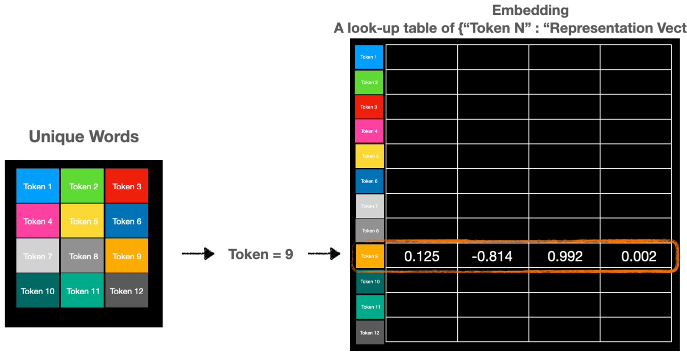
Designing NN-Based Generative Models of Music (Part 3)

Computational Music Creativity, SMC (2022/23)

Prepared by Behzad Haki, February 2023

Architecture

Transformers are designed to work with tokens! But, we want to represent our inputs more directly without tokenization! We should adapt the I/O layers!



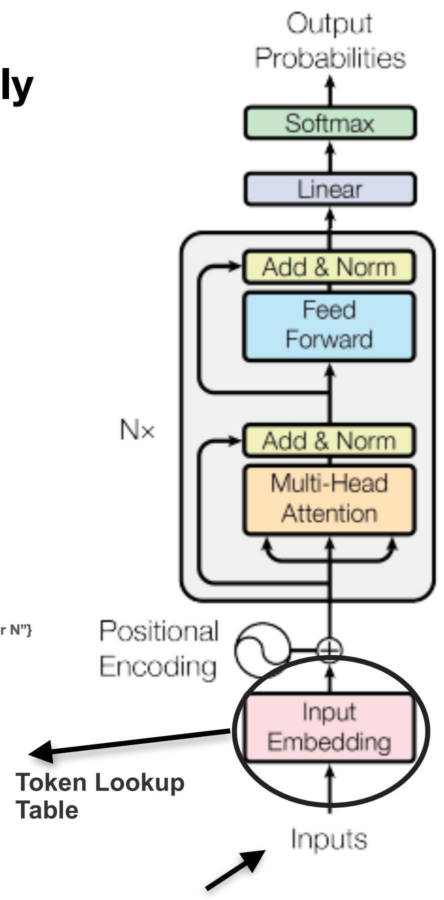
Unique Words

Token 1	Token 2	Token 3
Token 4	Token 5	Token 6
Token 7	Token 8	Token 9
Token 10	Token 11	Token 12

Embedding
A look-up table of {“Token N” : “Representation Vector N”}

Token 1				
Token 2				
Token 3				
Token 4				
Token 5				
Token 6				
Token 7				
Token 8				
Token 9	0.125	-0.814	0.992	0.002
Token 10				
Token 11				
Token 12				

KickOnset, Velocity_100, *Silence_0.25*,
HiHatOnset, Velocity_80, *Silence_2*,
HiHatOnset, Velocity_0.4, …

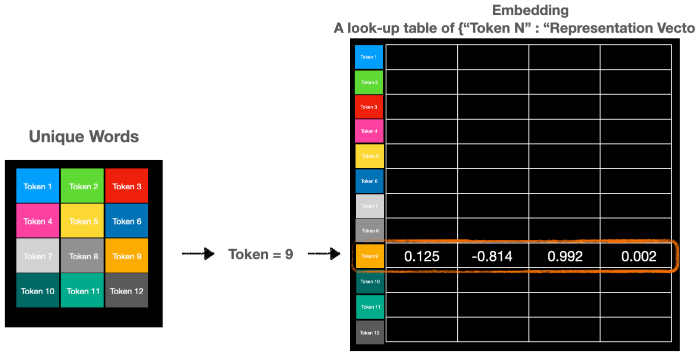


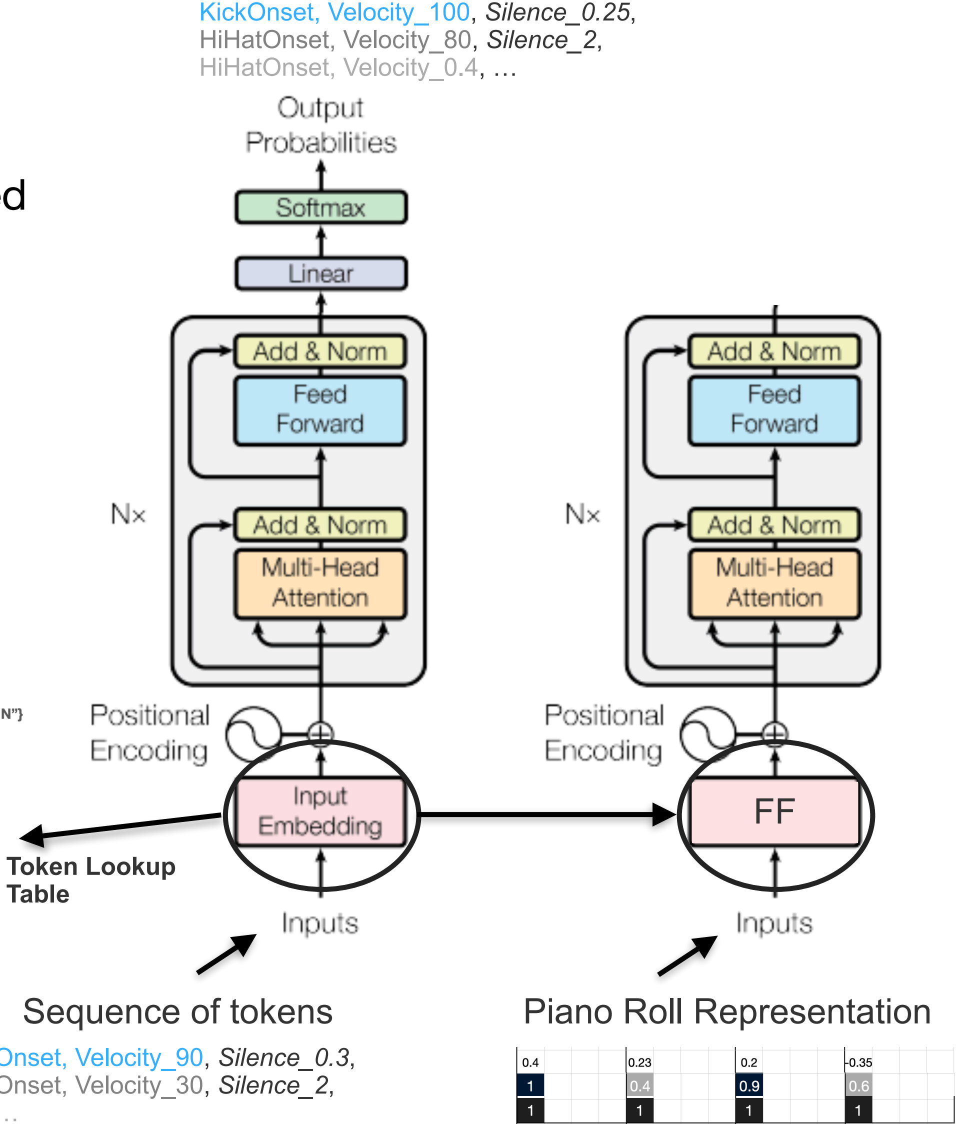
Sequence of tokens

Onset, Velocity_90, *Silence_0.3*,
Onset, Velocity_30, *Silence_2*,
…

Architecture:
Transformer for Piano Rolls Input/Outputs

Step 1: Replace Embedding layer With a Feed-forward fully connected layer




KickOnset, Velocity_100, *Silence_0.25*,
HiHatOnset, Velocity_80, *Silence_2*,
HiHatOnset, Velocity_0.4, …

Sequence of tokens

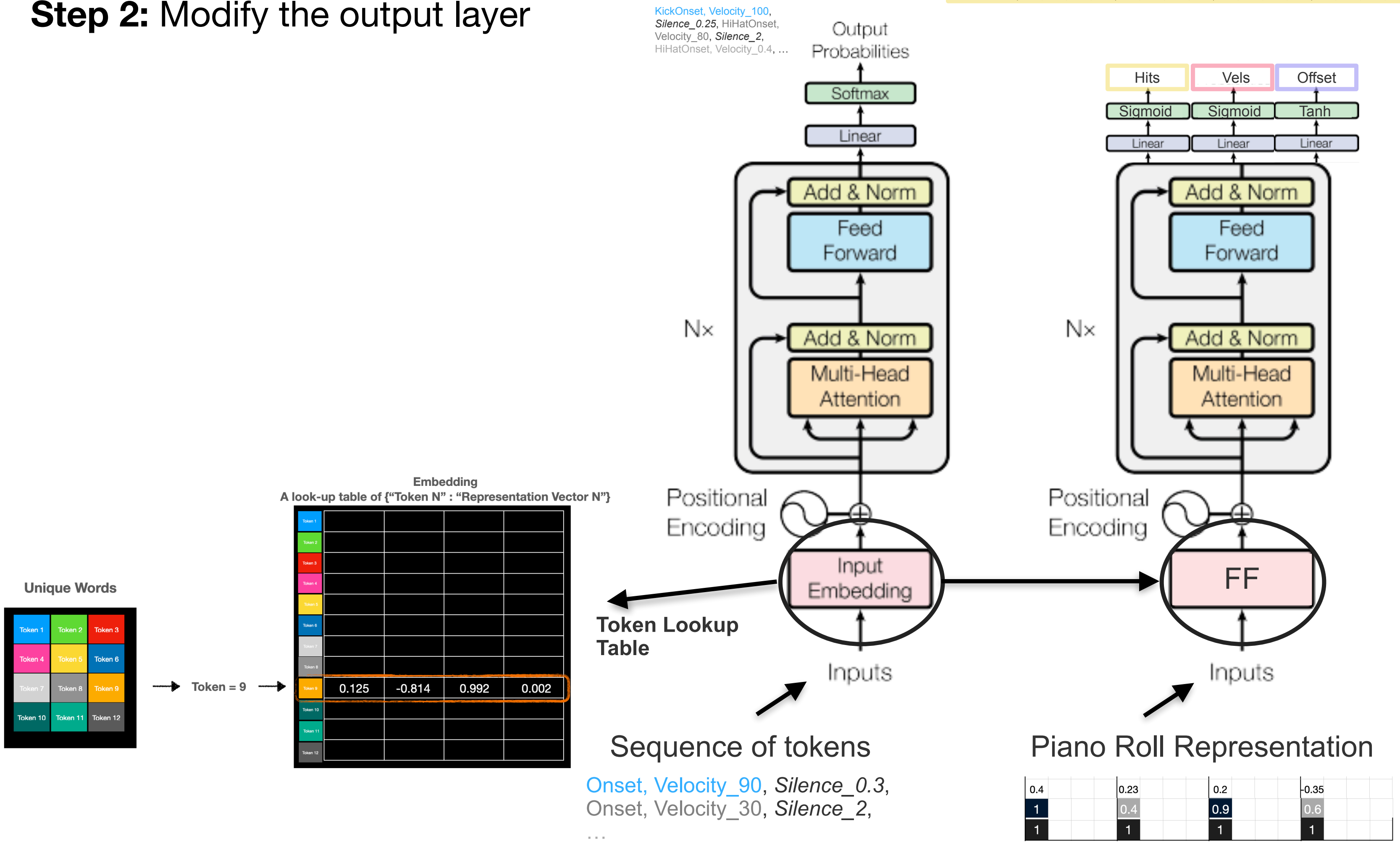
Onset, Velocity_90, *Silence_0.3*,
Onset, Velocity_30, *Silence_2*,
…

Piano Roll Representation

Architecture:
Transformer for Piano Rolls Input/Outputs

Hi-hat uTiming	0.2		0.42		-0.31		0.42		0.4		0		-0.12		0.18	
Snare uTiming													-0.1			
Kick uTiming	0.4				0.23				0.2				-0.35			
Hi-hat Velocity	1		0.7		0.78		0.62		0.73		0.54		0.89		0.51	
Snare Velocity													0.64			
Kick Velocity	1				0.4				0.9				0.6			
Hi-hat	1		1		1		1		1		1		1		1	
Snare													1			
Kick	1				1				1				1			

Step 2: Modify the output layer



KickOnset, Velocity_100, *Silence_0.25*, HiHatOnset, Velocity_80, *Silence_2*, HiHatOnset, Velocity_0.4, …

Unique Words

Token 1	Token 2	Token 3
Token 4	Token 5	Token 6
Token 7	Token 8	Token 9
Token 10	Token 11	Token 12

Embedding
A look-up table of {"Token N" : "Representation Vector N"}

Token 1				
Token 2				
Token 3				
Token 4				
Token 5				
Token 6				
Token 7				
Token 8				
Token 9	0.125	-0.814	0.992	0.002
Token 10				
Token 11				
Token 12				

Token Lookup Table

Sequence of tokens

Onset, Velocity_90, *Silence_0.3*,
Onset, Velocity_30, *Silence_2*,
…

Piano Roll Representation

0.4				0.23				0.2				-0.35			
1				0.4				0.9				0.6			
1				1				1				1			

Architecture

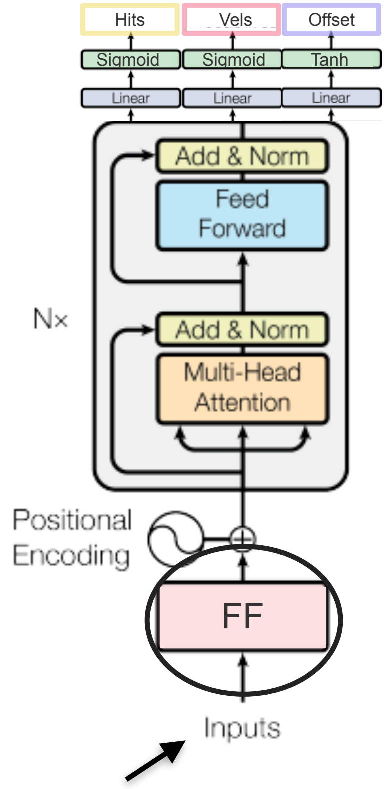
A few limitations:

1. We haven’t implemented any control parameters for the generations
2. We can’t generate any variations for a given input! (For one input groove, we get a single drum pattern)
3. We can’t generate from scratch using this model!

We’ll try to make improvements to the architecture by modifying it into a Variational Auto-encoder (VAE)

Drum Pattern

Hi-hat uTiming	0.2		0.42		-0.31		0.42		0.4		0		-0.12		0.18	
Snare uTiming													-0.1			
Kick uTiming	0.4				0.23				0.2				-0.35			
Hi-hat Velocity	1		0.7		0.78		0.62		0.73		0.54		0.89		0.51	
Snare Velocity													0.64			
Kick Velocity	1				0.4				0.9				0.6			
Hi-hat	1		1		1		1		1		1		1		1	
Snare													1			
Kick	1				1				1				1			

