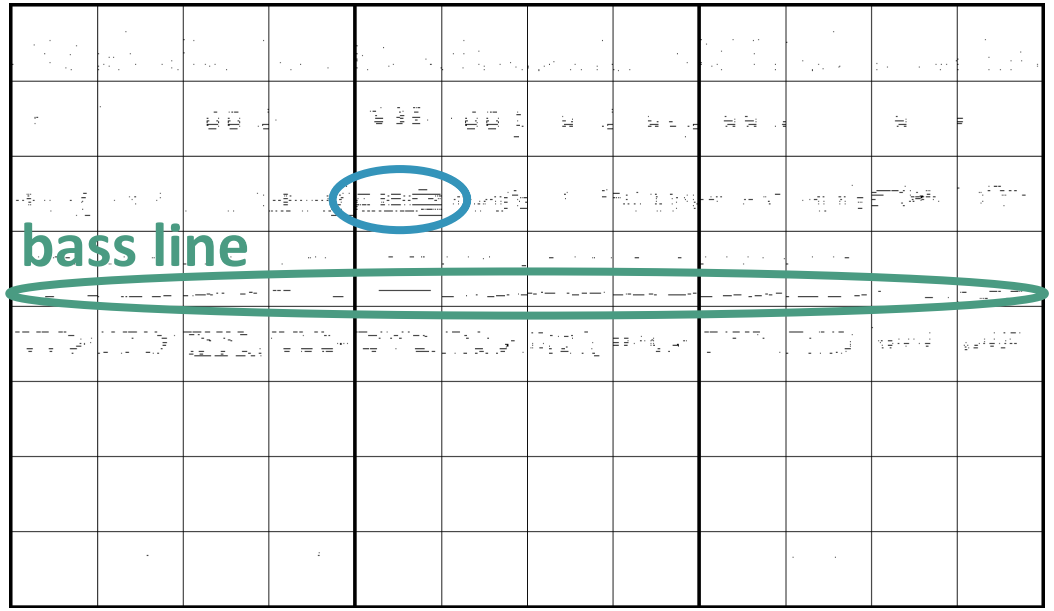
QN

PP

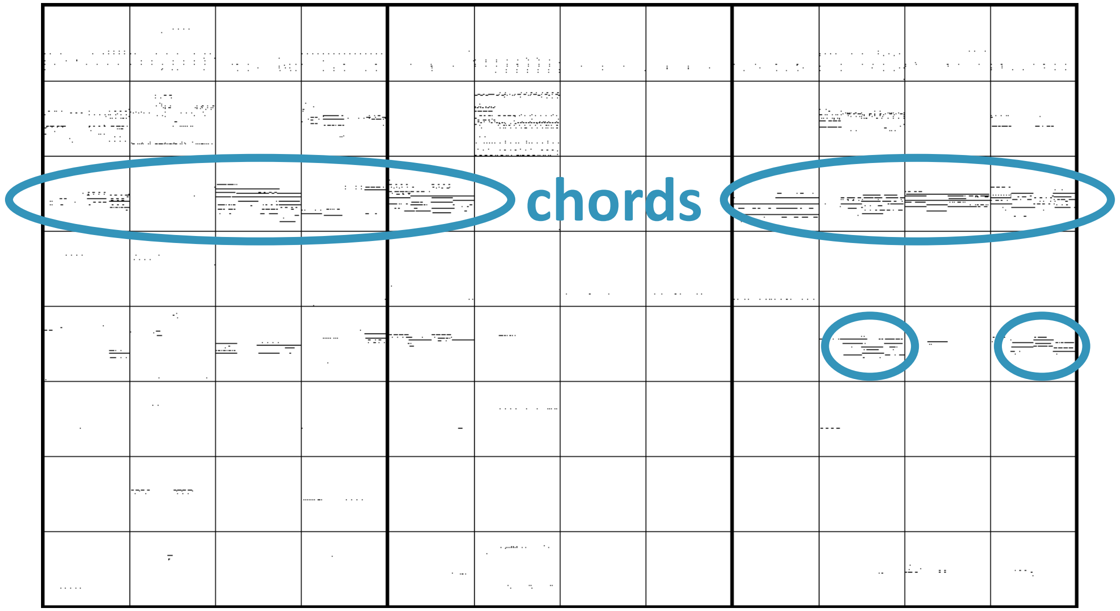


End-to-end Models

- First attempt, to our best knowledge, to **generate such high-dimensional data with binary neurons from scratch**

