THE MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL UNIVERSITY OF “KYIV-MOHYLA ACADEMY"
FACULTY OF COMPUTER SCIENCES
DEPARTMENT OF COMPUTER SCIENCES

PRESENTATION TO THE BACHELOR THESIS

DIFFUSION MODELS FOR MUSIC GENERATION

Made by:
Savkin Glib Ihorovych

Supervisor:
Kriukova Halyna Vitaliivna

Kyiv 2024

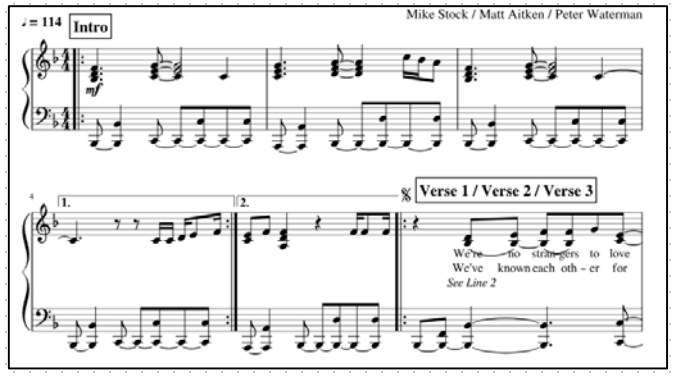
The purpose: the investigation of diffusion networks in the context of symbolic music generation

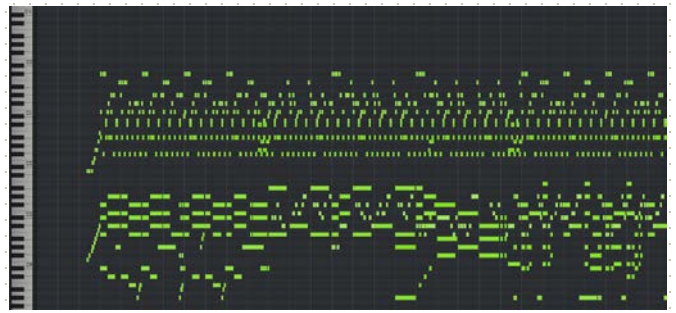
The relevance: the limited amount of research focused on the usage of diffusion models in such a domain

Goals:

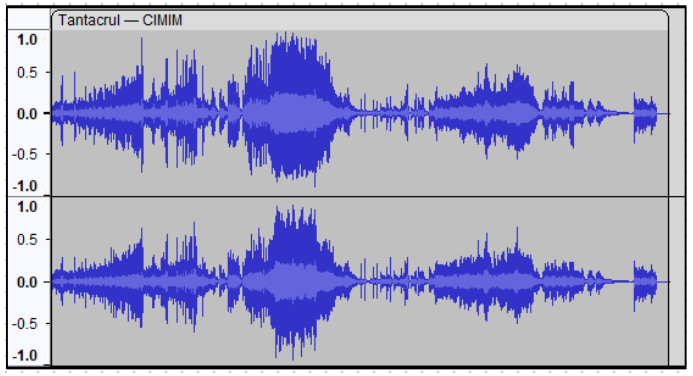
- to investigate existing solutions and their strengths and weaknesses
- to research the inner workings of diffusion networks
- to develop a symbolic music-generating model using the architecture of diffusion networks