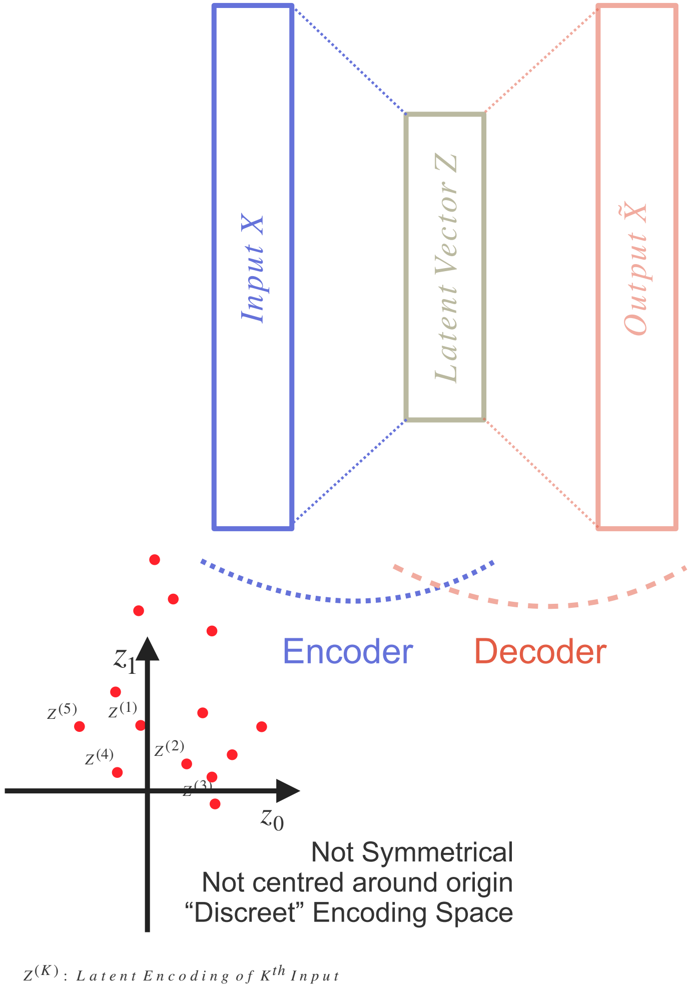


Input Monotonic Groove

0.4				0.23				0.2				-0.35			
1				0.4				0.9				0.6			
1				1				1				1			

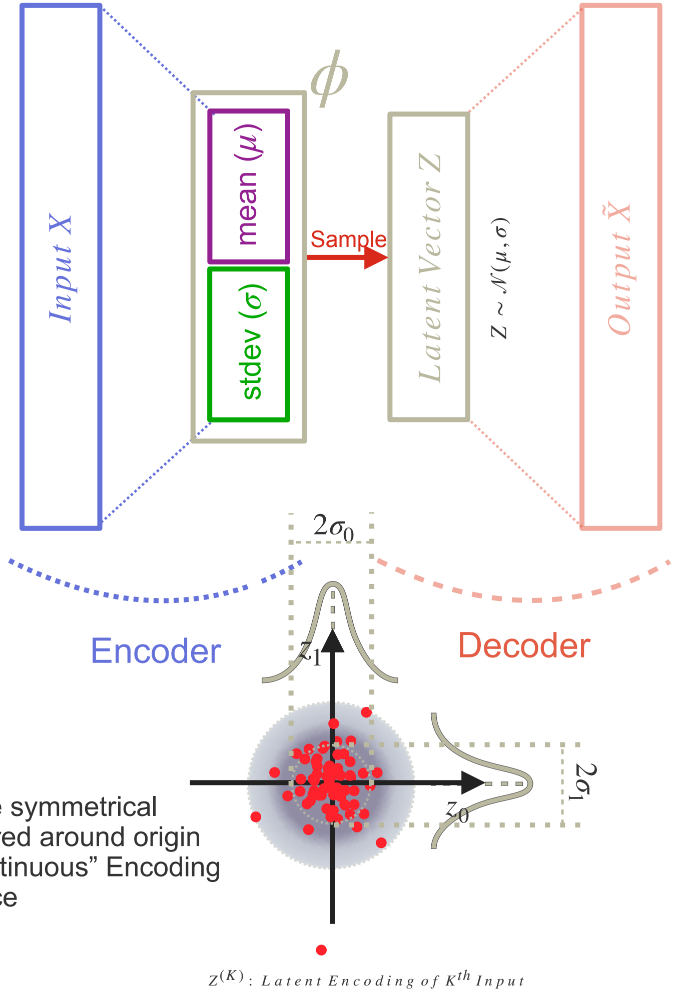
Architecture

Auto-Encoder



Not Symmetrical
Not centred around origin
“Discreet” Encoding Space

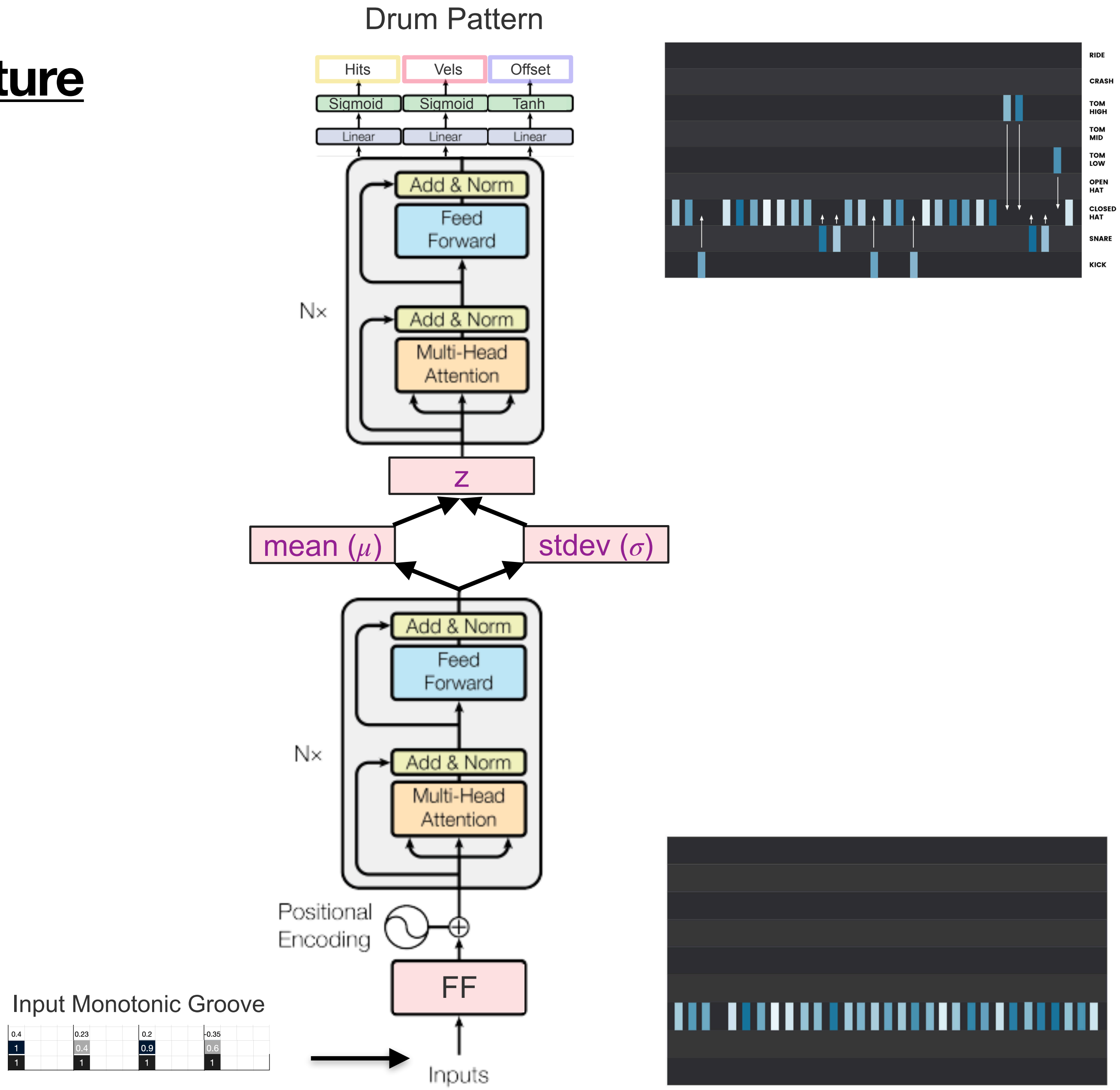
Variational Auto-Encoder



More symmetrical
centred around origin
“Continuous” Encoding
Space

Read this if you’re interested: https://towardsdatascience.com/generating-new-faces-with-variational-autoencoders-d13cfcb5f0a8

Architecture



Training

Two sets of Parameters:

1. Trainable:

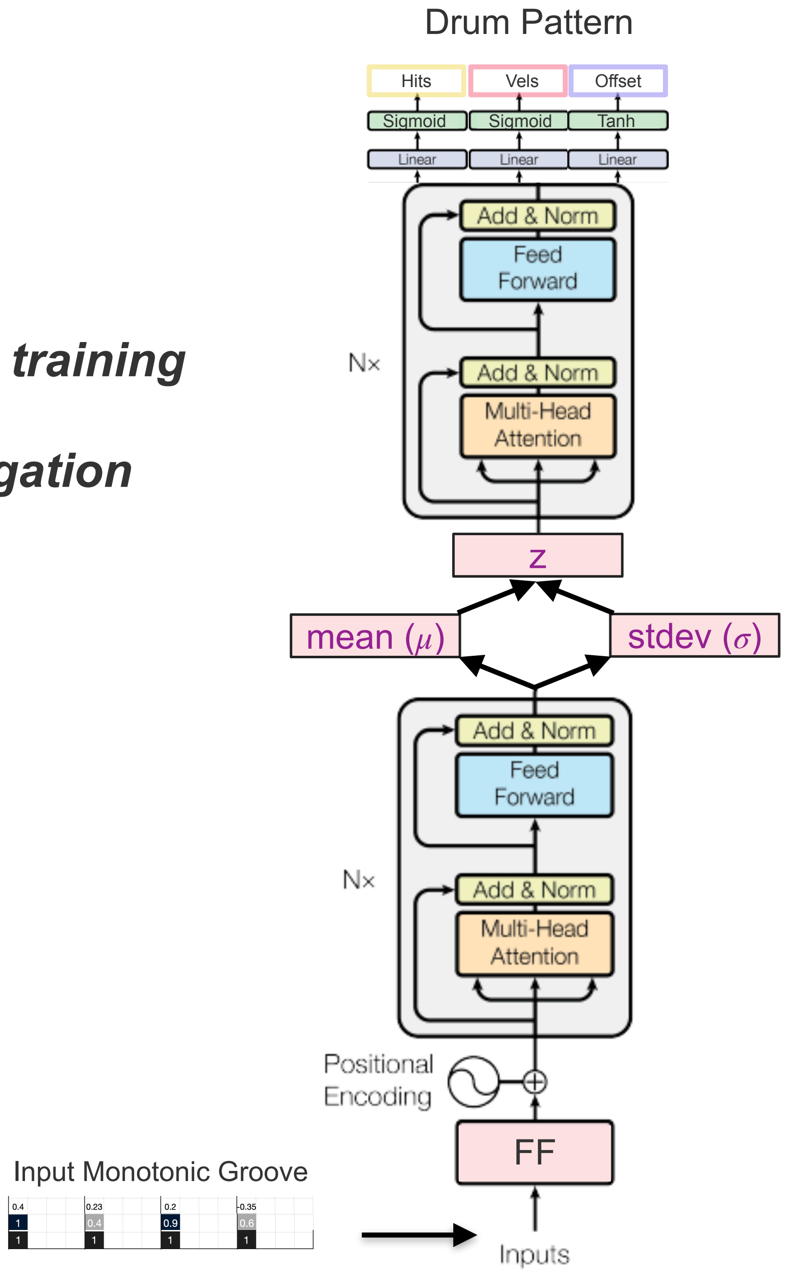
1. Weights of Network
2. Biases of Network

Learnt from training data using back-propagation

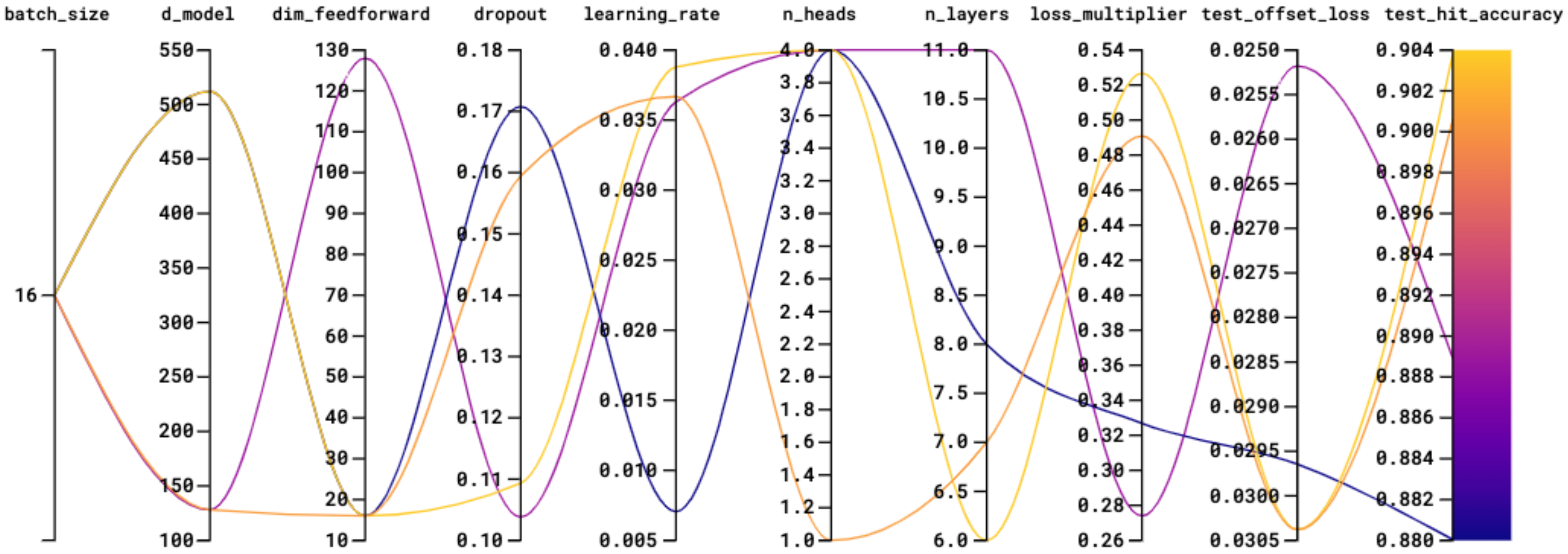
2. Hyper-Parameters (Non-trainable):

1. Number of Enc/Dec layers
2. FF dimensionality
3. Z dimensionality

…



Training



2. Hyper-Parameters (Non-trainable):

1. Number of Enc/Dec layers
2. FF dimensionality
3. Z dimensionality

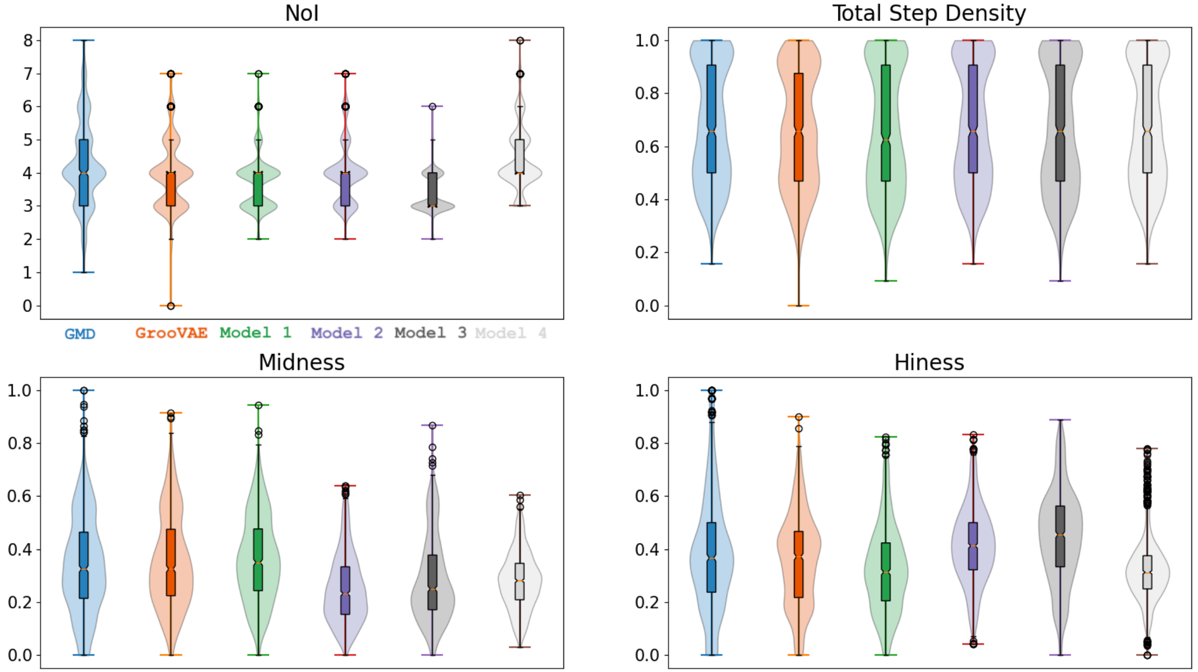
…

Tuned by Trial and Error
(AKA Hyper-parameter Tuning)

Read more:
https://docs.wandb.ai/guides/sweeps

Validation by Global Comparison (Feature-based)

Extract musical features and compare models against one another by comparing the global distribution of features

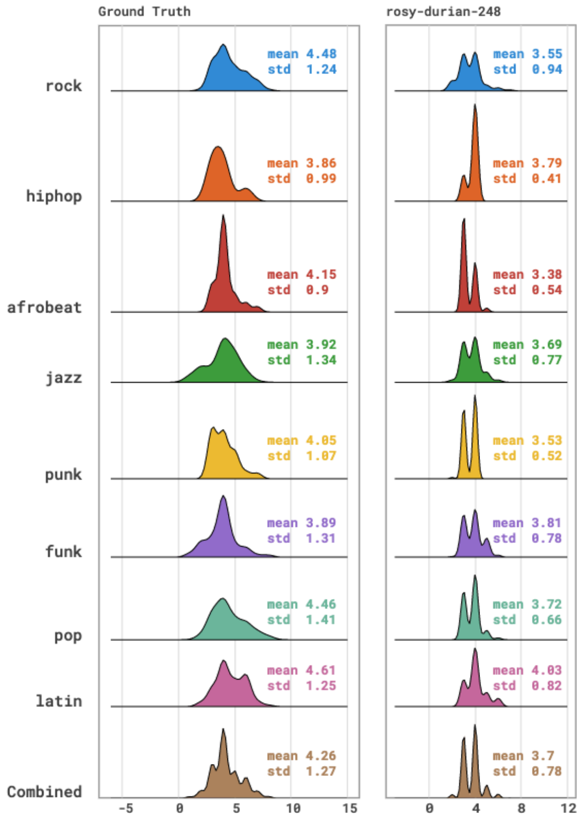


See more here:
https://wandb.ai/anonmmi/AIMC2022/reports/Absolute-Analysis--VmlldzoxOTU2OTc1

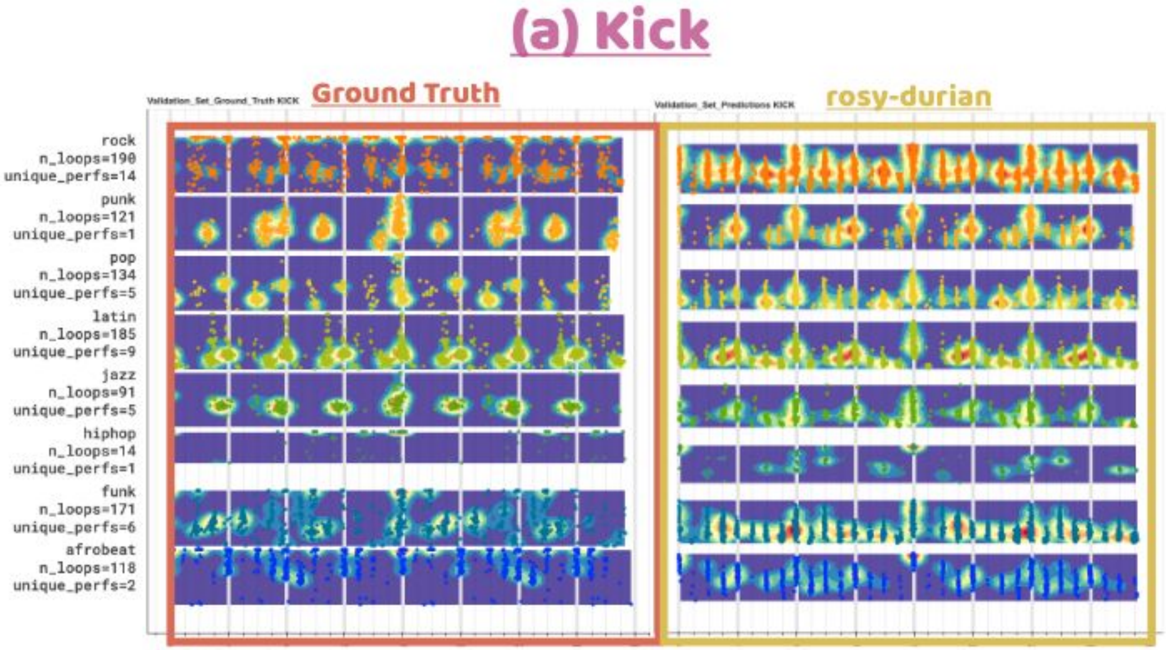
Validation by Global Comparison (Feature-based)