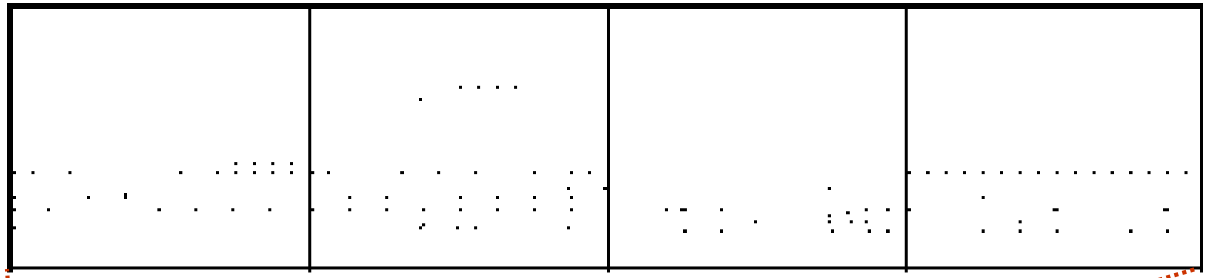


SBNs

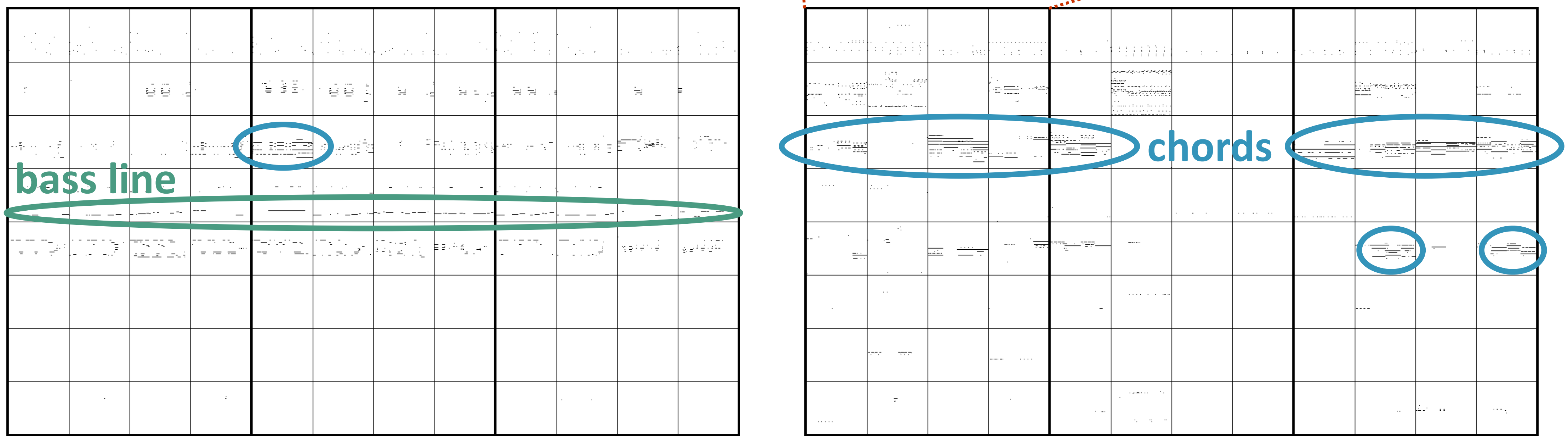


DBNs

End-to-end Models



- First attempt, to our best knowledge, to **generate such high-dimensional data with binary neurons from scratch**



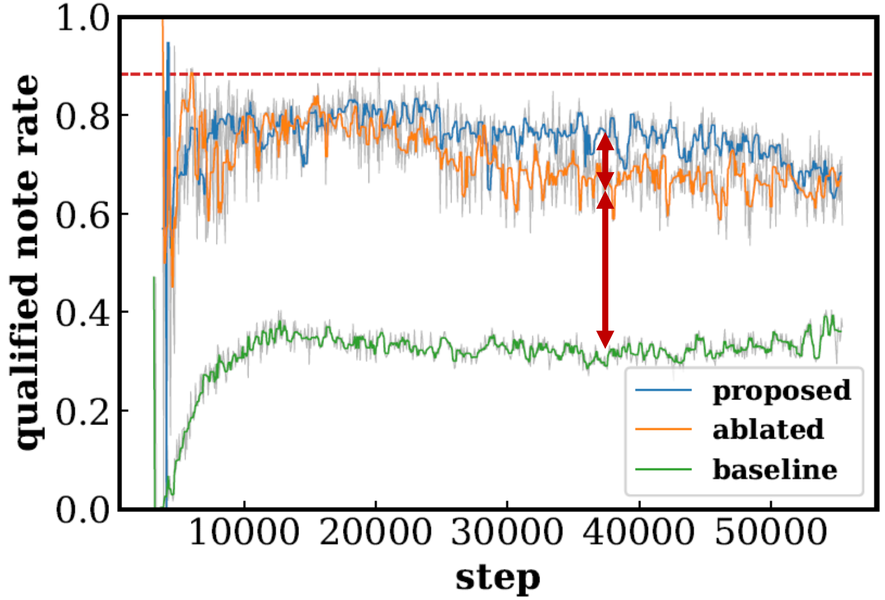
SBNs

DBNs

Effects of the Discriminator Design

- **pretrained** — *shared/private design + offset/onset stream + chroma stream*
- **ablated** — *shared/private design*
- **baseline** — *only one shared discriminator*

	training data	pretrained		ablated		baseline	
		BS	HT	BS	HT	BS	HT
QN	0.88	**0.67**	**0.72**	0.61	0.64	0.35	0.37
PP	0.48	0.20	0.22	0.19	0.20	0.14	0.14
TD	0.96	**0.98**	1.00	1.00	1.00	1.30	1.40



Future Works

Summary

- A convolutional GAN for **binary-valued multi-track piano-rolls**
  - **CNNs** + **residual units** + **binary neurons**
  - **Shared/private design** in both the generator and discriminator (proved effective)
  - **Onset/offset** and **chroma streams** in the discriminator (proved effective)
  - **Two-stage training** (proved effective)

- **Proposed model with *deterministic binary neurons* (DBNs)** features **fewer overly-fragmented notes** as compared with existing methods.

Future Works

- **Tradeoff**
  - *easy-to-train* generator + *hard-to-train* discriminator
  - *hard-to-train* generator + *easy-to-train* discriminator
- **Longer music**
  - RNNs (LSTMs or GRUs)
  - How to generate high-level/long-term structure?
- **More tracks**
  - Symphony/orchestra compositions
  - (hierarchical) sections → sub-sections → instruments

Thank you for your attention