Symbolic Data



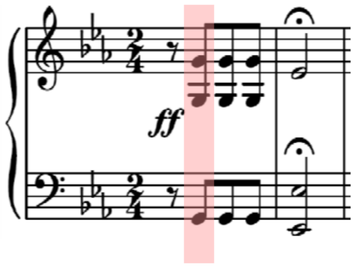


Non-symbolic data

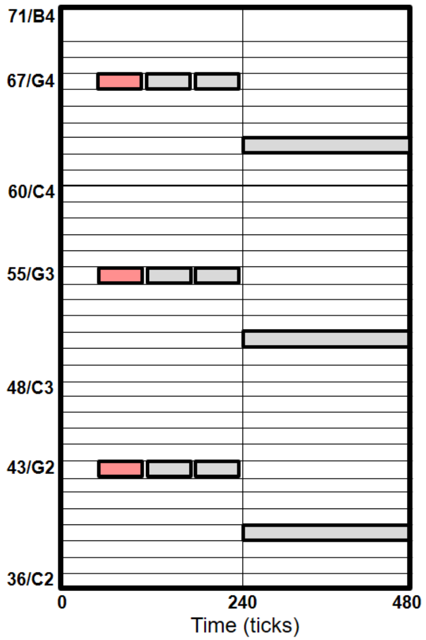


MIDI FILE STRUCTURE

Figure 1.13 from
[Müller, FMP, Springer 2015]



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0



TRACTABILITY VS FLEXIBILITY

Tractability

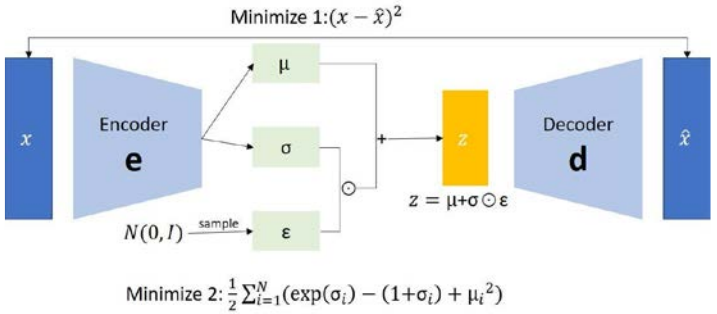
how easy it is to work with a model using mathematical methods and to fit the model to actual data.

Flexibility

how well a model can accurately capture and represent different types of situations or behaviors.

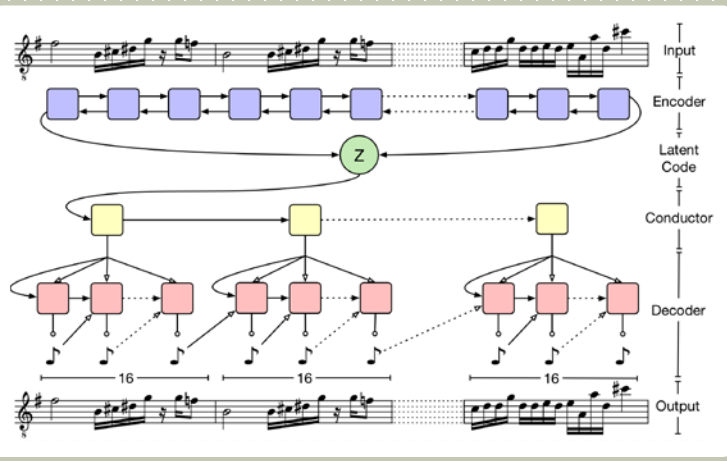
EXISTING MODELS FOR MUSIC GENERATION

Variational Autoencoders



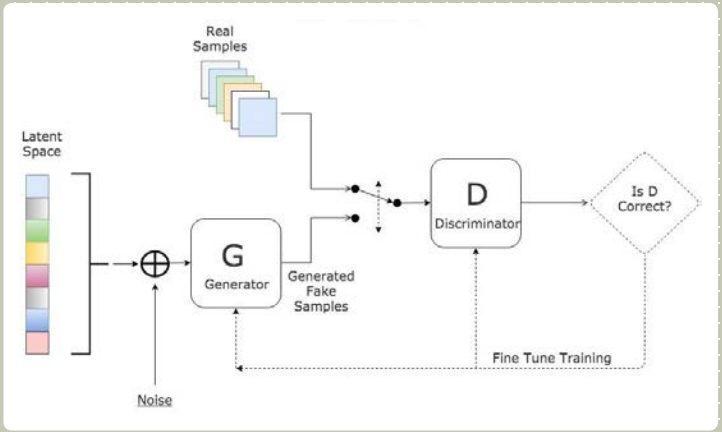
- Encoder – decoder architecture
- Loss – reconstruction loss + KL divergence
- The Reparameterization Trick