Now do the comparison per genre

Number of Instruments

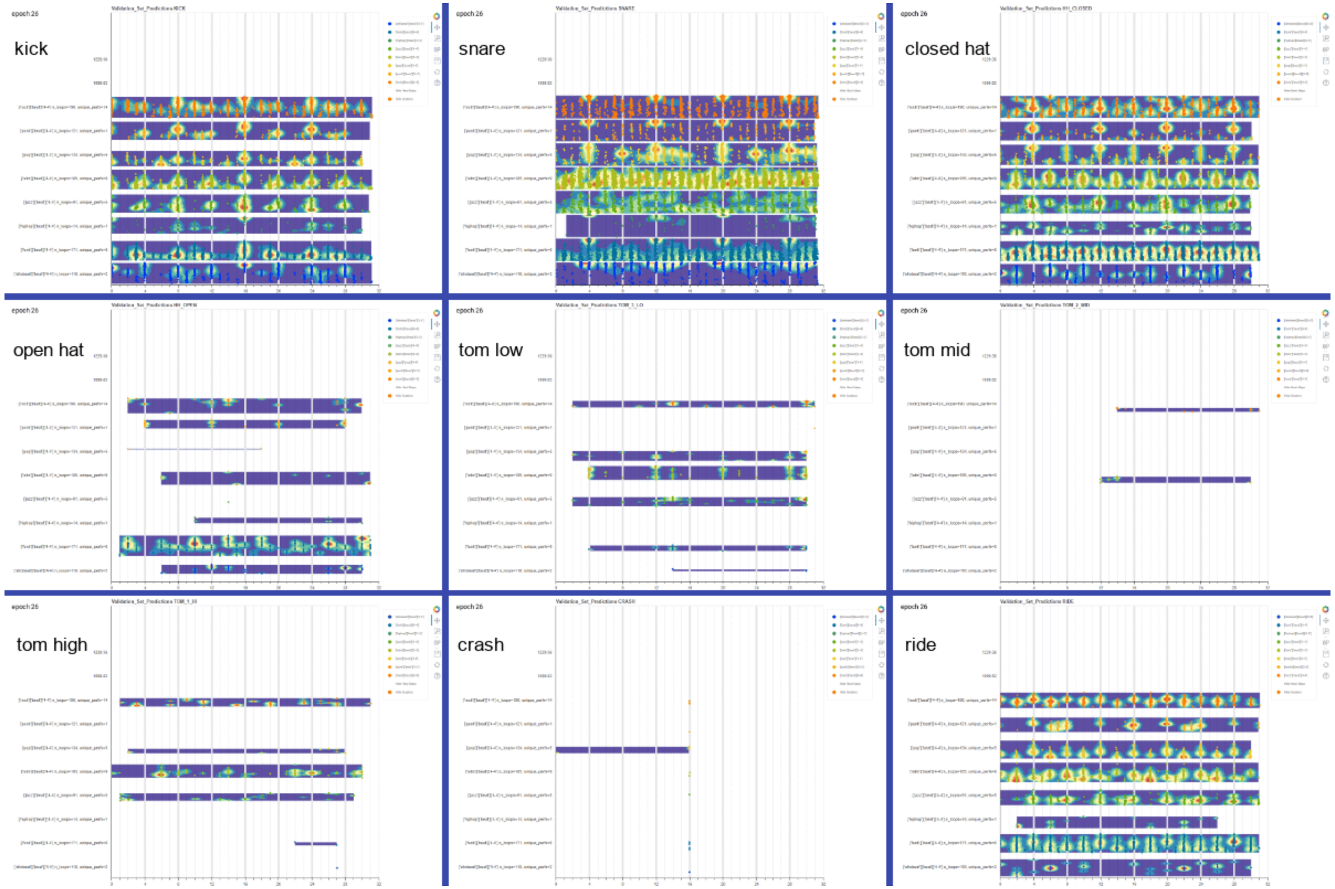


Validation by Global Comparison (Velocity Heat-maps)

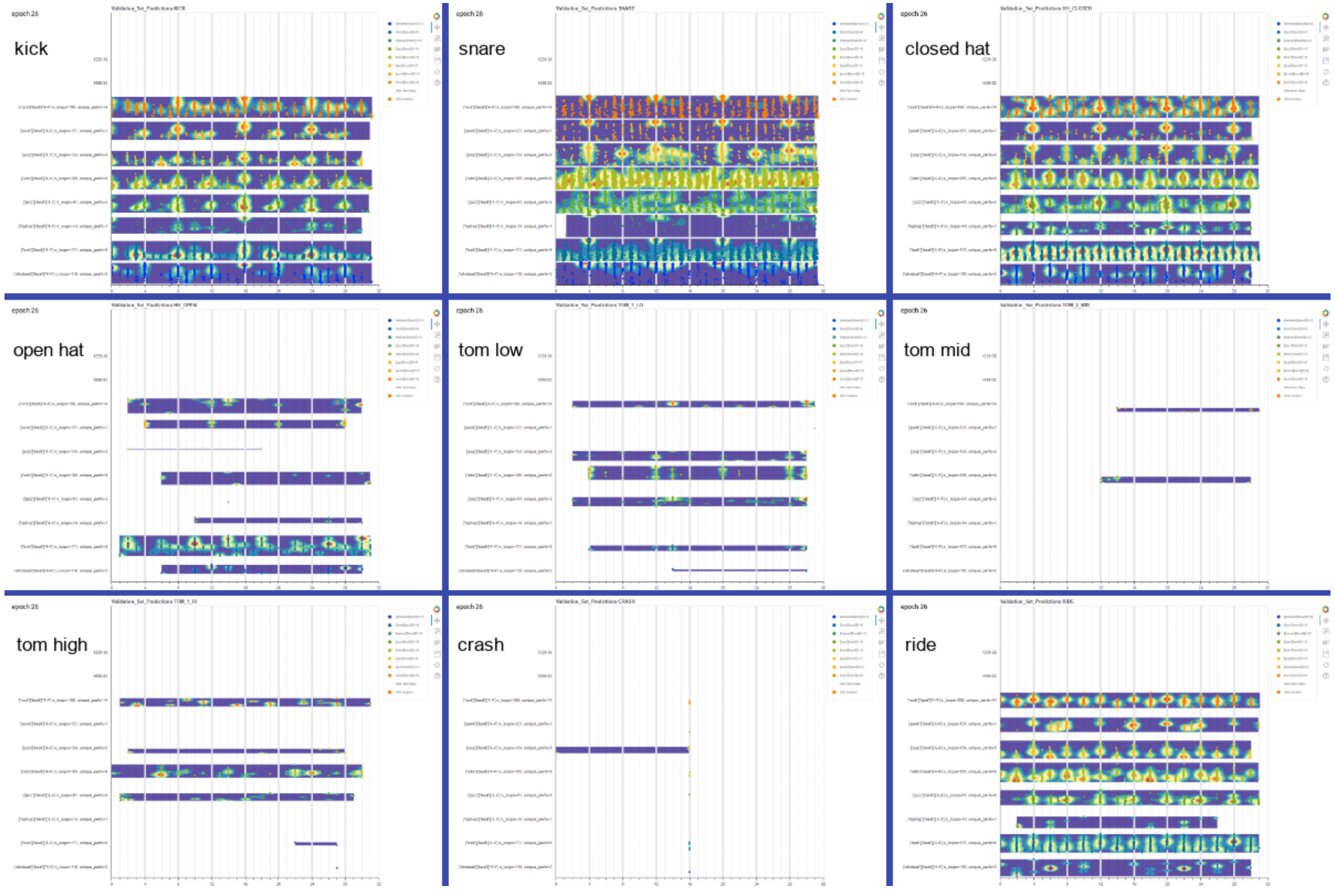


See more here:
https://wandb.ai/anonmmi/AIMC2022/reports/Velocity-heatmaps-for-Run-northern-sweep-26---VmlldzoxNTAxNjMy

Validation by Global Comparison (Velocity Heat-maps)

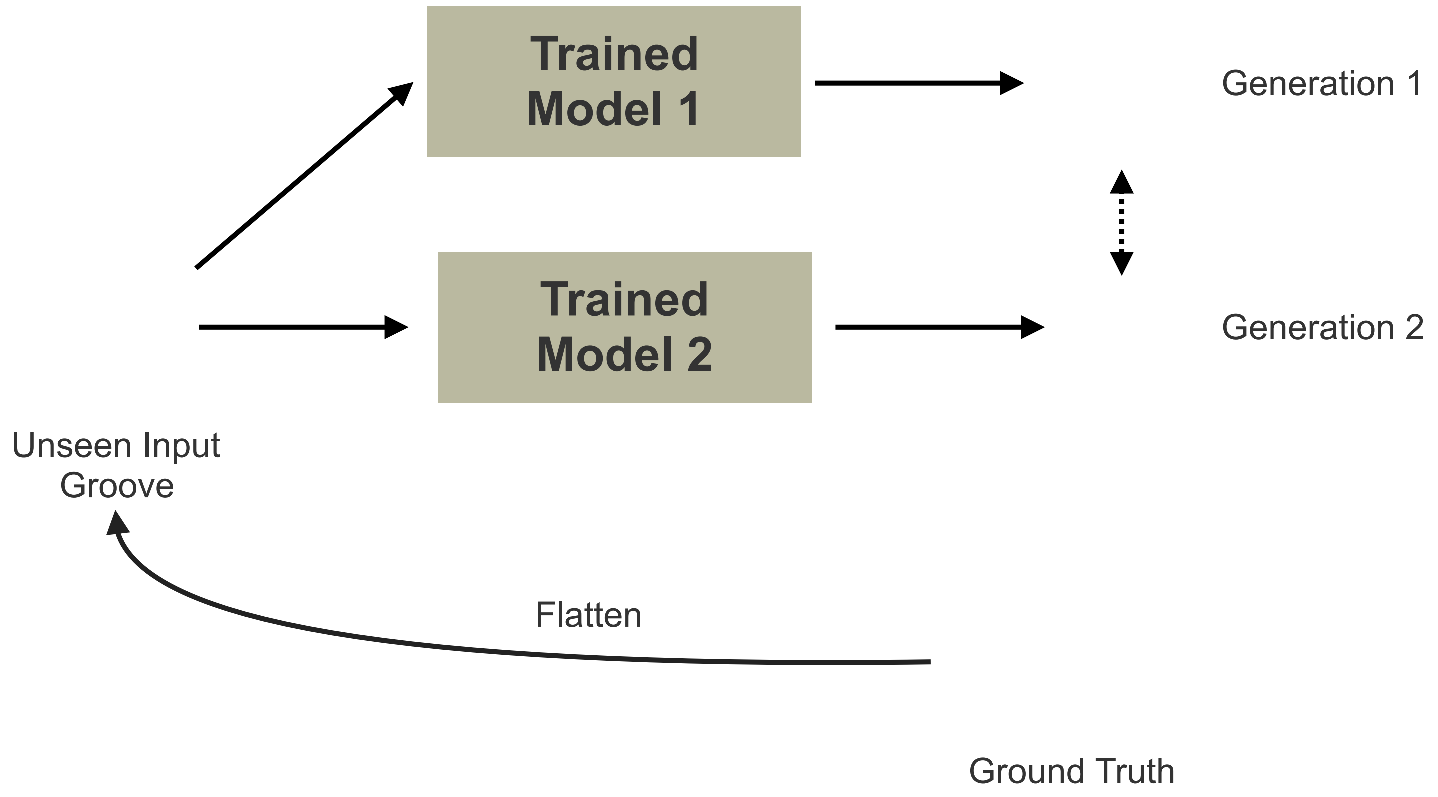


Validation by Global Comparison (Velocity Heat-maps)



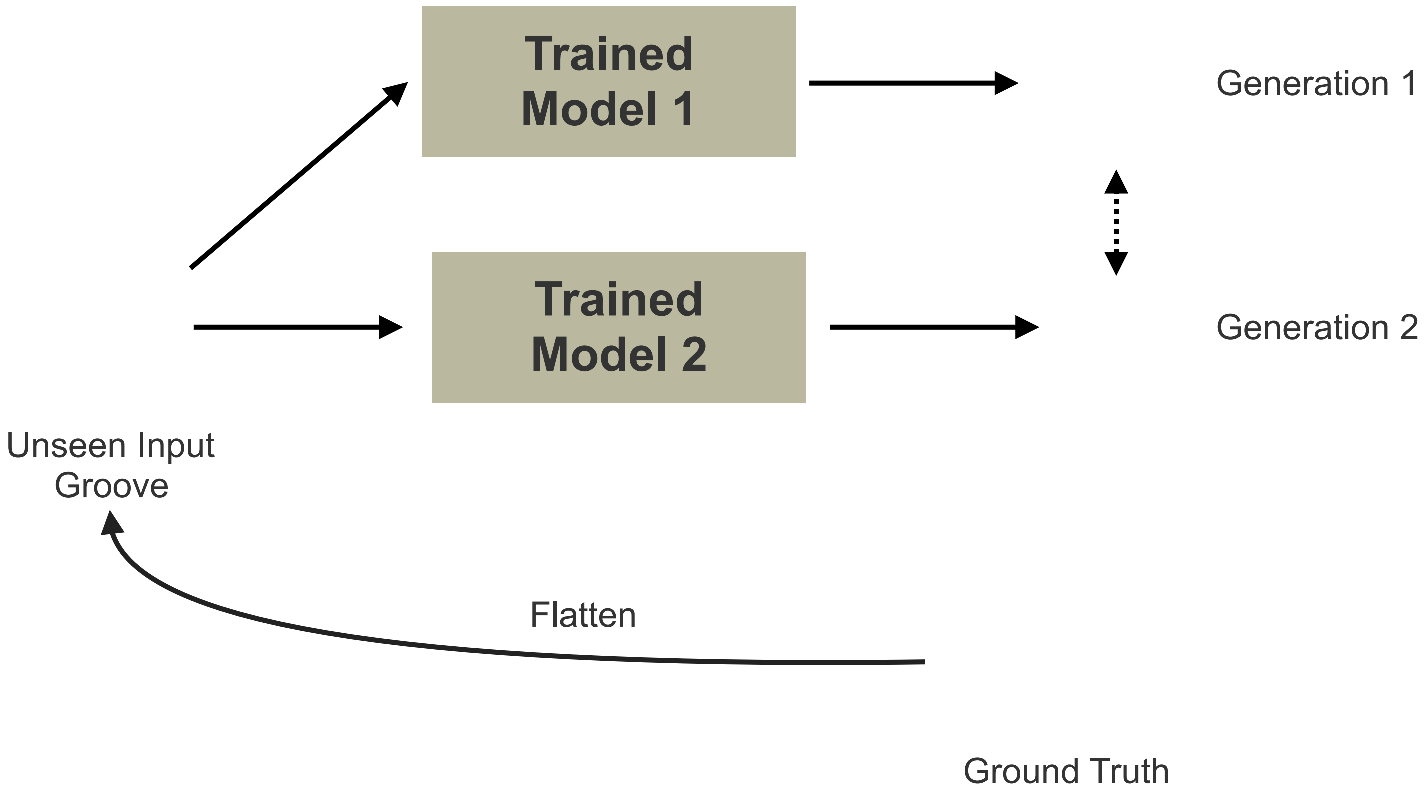
Validation by Listening (Subjective)

Use a held-out portion of data to evaluate the performance



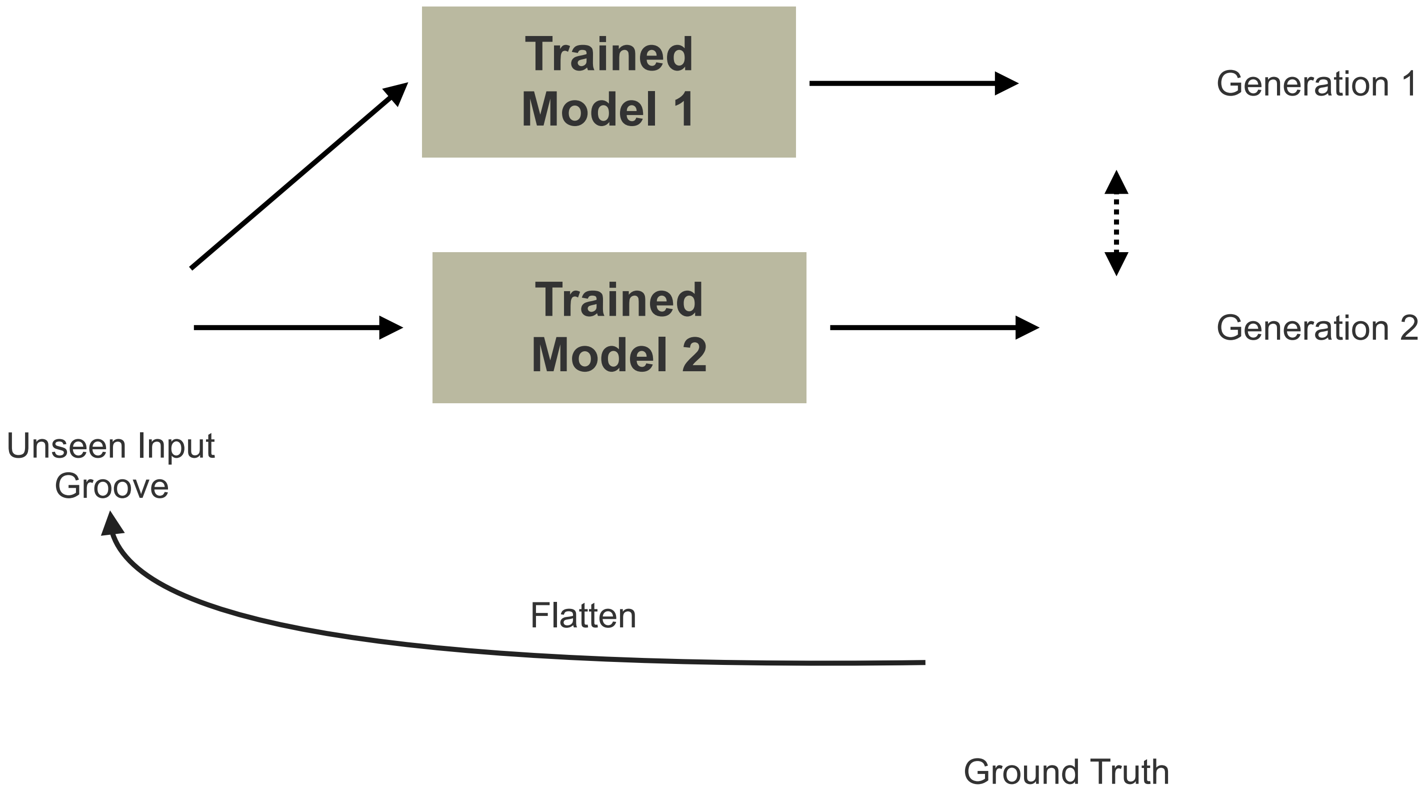
Validation by Listening (Subjective)

