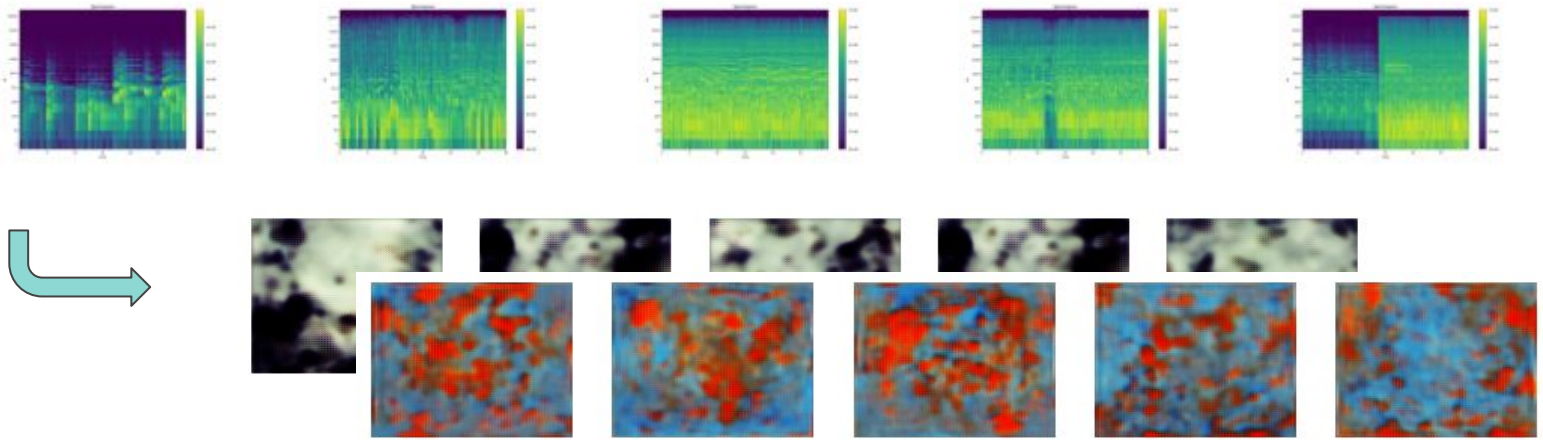
# Generating Visual Art From Music

Alivia Castor
Eli Kerstein

# Motivation

Our motivation for this project is to investigate how deep learning can be used to explore new realms of creativity and expression.

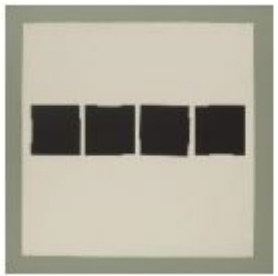
By using AI to process music and generate art, we aim to bridge the gap between auditory and visual experiences, offering innovative ways to interpret and appreciate musical works.



# Goal

Our project aims to create visual art inspired by music. By analyzing musical pieces, we capture key attributes such as mood, energy, and emotion. We then use these attributes as a guide to generate artwork that reflects the essence of the music.

This fusion of art and music aims to provide a visual representation of the auditory experience, enhancing our experience of the musical work.


# Main Reference Works

MeloHarmony: Exploring Emotion in Crafting AI-Generated Music with Generative Adversarial Network Powered Harmony-
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4575257

Generative artificial intelligence, human creativity, and art
https://academic.oup.com/pnasnexus/article/3/3/pgae052/7618478

Transparency in Music-Generative AI: A Systematic Literature Review
https://www.researchsquare.com/article/rs-3708077/v1

# Audio Dataset

Our audio dataset consists of approximately 400 songs. We sourced these tracks from a combination of manually downloaded popular songs from YouTube and free-to-download songs from Last.fm.

We use the **Spotify API** to access metadata for each song, allowing us to extract and label the following key features:

- **Genre:** The music category or style, providing insights into the song's general characteristics.
- **Energy:** A measure of intensity and activity in the track, with higher values indicating more energetic music.
- **Valence:** Represents the emotional content of the song, where higher values indicate a more positive or upbeat mood.