MusicVAE



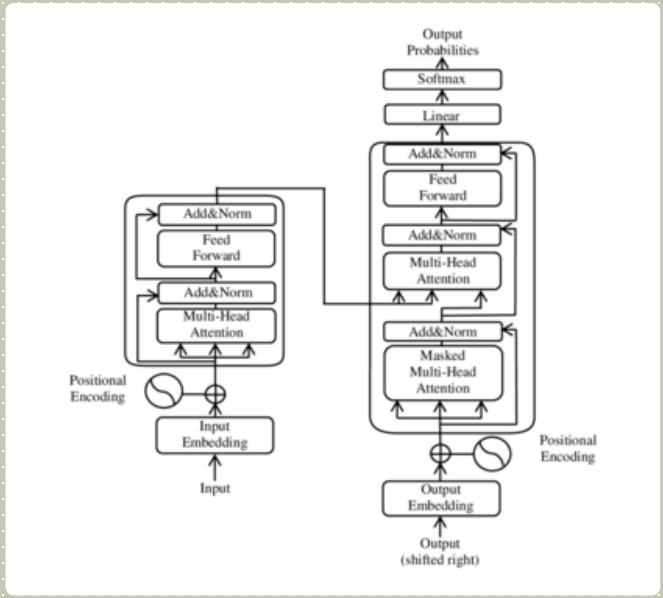
- Temporal dependencies with RNN
- Multilayer latent space

Generative Adversarial Networks



- **Generator** – attempts to generate new data
- **Discriminator** – attempts to classify input as real or fake