Use a held-out portion of data to evaluate the performance



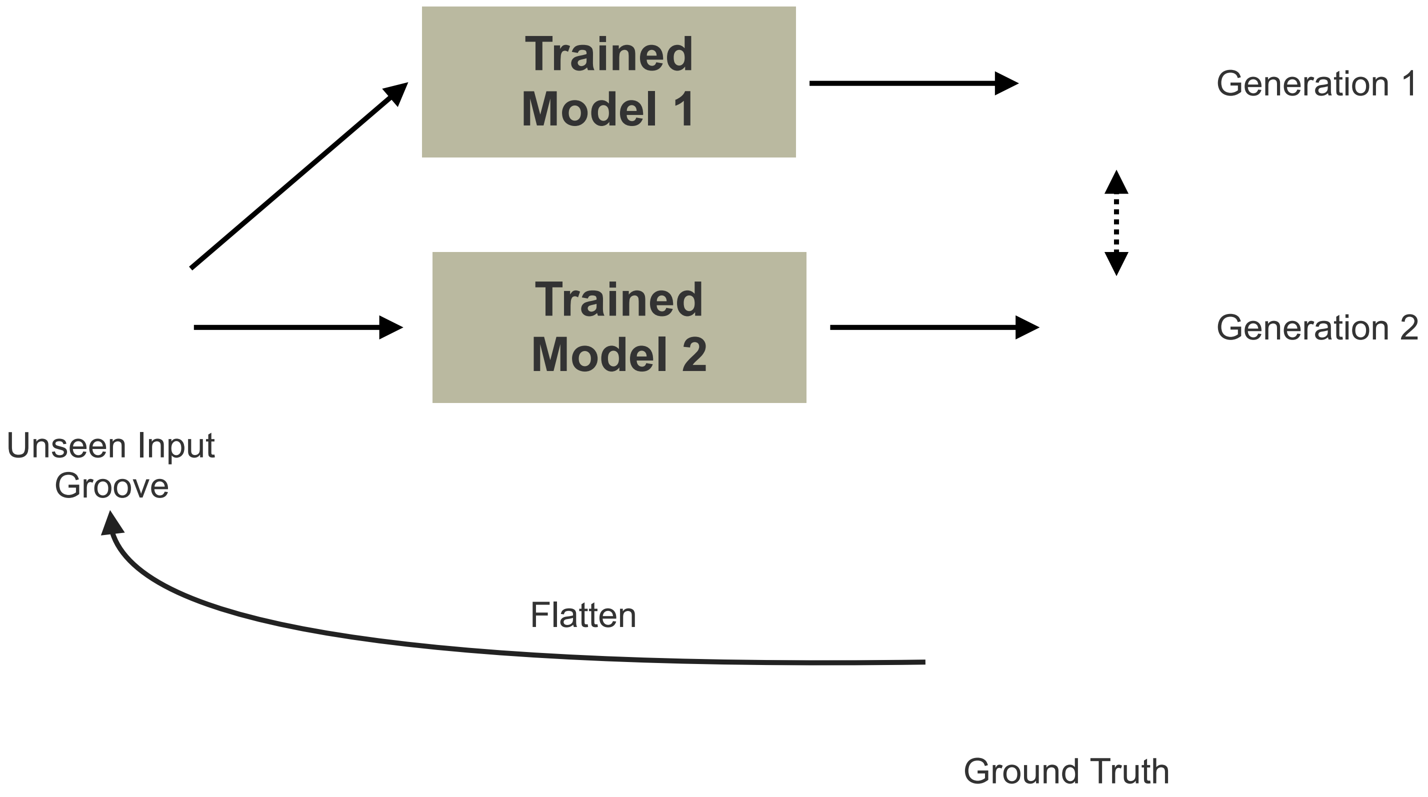
Validation by Listening (Subjective)

Use a held-out portion of data to evaluate the performance



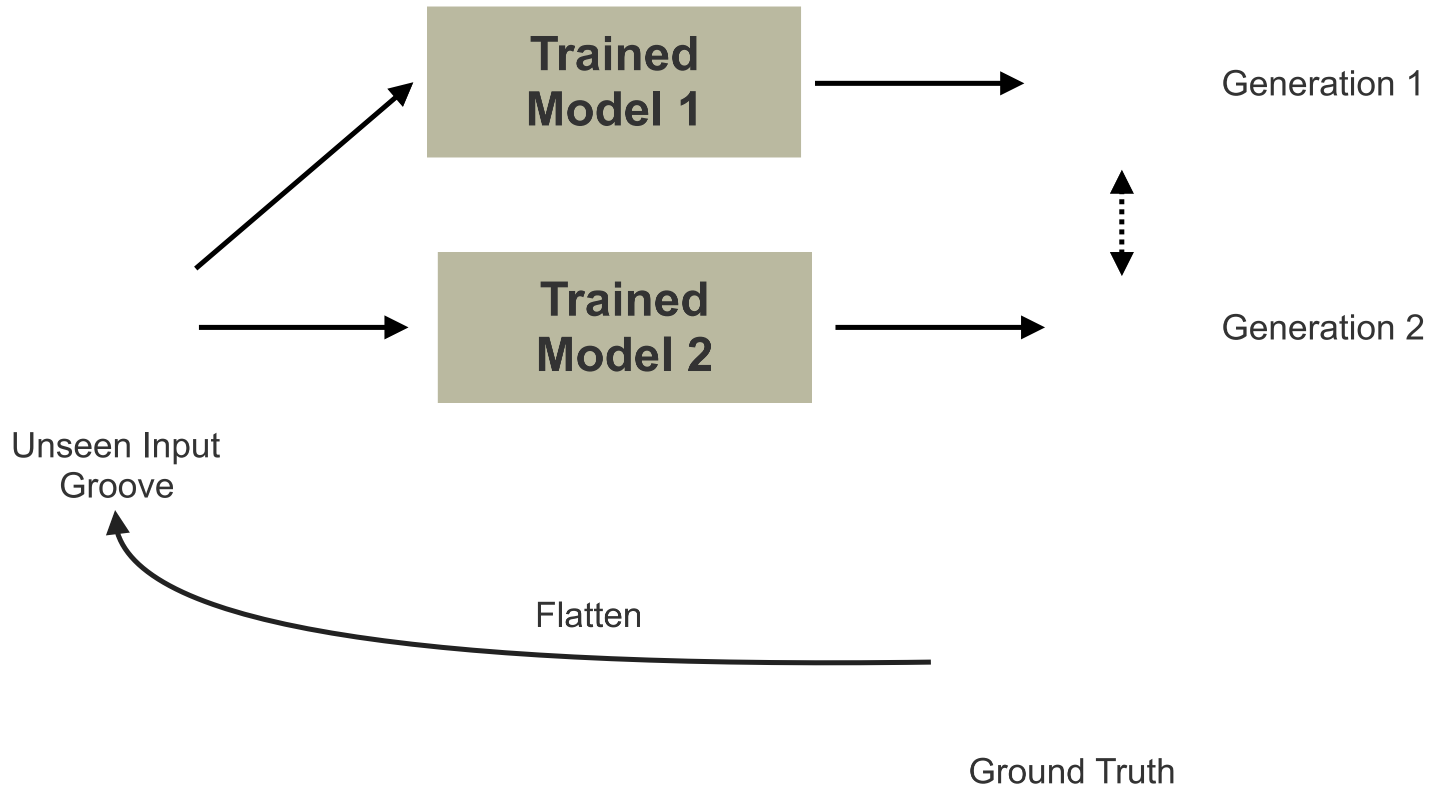
Validation by Listening (Subjective)

Use a held-out portion of data to evaluate the performance



Validation by Listening (Subjective)

Use a held-out portion of data to evaluate the performance



Trained Model 1

Trained Model 2

Generation 1

Generation 2

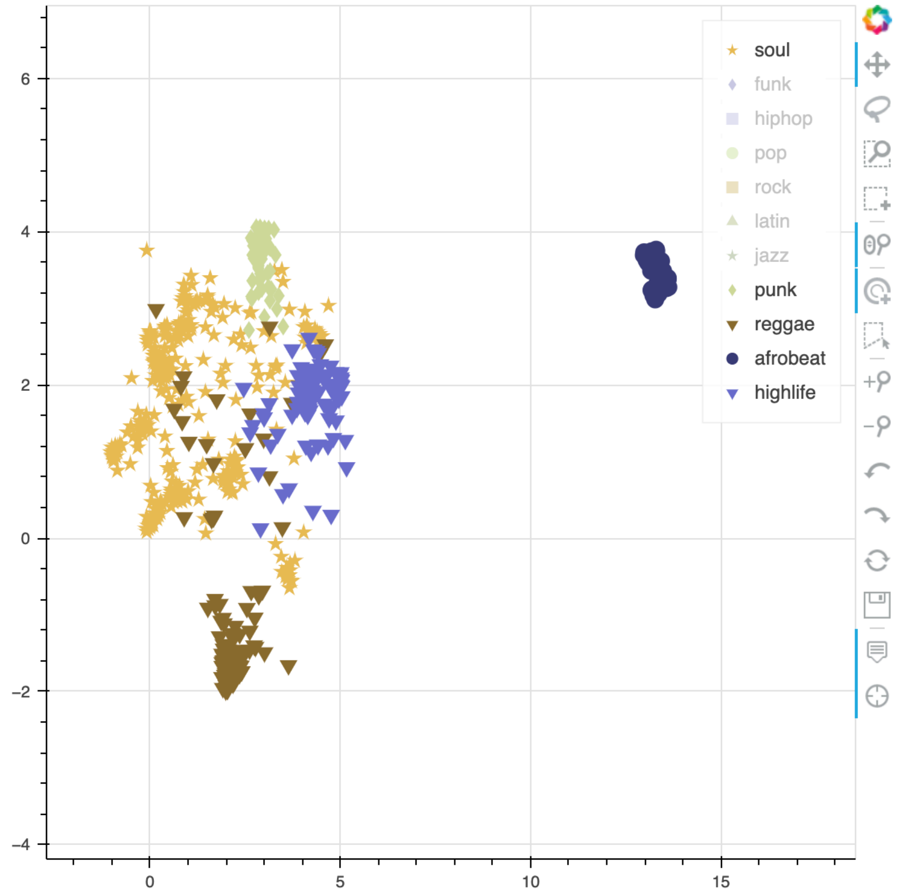
Unseen Input Groove

Flatten

Ground Truth

Validation of Embeddings

Use dimension reduction techniques (such as UMAP, T-SNE) to visualize the clustering of embeddings



Check out in supplementary material:

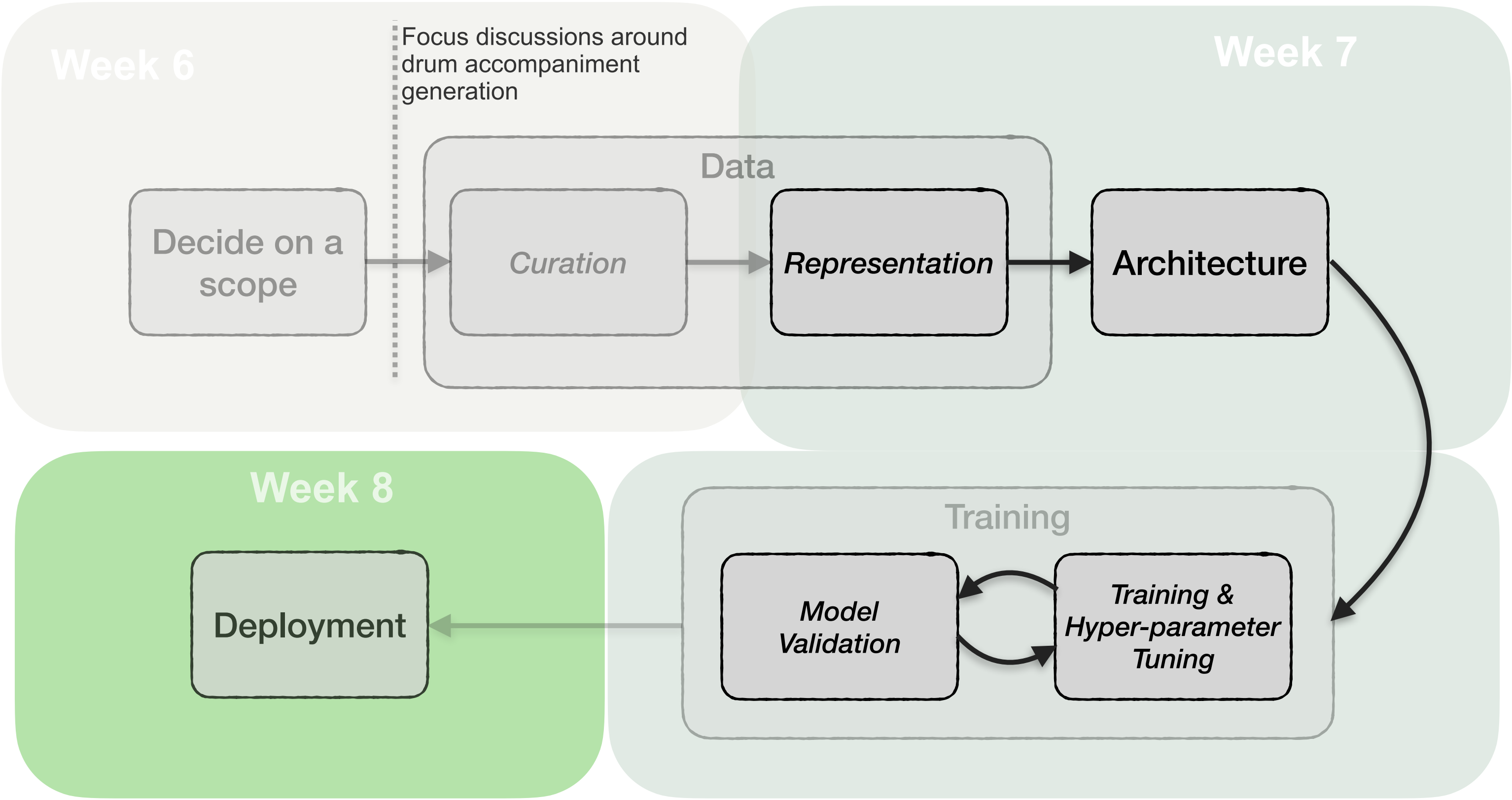
UMAP_drawn_river_6.html
UMAP_noble-field-7.html

Additional Reading

For detailed discussions on topics discussed today:

https://behzadhaki.com/blog/2022/trainingGrooveTransformer/

Focus of this week!



RECAP:

Create a real-time drum generation system that accompanies a given instrumental performance

How can we use the trained model in Real-Time?

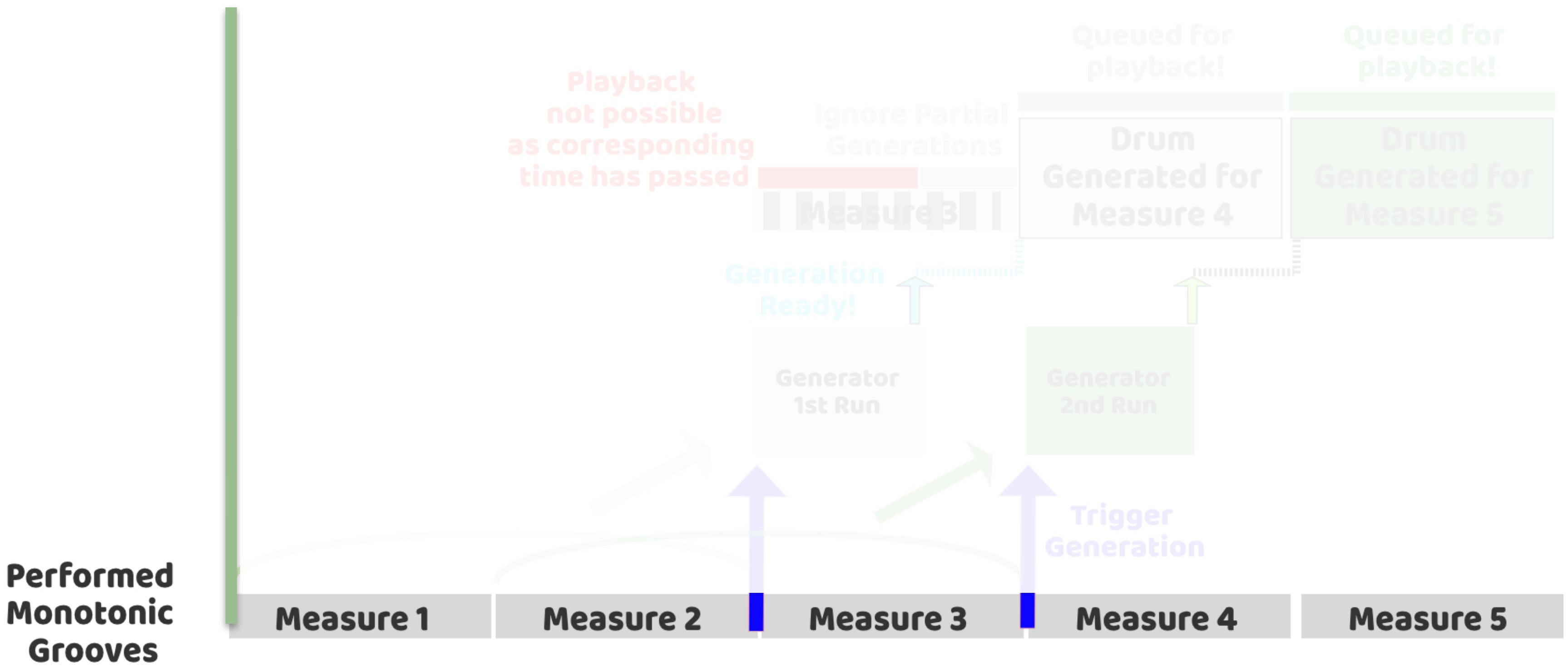
Limitations of the trained model:

- Non-causal Generation (at any time-step, we take into account future events)
- Limited to 2 bars of drums

What if we generate patterns at fixed intervals with a 1-bar delay?