# Audio Data Example

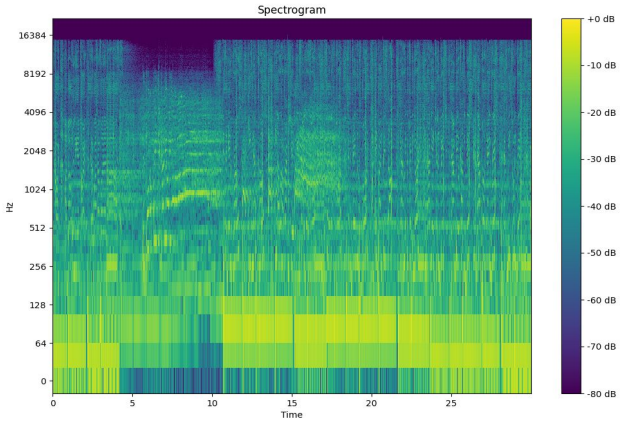
| Title | Valence | Genre | Energy |
|---|---|---|---|
| Foster The People - Pumped Up Kicks (Official Video) | 0.965 | indietronica | 0.71 |
| Ariana Grande - 7 rings (Official Video) | 0.327 | pop | 0.317 |
| Uncle Lucius - Keep The Wolves Away | 0.144 | classic texas country | 0.351 |
| 4 Non Blondes - What's Up (Official Music Video) | 0.366 | new wave pop | 0.67 |
| Shawn Mendes, Camila Cabello - Señorita | 0.749 | canadian pop | 0.548 |
| Jason Aldean - Burnin' It Down | 0.64 | contemporary country | 0.795 |
| The Echelon Effect - Your First Light My Eventide | 0.224 | ambient post-rock | 0.649 |
| Eluveitie - Meet The Enemy | 0.134 | celtic metal | 0.996 |
| Death Grips - Get Got | 0.405 | escape room | 0.993 |
| Kings Of Leon - Sex on Fire (Official Video) | 0.374 | modern rock | 0.905 |

# Spectrogram Generation

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

Spectrogram Generation

To create a spectrogram, we convert audio signals into a visual representation of their frequency content over time. The audio signal is divided into time bins, and each bin is transformed into the frequency domain using the Discrete Fourier Transform (DFT). This process yields a set of frequencies for each time bin.



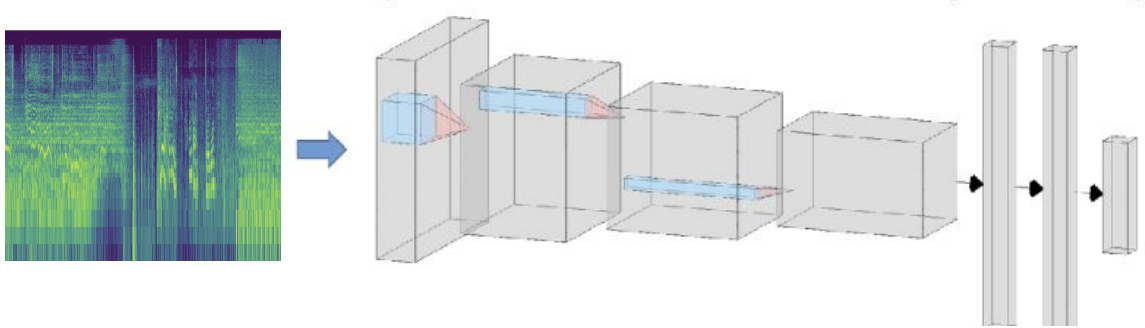

By reducing the time bin size, we achieve finer temporal resolution, enabling us to capture rapid changes and dynamics within the audio signal. This is key in enabling the CNN to capture the mood and energy of the music in the spectrogram.

# Art Dataset

Our art dataset comprises a curated collection of 9,000 abstract art images. The surrealist style, known for its abstract and imaginative qualities, gives us significant creative freedom to represent the distinct characteristics of each musical piece.

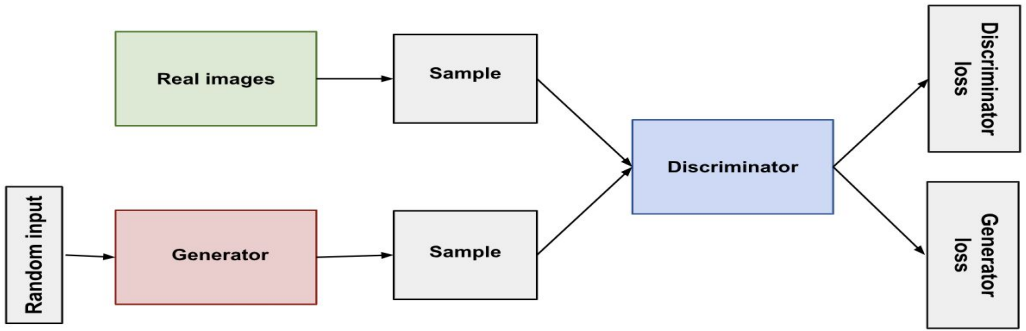
# Architecture

$$J_D = -E_{x \sim P_{data}}[\log D(x)] - E_{z \sim P_z}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$