Transformer-based Networks

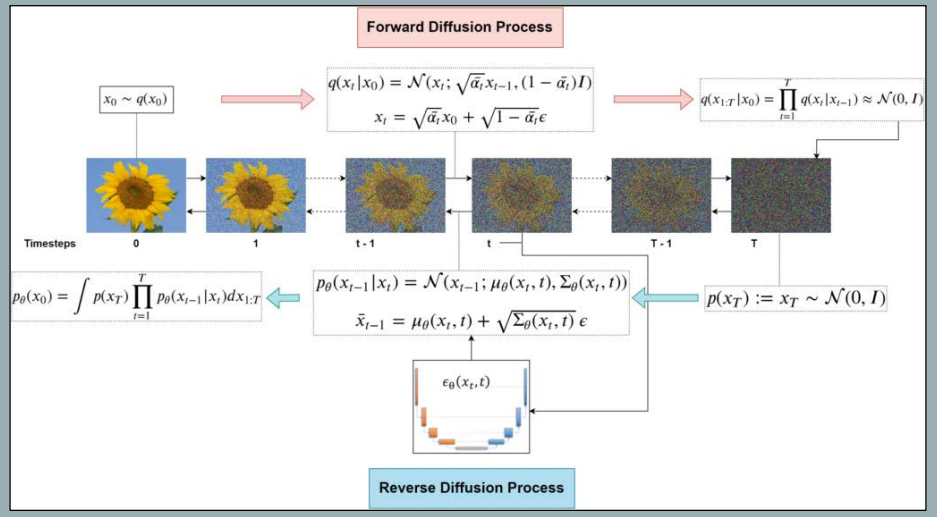


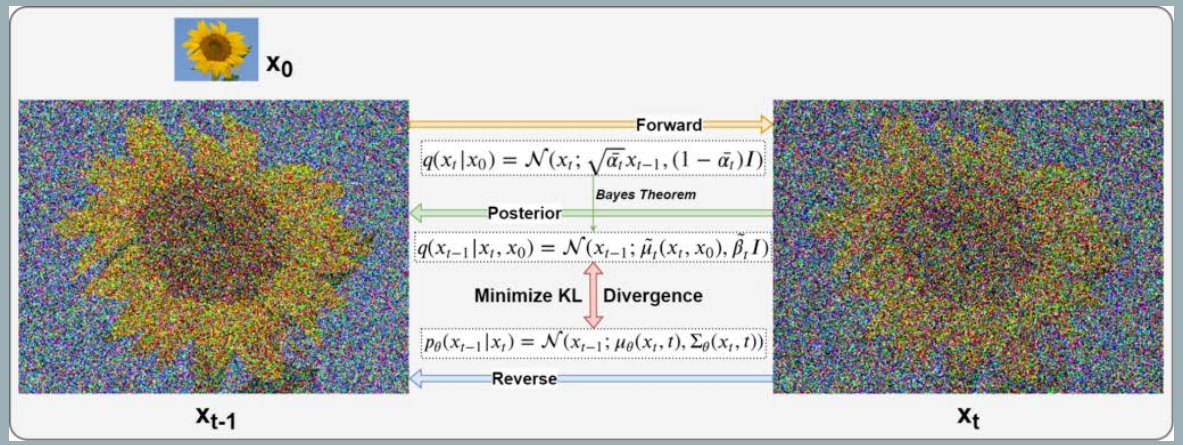
- Built around the idea of attention, perfect for long sequential data

DENOISING DIFFUSION PROBABILISTIC MODELS

- First introduced in a "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" paper in 2015
- Takes inspiration from physics
- Iteratively adds noise to a sample and trains to predict which noise has to be removed

FORWARD AND REVERSE DIFFUSION

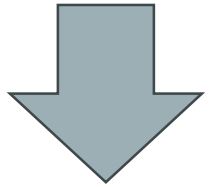




MODEL IMPLEMENTATION

CORE MODEL IDEA

Diffusion models excel at image generation



Convert MIDI data into something that closely resembles image data

DATASET COMPUTATION

```python
TIME_SHIFT_RESOLUTION = 0.01

events = []
for instrument in midi_data.instruments:
    for note in instrument.notes:
        events.append(('note_on', note.start, note.pitch, note.velocity))
        events.append(('note_off', note.end, note.pitch))
events.sort(key=lambda x: x[1])

last_time = 0
event_sequence = []
for event in events:
    event_time = event[1]
    time_shift = event_time - last_time

    if time_shift > 0:
        time_shift_steps = int(time_shift // TIME_SHIFT_RESOLUTION)
        for _ in range(time_shift_steps):
            event_sequence.append(('time_shift', TIME_SHIFT_RESOLUTION))
    event_sequence.append(event)
    last_time = event_time
return event_sequence
```

MIDI files to events conversion

```python
NOTE_ON_FLAG = 0
NOTE_OFF_FLAG = 128
TIME_SHIFT_FLAG = 256

1 usage  glib
def tokenize_event(event):
    if event[0] == 'note_on':
        return NOTE_ON_FLAG + event[2]
    elif event[0] == 'note_off':
        return NOTE_OFF_FLAG + event[2]
    elif event[0] == 'time_shift':
        return TIME_SHIFT_FLAG + int(event[1] / TIME_SHIFT_RESOLUTION)
```

Event tokenization

METRIC AND MODEL DEFINITION

```
# Calculate histograms every 1000 iterations
if index % 1000 == 0:
    generated_sample = diffusion.sample(batch_size=1)[0]
    real_histograms.append(calculate_pitch_class_distance_histogram(generated_sample))
    real_histograms.append(calculate_pitch_class_distance_histogram)
```

Calculated metrics (pitch class histogram distance)

```
model = Unet1D(
    dim=64,
    dim_mults=(1, 2, 4, 8)
)

diffusion = GaussianDiffusion1D(
    model,
    seq_length=dataset.max_length,
    timesteps=1000,
    beta_schedule='cosine',
)
```

Model definition using the denoising_diffusion_pytorch library

CONCLUSIONS

- We have investigated the possibility of symbolic audio generation using diffusion models
- We have compared diffusion models in this task to other architectures
- We have built a diffusion-based model that is capable of generating musical pieces
- We have discussed the issues that plague diffusion models in the domain of music synthesis

THANK YOU FOR YOUR ATTENTION!