How can we use the trained model in Real-Time?

What if we generate patterns at fixed intervals with a 1-bar delay?



Queued for playback!

Queued for playback!

Playback not possible as corresponding time has passed

Ignore Partial Generations

Measure 3

Drum Generated for Measure 4

Drum Generated for Measure 5

Generation Ready!

Generator 1st Run

Generator 2nd Run

Trigger Generation

Performed Monotonic Grooves

Measure 1

Measure 2

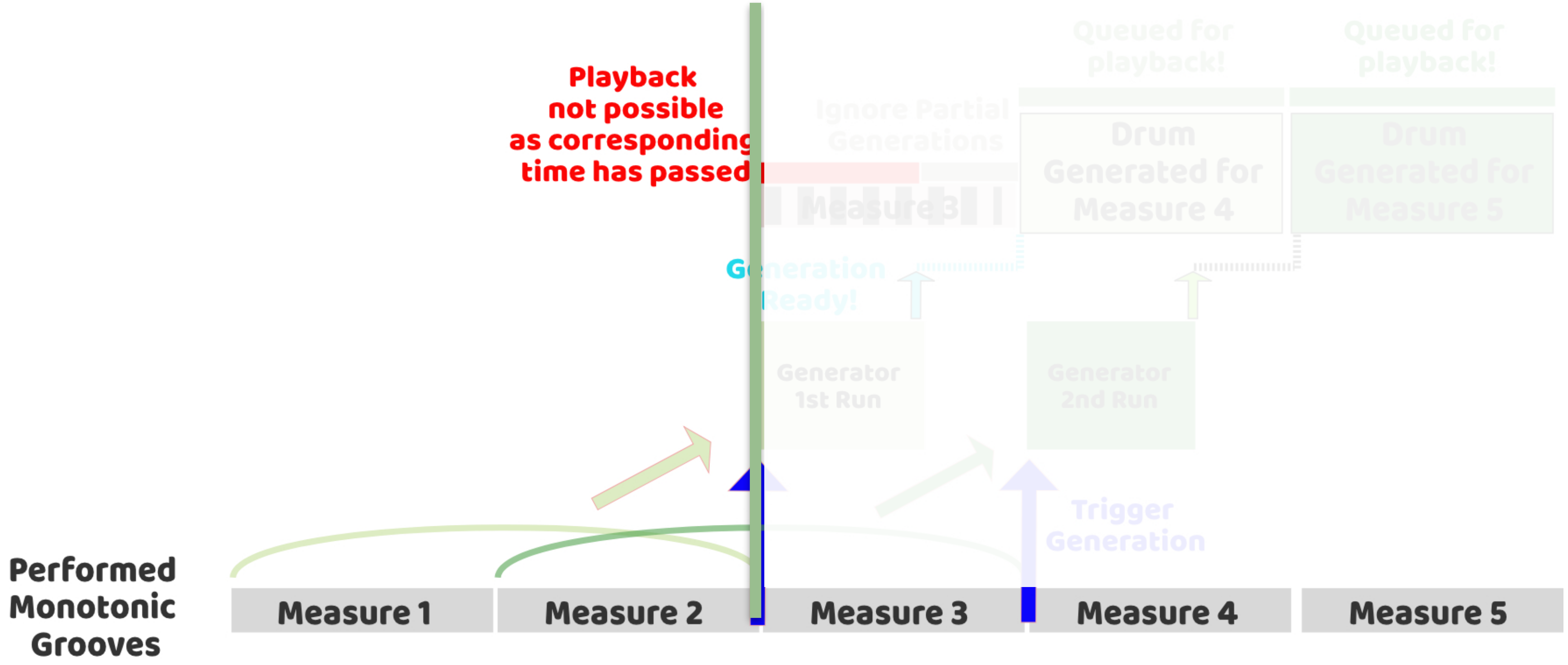
Measure 3

Measure 4

Measure 5

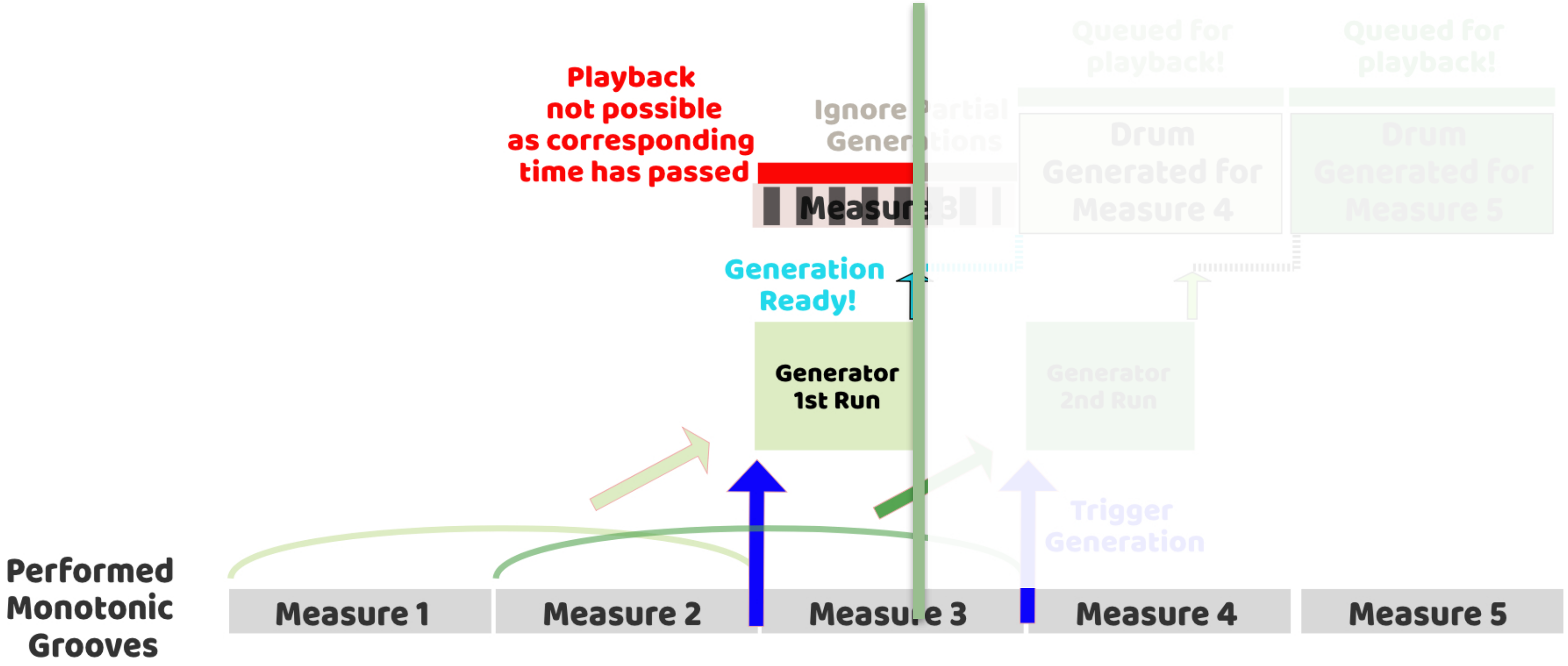
How can we use the trained model in Real-Time?

What if we generate patterns at fixed intervals with a 1-bar delay?


Queued for playback!
Queued for playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Drum Generated For Measure 4
Drum Generated For Measure 5
Measure 3
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

How can we use the trained model in Real-Time?

What if we generate patterns at fixed intervals with a 1-bar delay?



Queued for playback!

Queued for playback!

Playback not possible as corresponding time has passed

Ignore Partial Generations

Drum Generated For Measure 4

Drum Generated For Measure 5

Measure 3

Generation Ready!

Generator 1st Run

Generator 2nd Run

Trigger Generation

Performed Monotonic Grooves

Measure 1

Measure 2

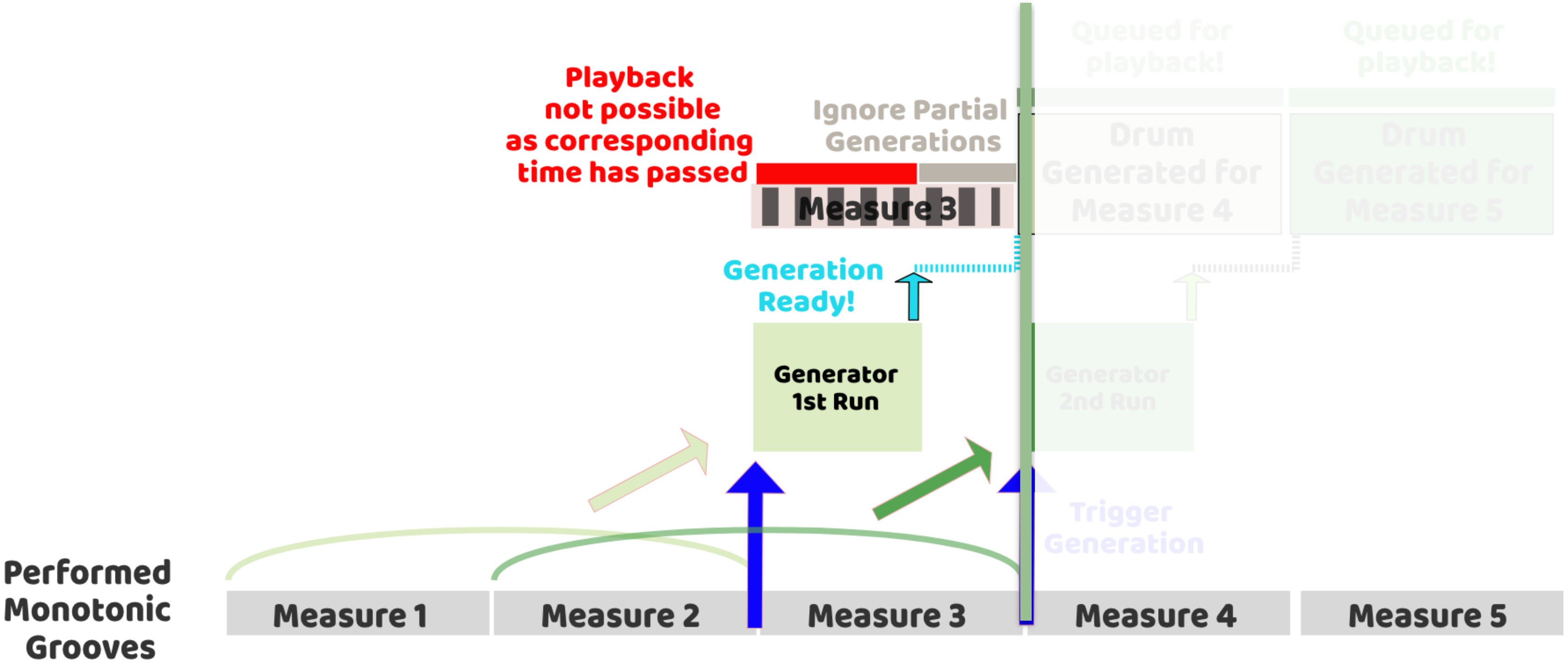
Measure 3

Measure 4

Measure 5

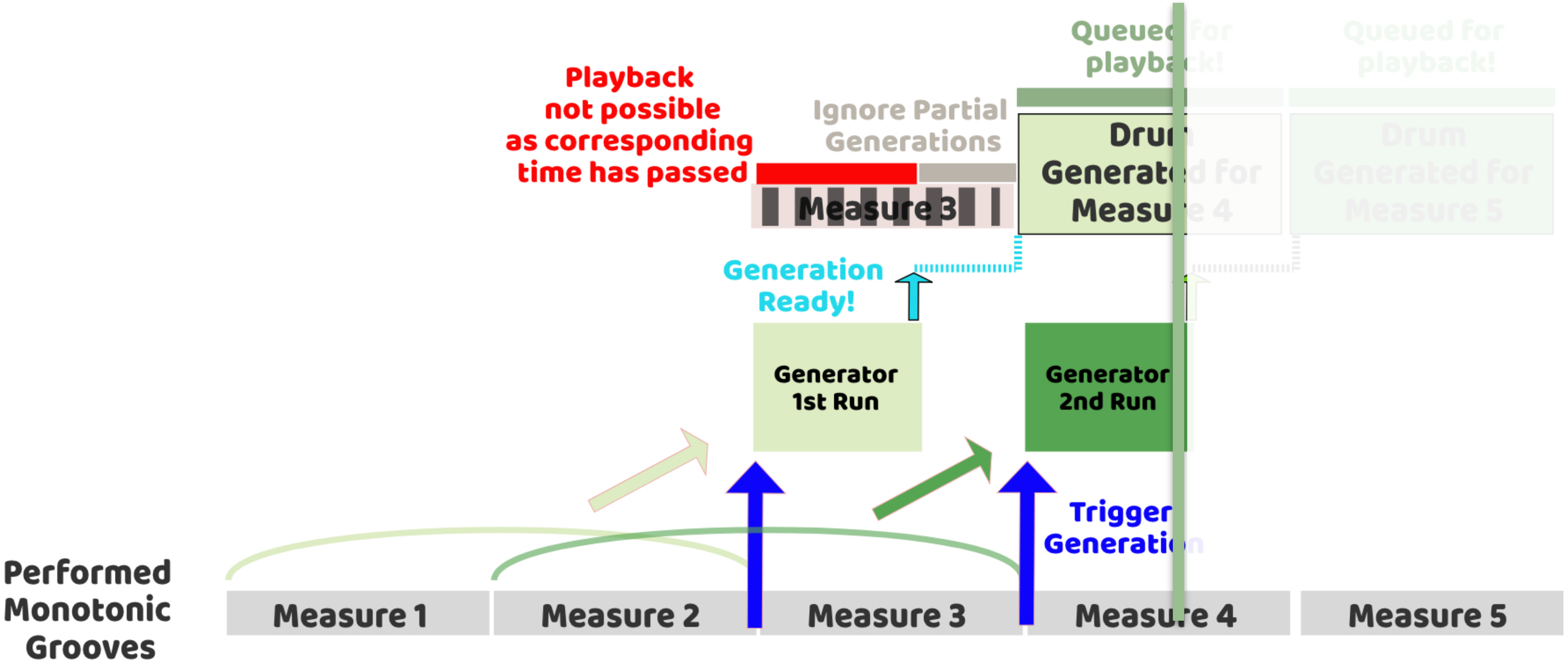
How can we use the trained model in Real-Time?

What if we generate patterns at fixed intervals with a 1-bar delay?


Queued for playback!
Queued For playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Measure 3
Drum Generated For Measure 4
Drum Generated For Measure 5
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

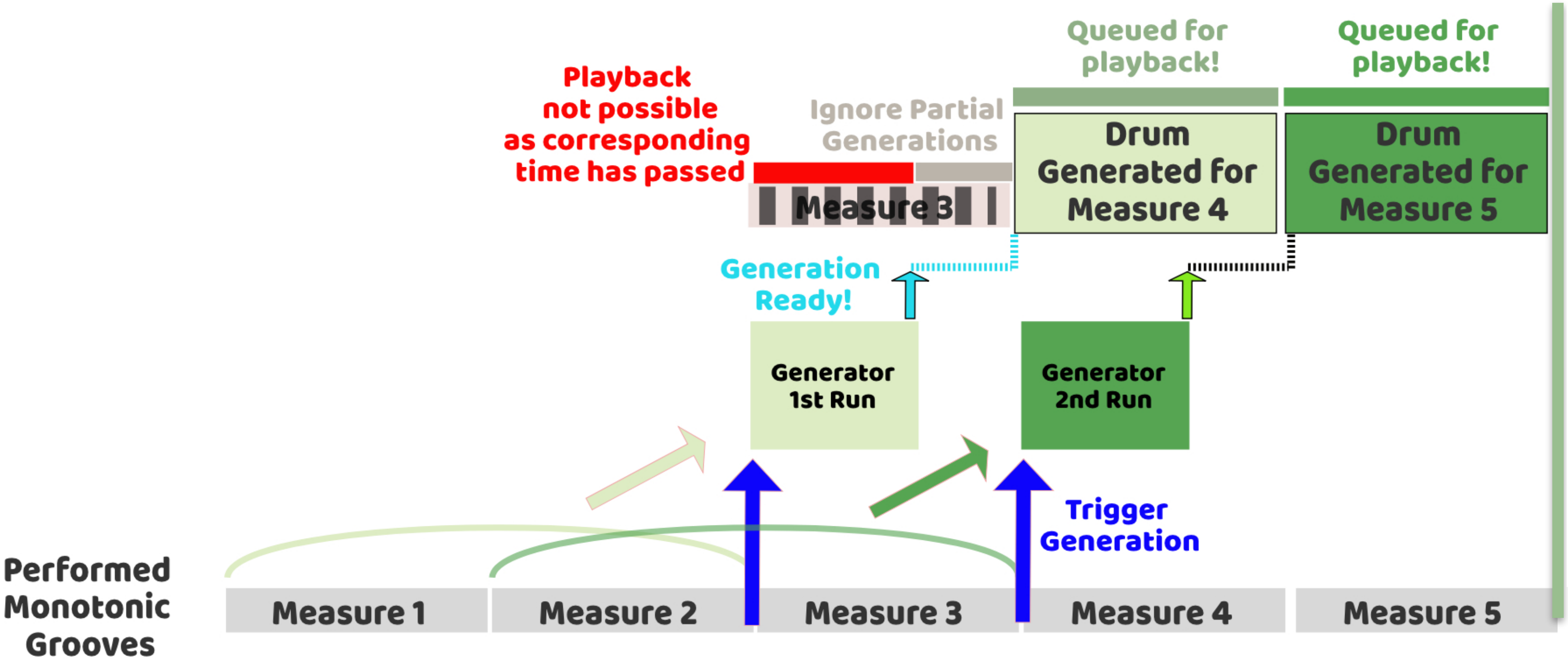
How can we use the trained model in Real-Time?