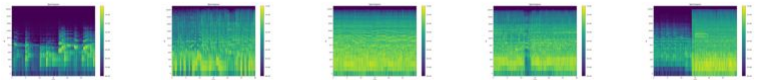
Extracts Features such as valence and energy

GAN is conditioned on the features extracted from the CNN



# Experimental Setup

```python
def build_cnn_classifier(input_shape):
    input_layer = Input(shape=input_shape)
    x = Conv2D(32, (5, 5), activation='relu', padding='same', kernel_regularizer=l2(0.01))(input_layer)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.3)(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = BatchNormalization()(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.3)(x)

    x = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = BatchNormalization()(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.3)(x)

    x = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = BatchNormalization()(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01))(x)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.3)(x)

    x = Flatten()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)

    # Change to a single output layer with 2 units for valence and energy
    outputs = Dense(2, activation='linear')(x)

    model = Model(inputs=input_layer, outputs=outputs)
    model.compile(optimizer=RMSprop(learning_rate=0.0001, momentum=0.9),
                  loss='mean_squared_error',  # Use one loss for the entire output
                  metrics=['mean_squared_error'])
    return model
```



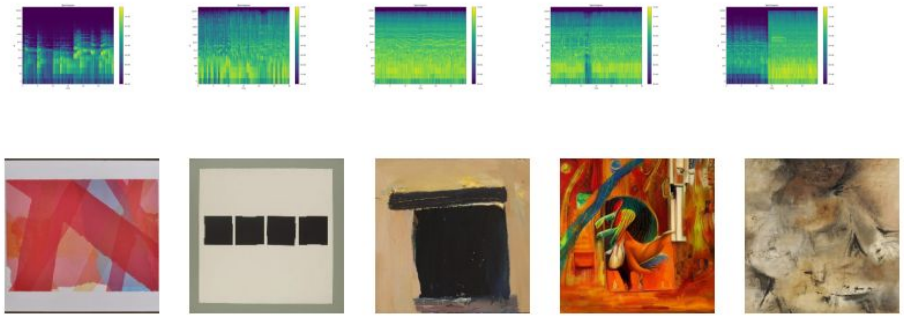
```python
datagen = ImageDataGenerator(
    rotation_range=10,  # Incr
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    brightness_range=(0.8, 1.2)
    fill_mode='nearest'
)
```

```python
# Create ModelCheckpoint callback to save
checkpoint = ModelCheckpoint(
    checkpoint_path,
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

# Create EarlyStopping callback to halt t
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    verbose=1,
    restore_best_weights=True
)

# Function to update the learning rate
def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * tf.math.exp(-0.1)
```

# Experimental Setup

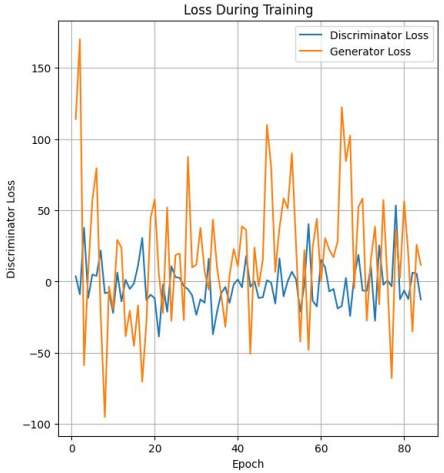
```python
# Build Generator Model
def build_generator(latent_dim):
    generator_input = Input(shape=(latent_dim,))
    x = Dense(128 * 64 * 64)(generator_input)
    x = LeakyReLU()(x)
    x = Reshape((64, 64, 128))(x)
    x = BatchNormalization()(x)
    x = Conv2DTranspose(128, 5, strides=2, padding='same')(x)
    x = LeakyReLU()(x)
    x = BatchNormalization()(x)
    x = Conv2DTranspose(64, 5, strides=2, padding='same')(x)
    x = LeakyReLU()(x)
    x = Conv2DTranspose(3, 7, activation='tanh', padding='same')(x)
    generator = Model(inputs=generator_input, outputs=x)
    return generator


# Build Discriminator Model
def build_discriminator(image_shape):
    discriminator_input = Input(shape=image_shape)
    x = Conv2D(64, 5, strides=2, padding='same')(discriminator_input)
    x = LeakyReLU()(x)
    x = Dropout(0.4)(x)
    x = Conv2D(128, 5, strides=2, padding='same')(x)
    x = LeakyReLU()(x)
    x = Dropout(0.4)(x)
    x = Conv2D(256, 5, strides=2, padding='same')(x)
    x = LeakyReLU()(x)
    x = Dropout(0.4)(x)
    x = Flatten()(x)
    x = Dense(1)(x)
    discriminator = Model(inputs=discriminator_input, outputs=x)
    return discriminator
```

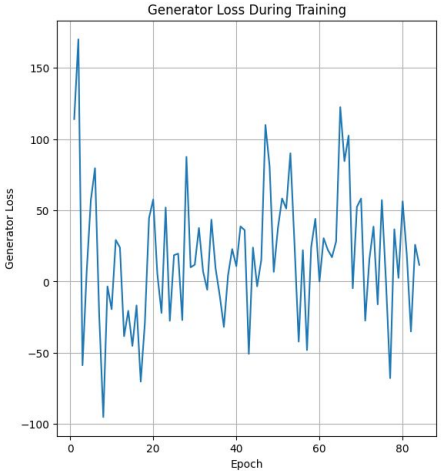
```python
# Training Generator
noise = np.random.normal(0, 1, (batch_size, latent_dim))
noise = tf.convert_to_tensor(noise, dtype=tf.float32)  # Convert noise to tensor again
with tf.GradientTape() as tape:
    gen_imgs = generator(noise, training=True)
    g_loss = -discriminator(gen_imgs, training=True)
g_gradients = tape.gradient(g_loss, generator.trainable_variables)
generator_optimizer.apply_gradients(zip(g_gradients, generator.trainable_variables))
```



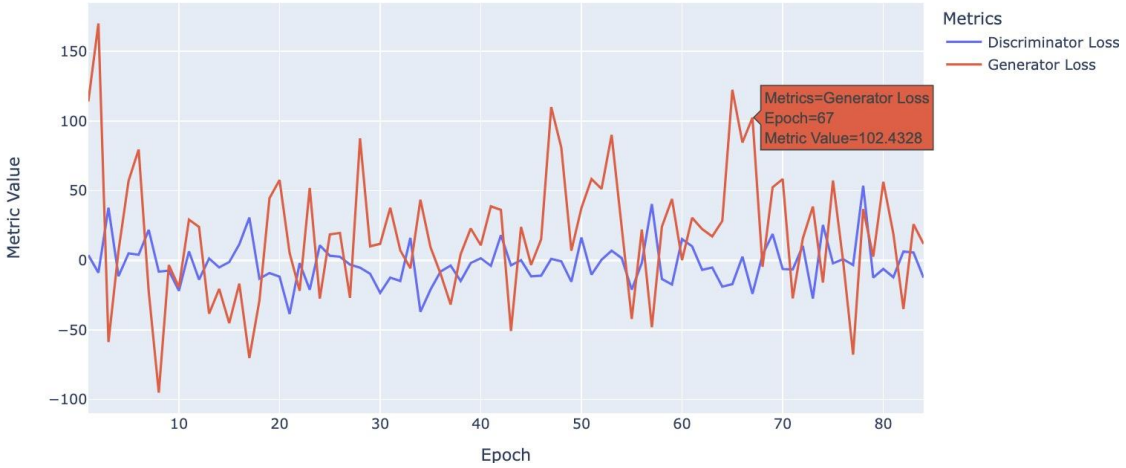
# Training Details

## GAN



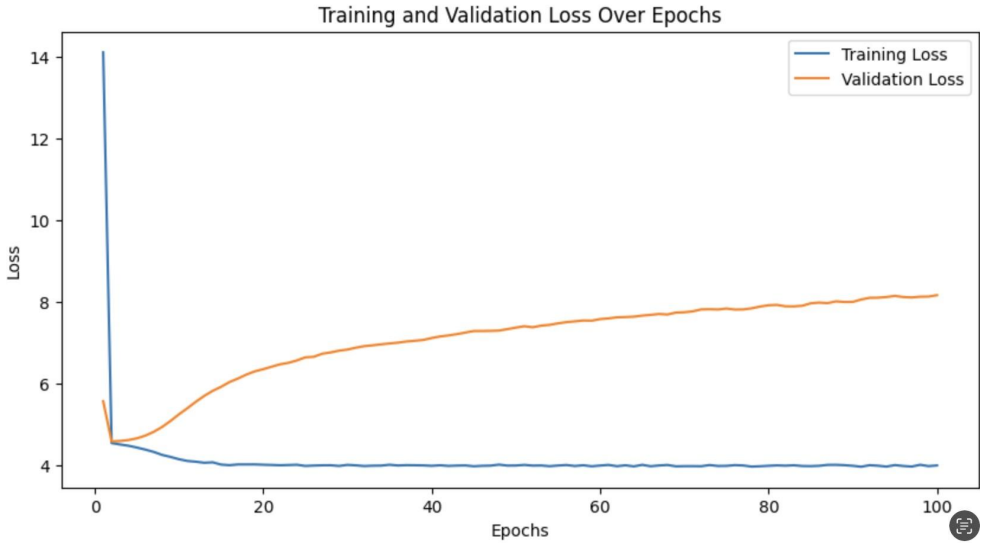




Interactive Training Metrics Visualization



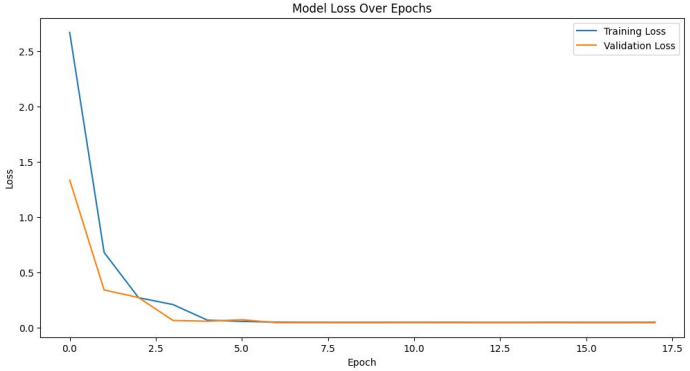
# Training Details

## CNN

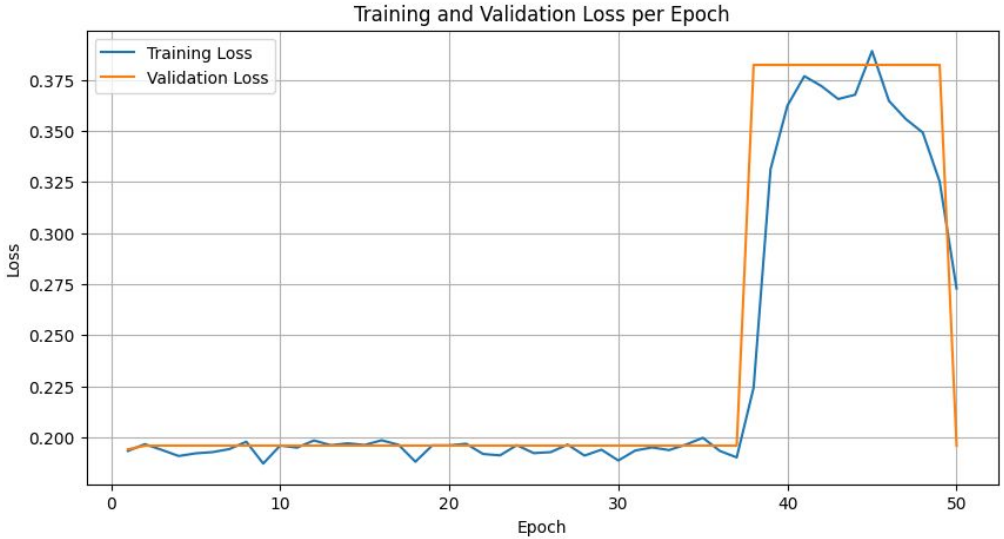