What if we generate patterns at fixed intervals with a 1-bar delay?

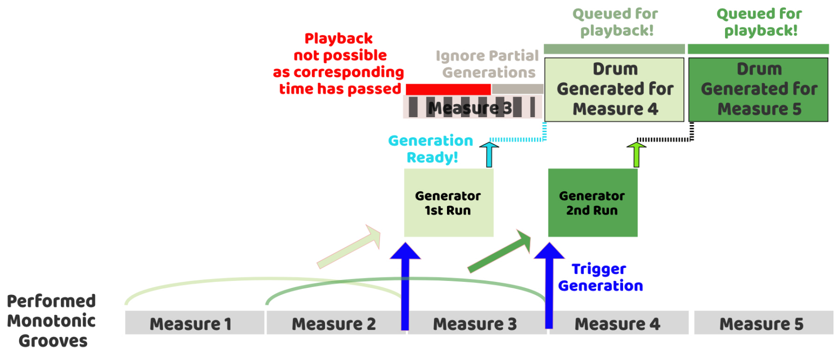

Queued for playback!
Queued For playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Drum Generated For Measure 4
Drum Generated For Measure 5
Measure 3
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

How can we use the trained model in Real-Time?

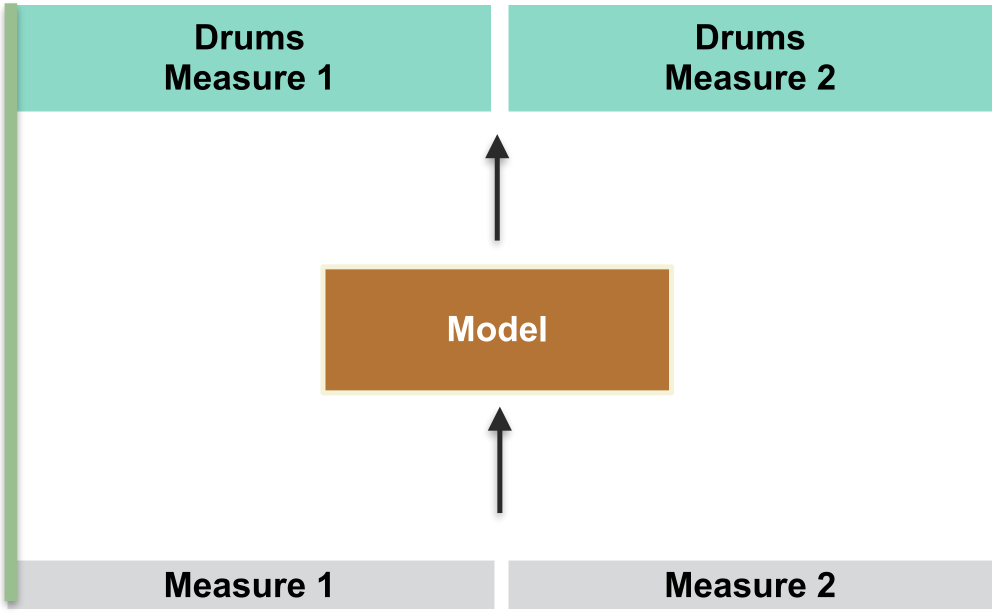
What if we generate patterns at fixed intervals with a 1-bar delay?


Queued for playback!
Queued for playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Measure 3
Drum Generated for Measure 4
Drum Generated for Measure 5
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

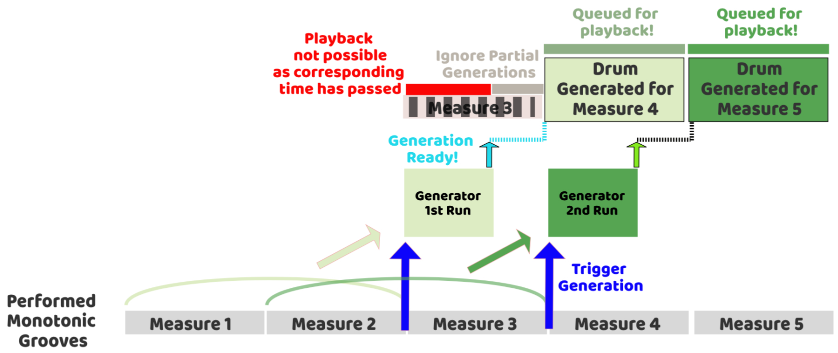
How can we use the trained model in Real-Time?


Queued for playback!
Queued for playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Drum Generated for Measure 4
Drum Generated for Measure 5
Measure 3
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

What if we think of the performance in a looped manner?



How can we use the trained model in Real-Time?



Queued for playback!

Queued for playback!

Playback not possible as corresponding time has passed

Ignore Partial Generations

Drum Generated for Measure 4

Drum Generated for Measure 5

Measure 3

Generation Ready!

Generator 1st Run

Generator 2nd Run

Trigger Generation

Performed Monotonic Grooves

Measure 1

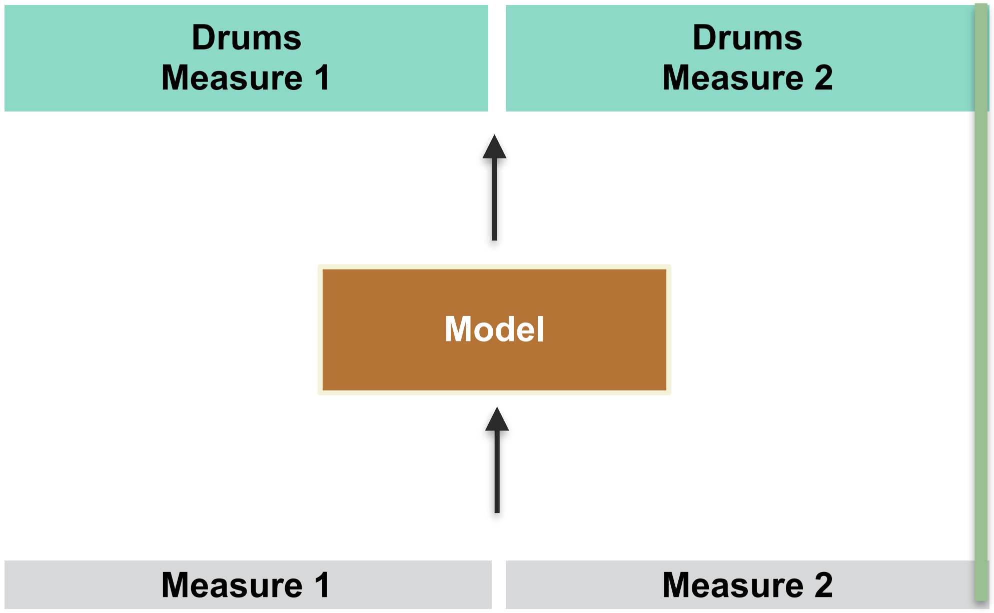
Measure 2

Measure 3

Measure 4

Measure 5

What if we think of the performance in a looped manner?



Drums Measure 1

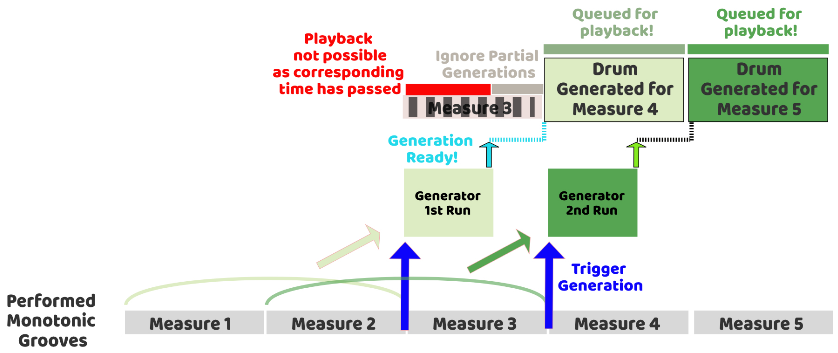
Drums Measure 2

Model

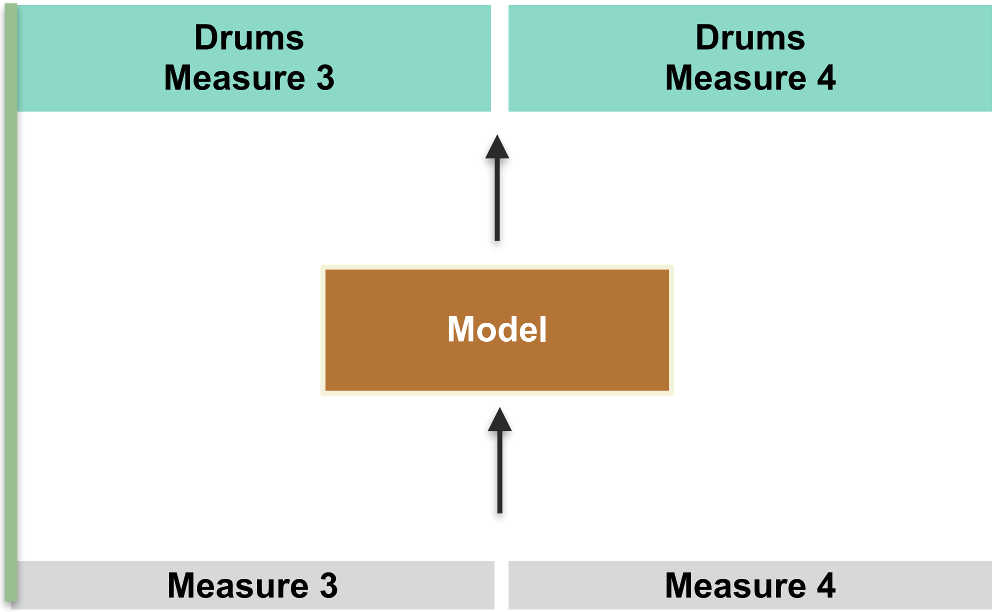
Measure 1

Measure 2

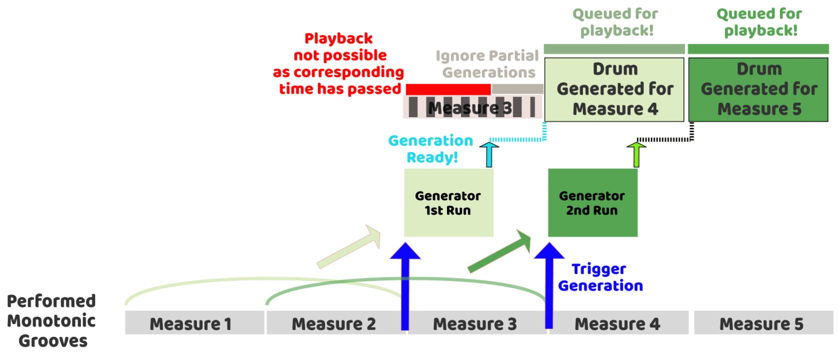
How can we use the trained model in Real-Time?


Queued for playback!
Queued for playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Drum Generated for Measure 4
Drum Generated for Measure 5
Measure 3
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

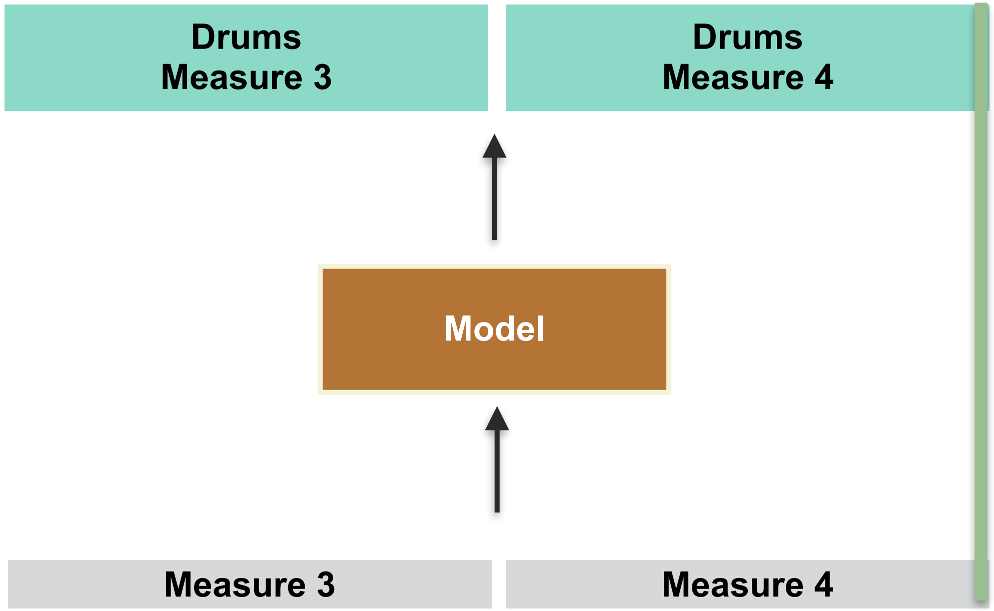
What if we think of the performance in a looped manner?



How can we use the trained model in Real-Time?


Queued for playback!
Queued for playback!
Playback not possible as corresponding time has passed
Ignore Partial Generations
Measure 3
Drum Generated for Measure 4
Drum Generated for Measure 5
Generation Ready!
Generator 1st Run
Generator 2nd Run
Trigger Generation
Performed Monotonic Grooves
Measure 1
Measure 2
Measure 3
Measure 4
Measure 5

What if we think of the performance in a looped manner?

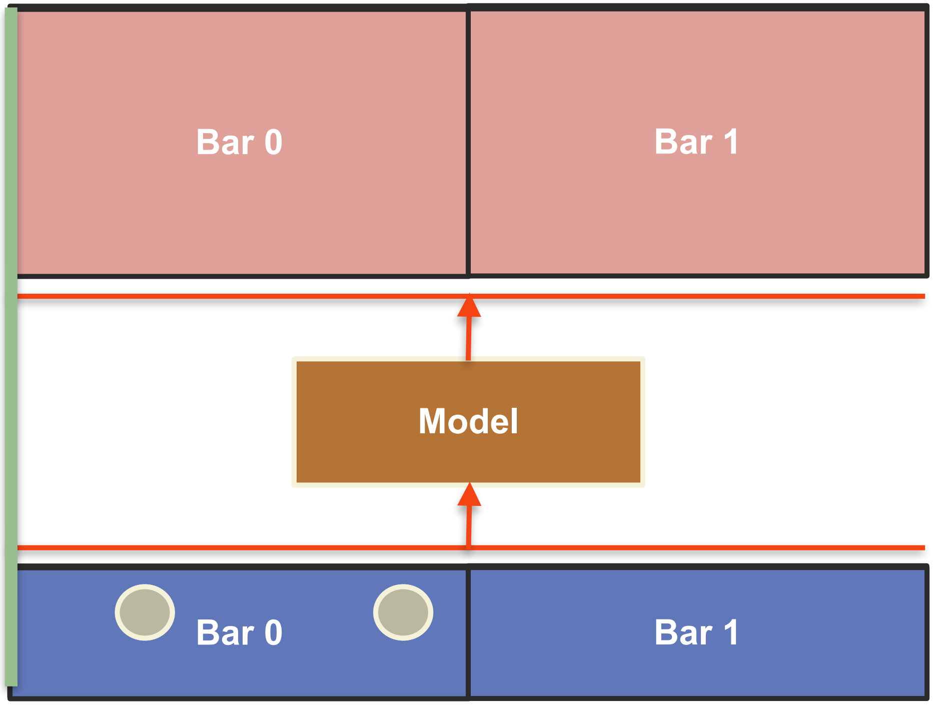


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

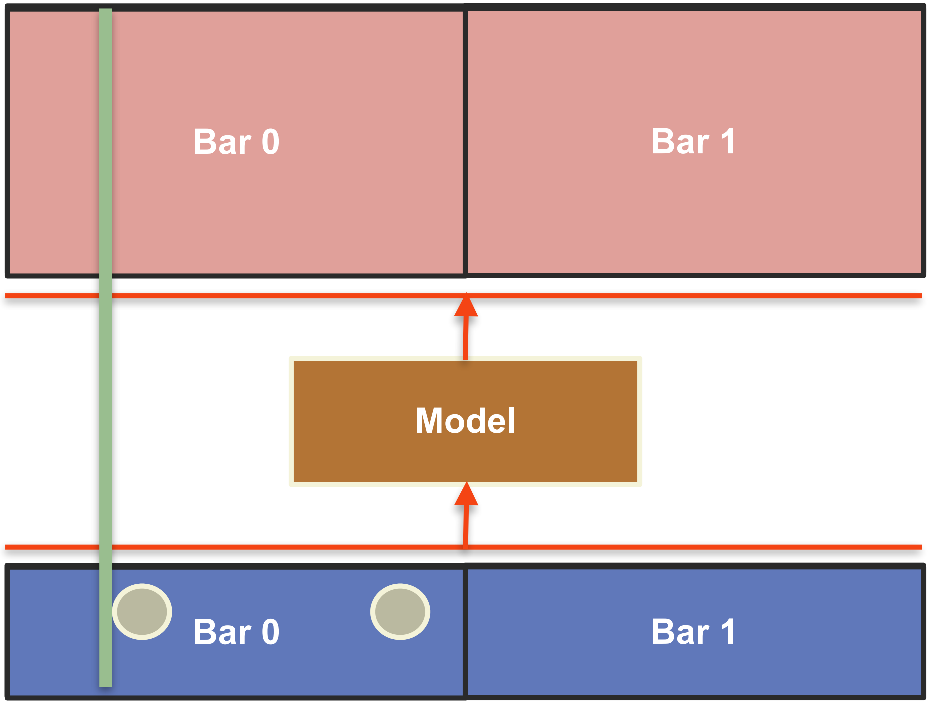


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

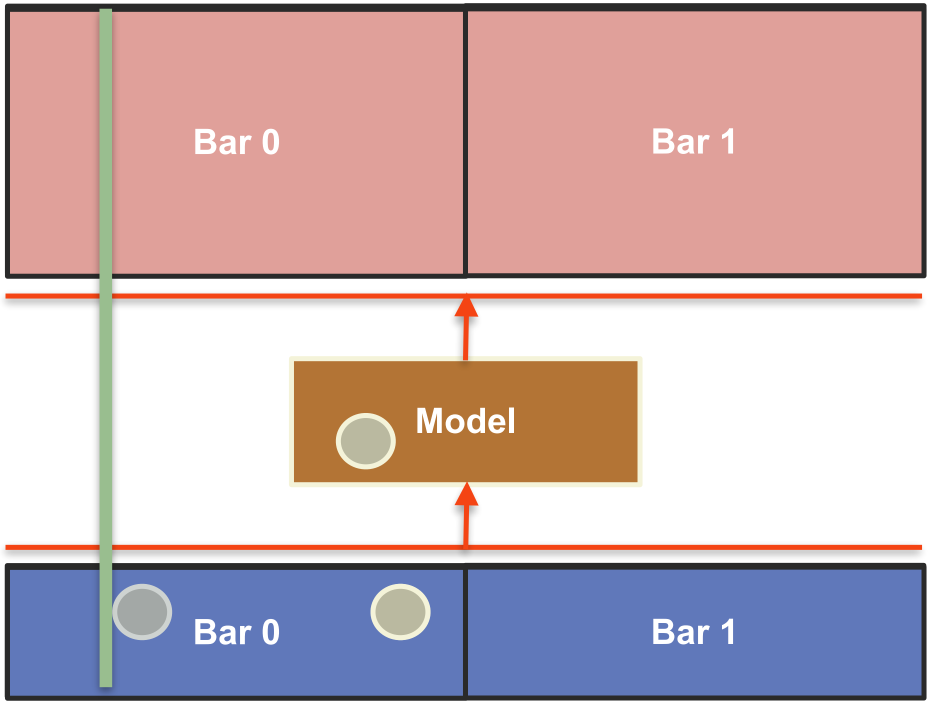


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

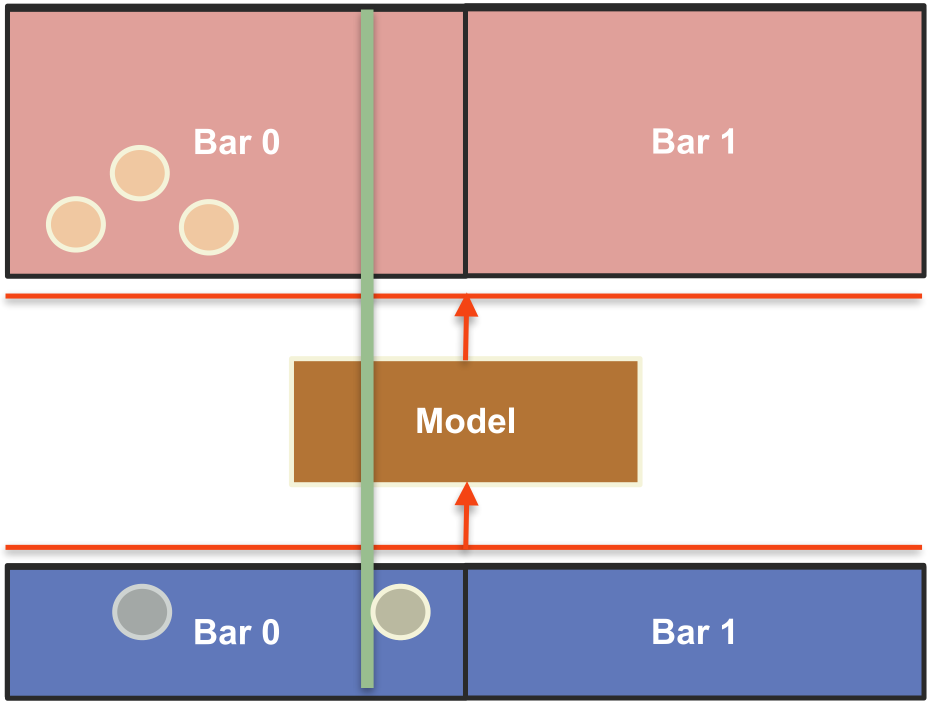


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

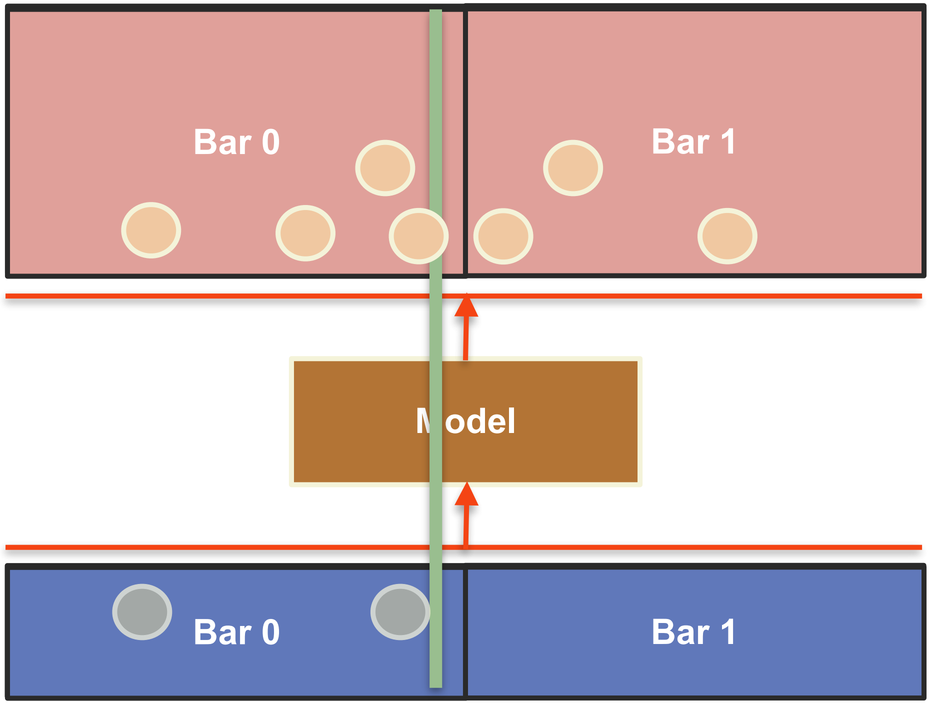


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

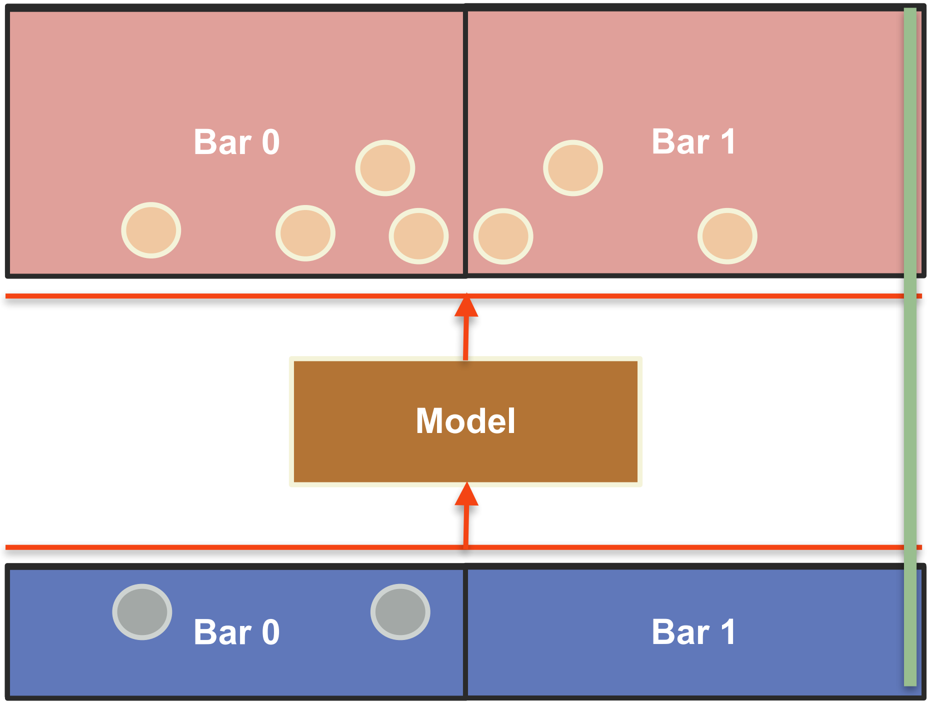


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

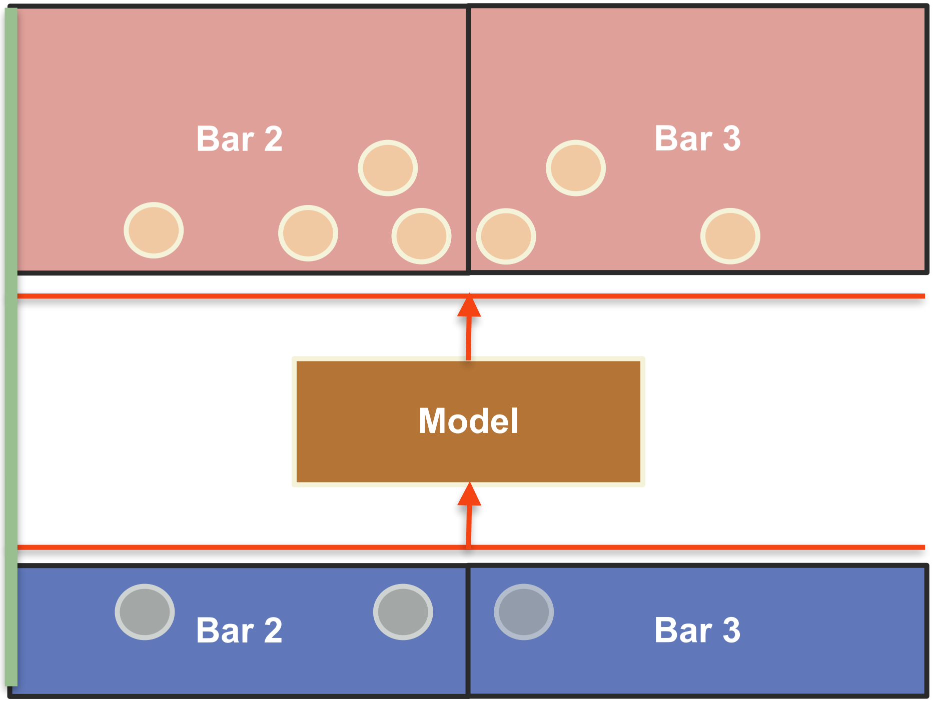


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

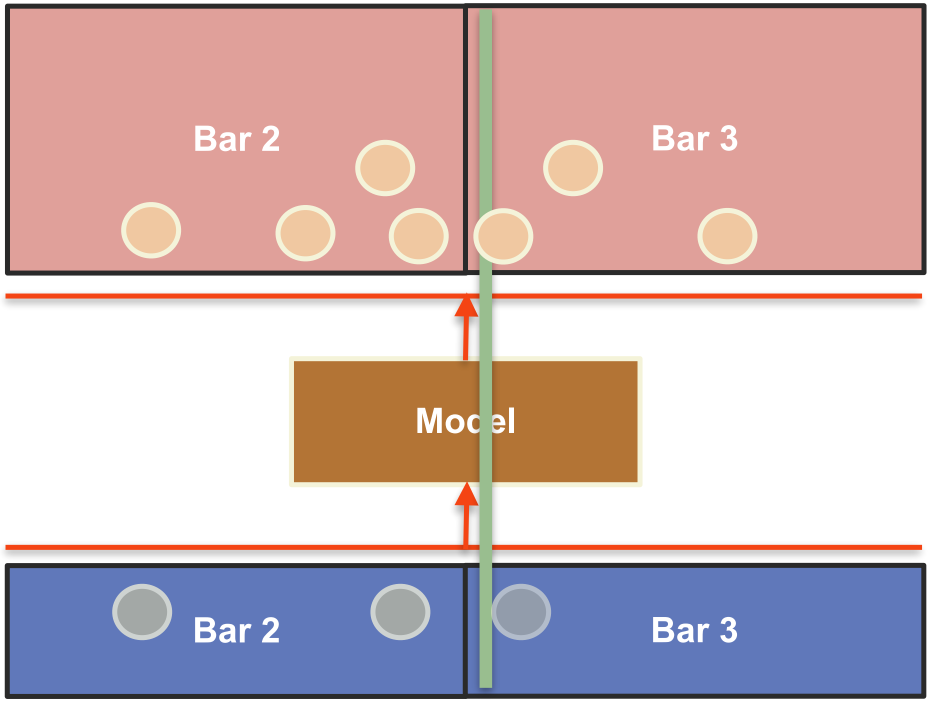


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting

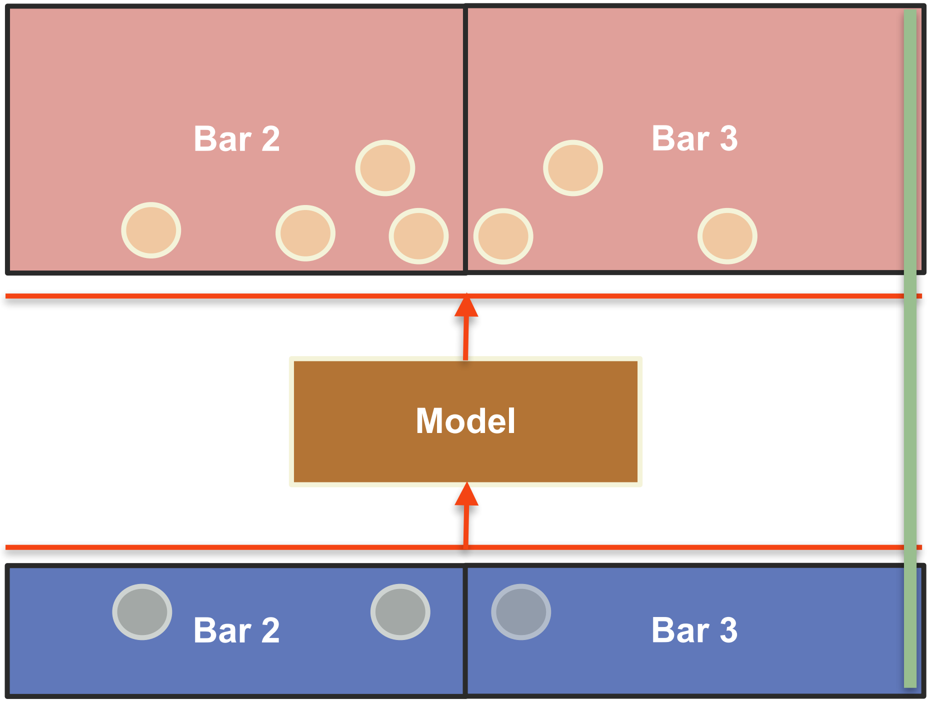


How can we use the trained model in Real-Time?

Is inference fast enough for real-time deployment?