# Training Details

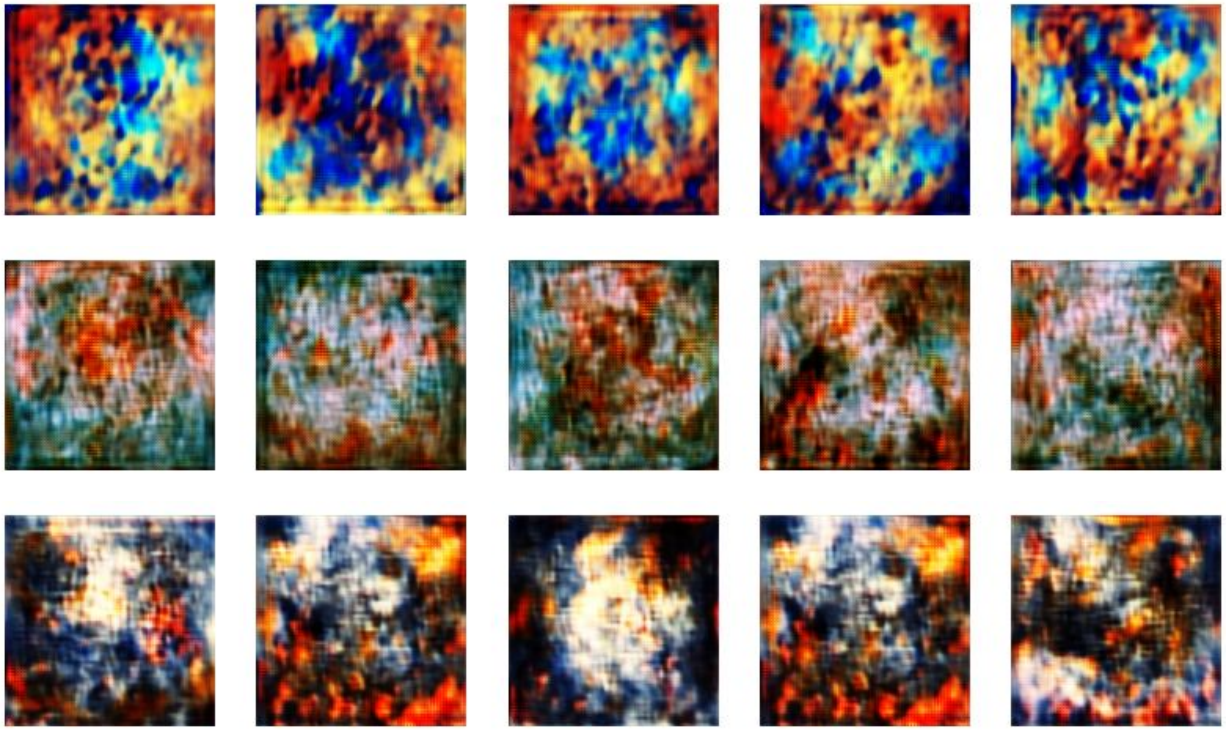
CNN

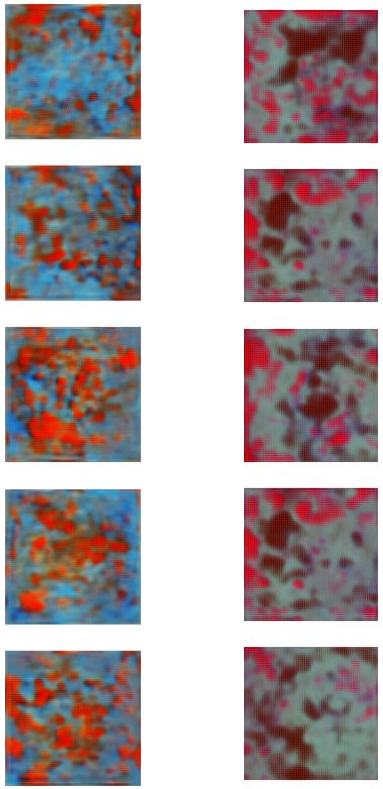



GAN



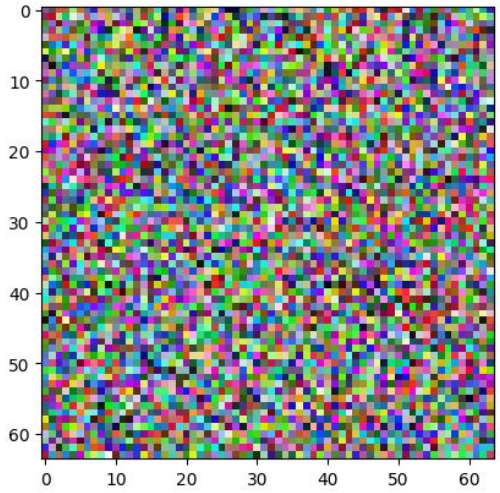
# Experimental Results 1





# Future Work

- Lucid Sonic Dreams
- Dynamic Renditions
- Enhance GAN & Dual Discrimination Possibilities
- WGAN, LSGAN



# Github Link

git@github.com:sajc11/COSC_5470_ARTGEN.git