- < 10ms (running in python)

Do we need any delay compensation? —-> Not in this looping setting



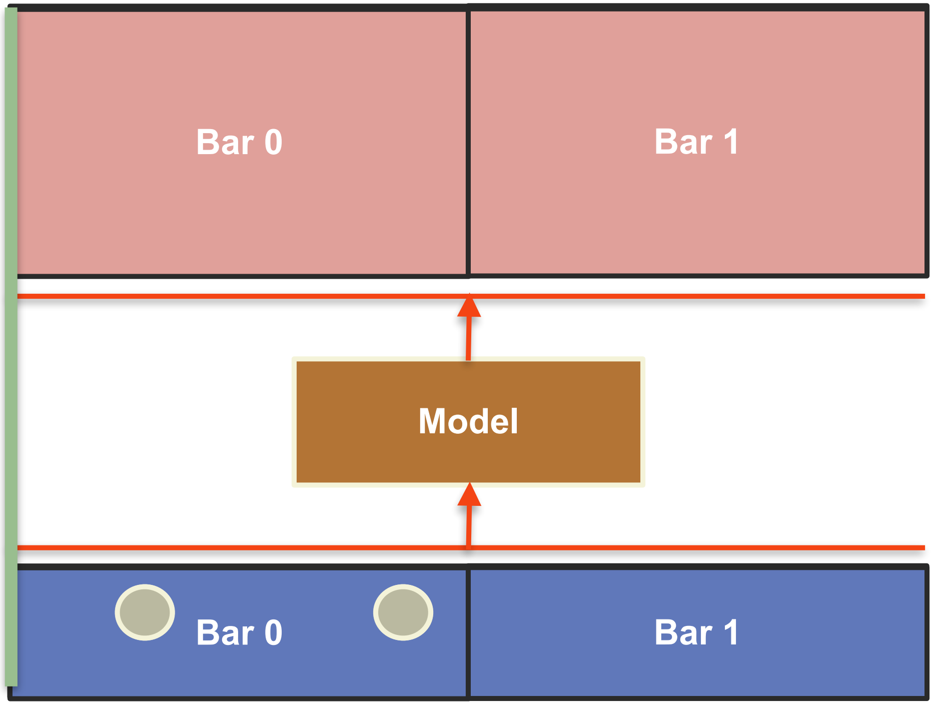
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



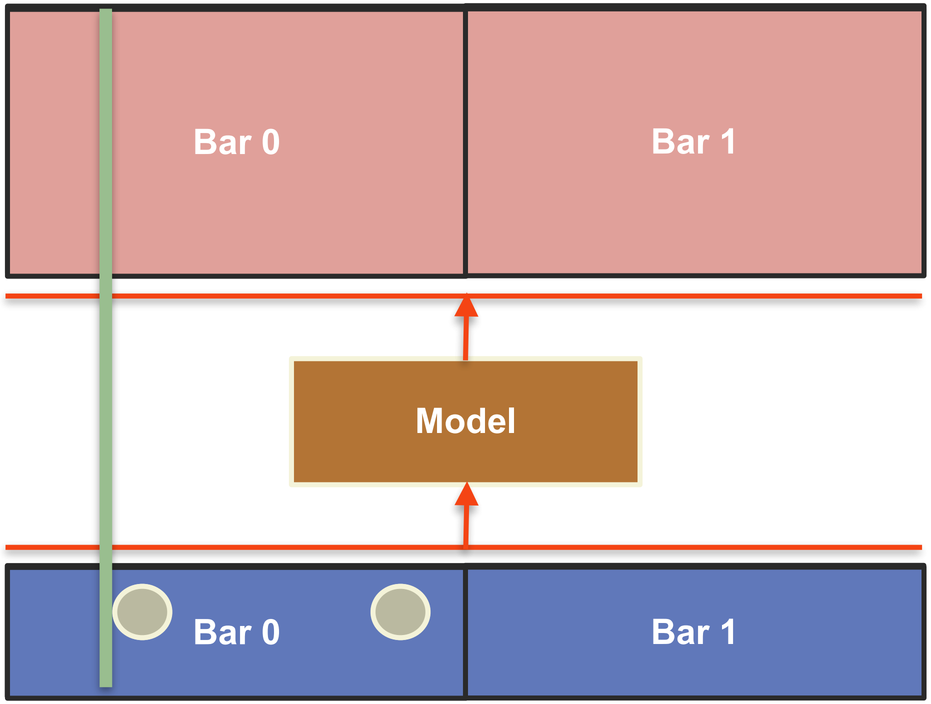
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



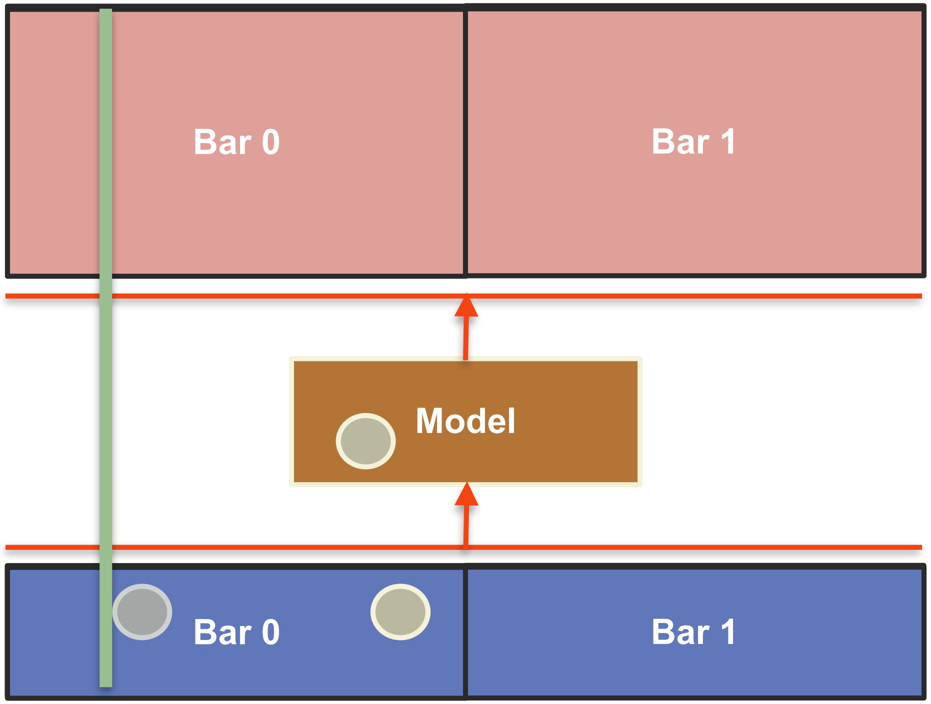
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



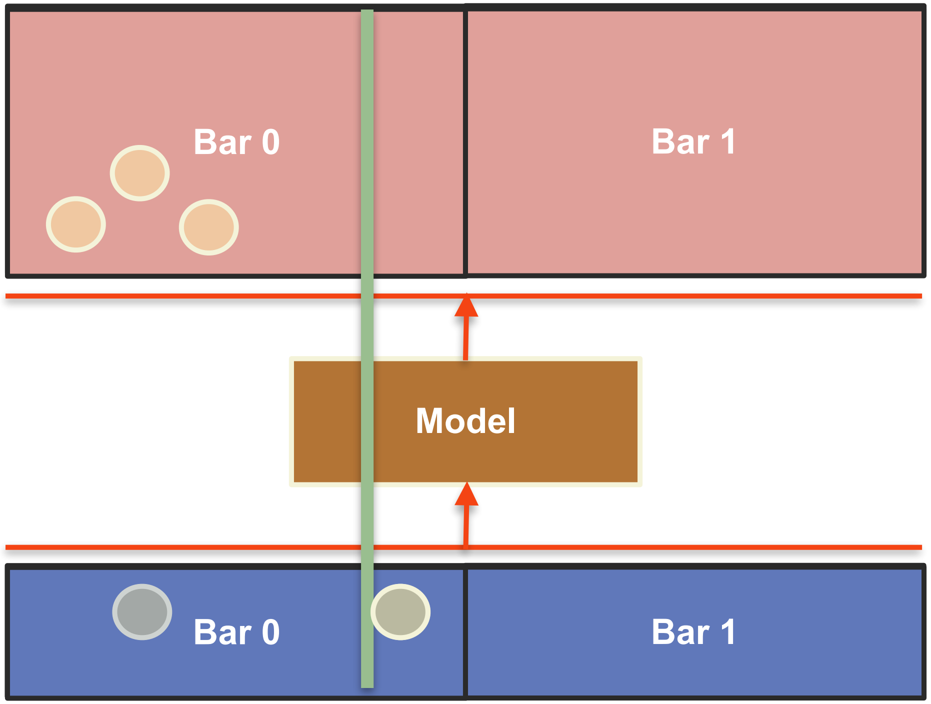
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



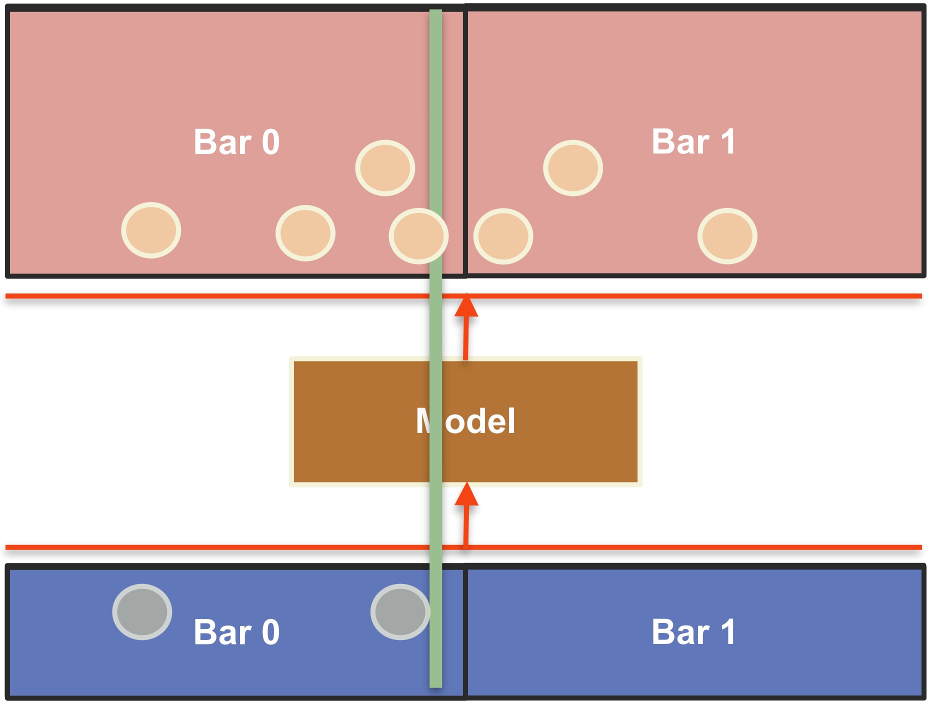
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



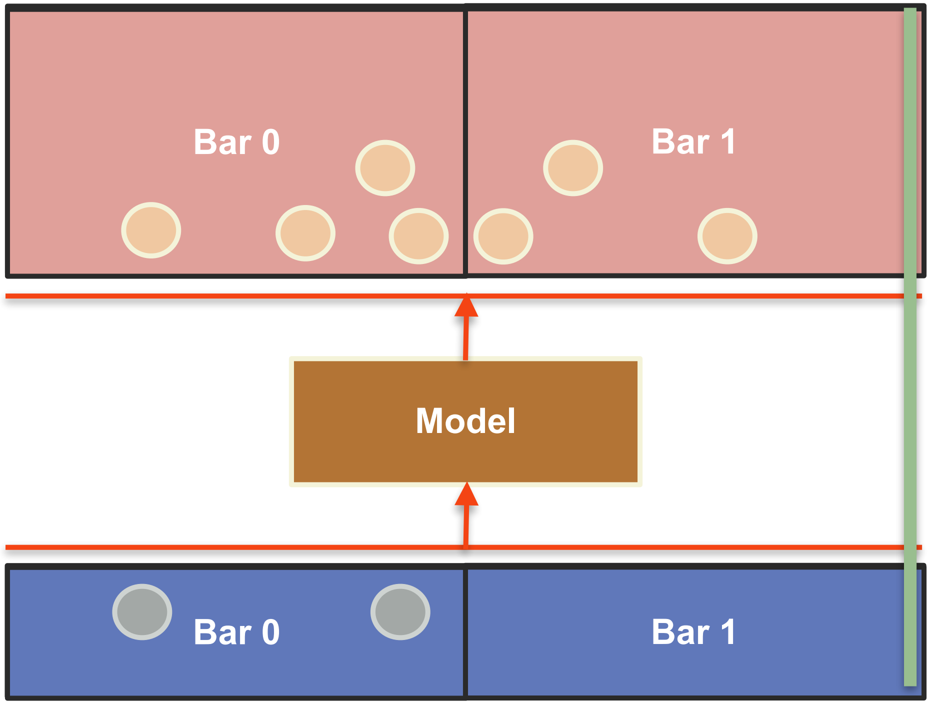
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



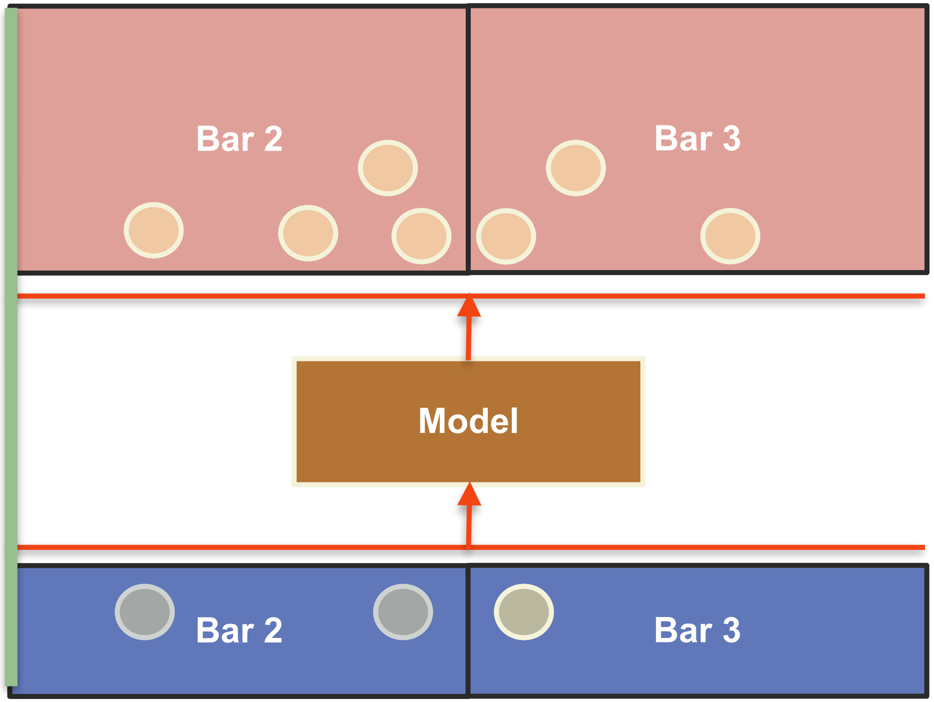
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



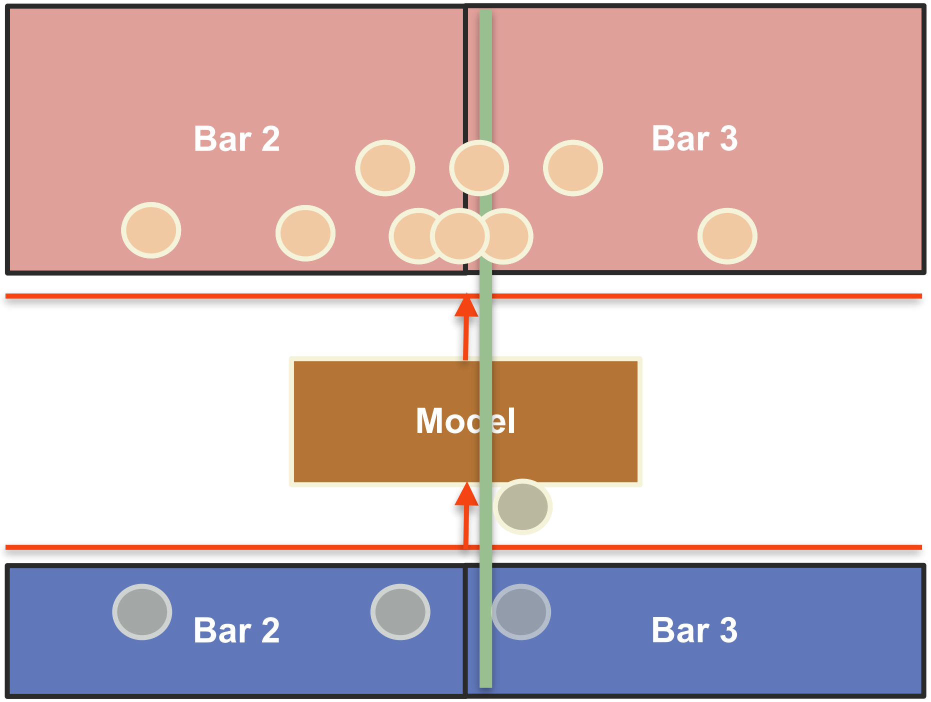
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input



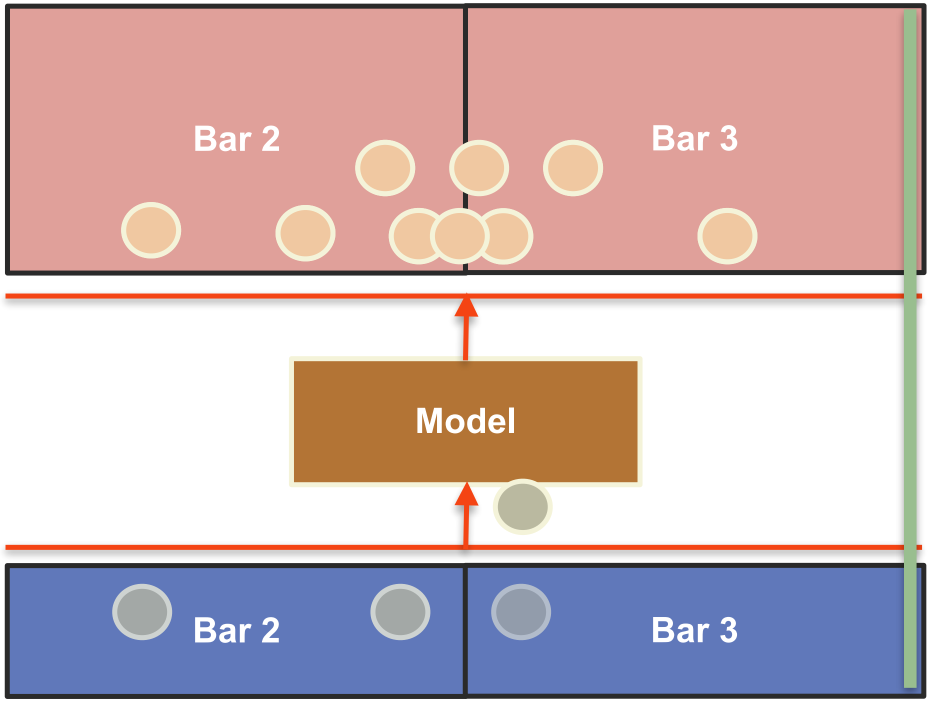
How can we use the trained model in Real-Time?

One major issue with the model:

Always uses 2-bars of input! —> What about long term coherence of the generations?

Ideally, the system should have been trained on longer sequences

We will try to provide information about the past manually by ***OVERDUBBING*** the input


Bar 2
Bar 3
Model
Bar 2
Bar 3