Algorithm Design Architecture

The **algorithm** for generating Alien House Music is built on a modular framework combining symbolic and audio-domain processes. It leverages a hierarchical Variational Autoencoder (MusicVAE) for interpolation and blending of musical ideas in latent space, complemented by custom diffusion-based methods for timbral and textural synthesis[1]. The core pipeline consists of an **Audio Encoder/Decoder**, a **Text Encoder**, a **Fusion Network**, and a **Generation Engine**, each optimized for two-minute track creation. To ensure genre fidelity, the model integrates genre-specific rhythmic templates derived from Wavelet-based Beat Histograms which capture periodicities in the 40-200 BPM range characteristic of House music[2]. Finally, a post-generation **Diversity Controller** layer evaluates novelty metrics such as inter-track spectral variance and Self-BLEU-inspired diversity scores on note sequences to prevent monotony[3].

Model Components

- **MusicVAE Latent Space**: Uses a bidirectional LSTM encoder and hierarchical LSTM decoder to process 2-bar and 16-bar segments, yielding a continuous Z-space for interpolation tasks[4] .
- **Text and Audio Embeddings**: Text prompts are encoded via Facebook's CLAP model for text-audio alignment, while raw audio features (mfccs, spectrograms) are extracted using Torchaudio's Spectrogram and MelSpectrogram transforms[5].
- **Diffusion Backbone**: A UNet-like Conditional Diffusion Model denoises audio spectrograms guided by multi-modal controls, inspired by Music ControlNet's time-varying control strategy for melody, rhythm, and dynamics[1].
- **Novel Genre Filter**: Implements a Gaussian Mixture Model over beat and spectral features (e.g., spectral centroid, rolloff) to enforce "alien" textures by sampling from out-of-distribution regions of the learned latent distribution[3].

Input Mapping

The input mapping translates three user-specified modalities-text, audio, and sound choice-into embeddings compatible with our generative backbone. This ensures a coherent fusion of lyrical, contextual, and timbral cues.

Text-to-Embedding

Text prompts detailing mood, style, or narrative cues are processed by a pre-trained LLM (T5) to obtain 512-dimensional contextual embeddings. We further refine these via a Retrieval-Augmented Text-to-Music encoder using Spotify's Annoy library to query similar captions from the MusicCaps dataset, enhancing genre adherence and diversity.

```
from transformers import AutoTokenizer, AutoModel
import annoy
```

```
tokenizer = AutoTokenizer.from_pretrained("UMT5")
model = AutoModel.from_pretrained("UMT5")

# Pre-compute text embeddings
text_embeddings = model(**tokenizer(prompts, return_tensors="pt")).last_hidden_state
index = annoy.AnnoyIndex(512, 'euclidean')
for i, emb in enumerate(text_embeddings):
index.add_item(i, emb.numpy())
index.build(10)
```

Audio Feature Extraction

Raw audio inputs-such as a user's favorite track-are mapped through torchaudio.functional and torchaudio.transforms to extract spectral, rhythmic, and timbral embeddings:

- **Spectrogram** (n_fft=512, hop_length=128): Captures time-frequency content[5].
- **MelSpectrogram**: Provides perceptually scaled features for timbre mapping.
- **MFCC & Chroma**: Extracts harmonic and pitch information for melodic color mapping.

```
import torchaudio.transforms as T

wav, sr = torchaudio.load("user_audio.wav")
mel_spec = T.MelSpectrogram(sr, n_fft=1024, hop_length=256)(wav)
mfcc = T.MFCC(sr, n_mfcc=13, melkwargs={'n_fft':1024, 'hop_length':256})(wav)
```

Sound Choice Integration

Users can upload custom sound samples (e.g., alien ambience, metallic clangs). These are processed via an **OpenL3** embedding model or **VGGish**, yielding 512-dimensional vectors representing timbral "identity vectors" for seamless fusion:

```
import openl3

embeddings, ts = openl3.get_audio_embedding("alien_clip.wav", sr, input_repr="mel128",
embedding_size=512)
```

Fusion Engine Techniques

Our **Fusion Engine** merges text, audio, and sound choice signals in a shared latent space, balancing global style and time-varying details.

Cross-Attention Fusion

We adopt a **Dominant Head Fusion** strategy inspired by MAiVAR-T, concatenating modality-specific vectors and refining them through a transformer-based cross-attention block[6]. This generates a fused sequence:

- Query: MusicVAE latent features.
- Keys/Values: Text, audio, sound choice embeddings.
- Output: Contextually enriched latent representation for spectrogram generation.

MusicVAE Interpolation

Interpolations occur in the MusicVAE latent space, enabling smooth transitions between two seed embeddings, accommodating user-provided text and audio contexts:

```
from magenta.models.music_vae import TrainedModel

mvae = TrainedModel(config="cat-mel_2bar_big", checkpoint_file="cat-mel_2bar_big.ckpt")
z1 = mvae.encode([seed_seq_1])[0]
z2 = mvae.encode([seed_seq_2])[0]
interpolations = mvae.interpolate(z1, z2, num_steps=8)
decoded_seqs = mvae.decode(interpolations)
```

This process produces two-minute-long sequences by chaining interpolated segments while preserving House-style rhythmic anchors via a **Beat Histogram** control vector.

Export Options

Generated tracks are exportable as **MIDI** or **WAV**, ensuring compatibility with DAWs and playback environments.

MIDI Export

We convert decoded Magenta sequences into MIDI using **Mido** and **MIDIUtil**, setting tempo and channel information:

```
from magenta.music import midi_io

for i, seq in enumerate(decoded_seqs):
path = f"alien_house_{i}.mid"
midi_io.sequence_proto_to_midi_file(seq, path)
```

Alternative direct writing with MIDIUtil:

```
from midiutil import MIDIFile

midi = MIDIFile(1)
midi.addTempo(0, 0, 124)
for note in midi_notes:
midi.addNote(0, 0, note.pitch, note.start, note.duration, note.velocity)
with open("alien_house_track.mid","wb") as f:
midi.writeFile(f)
```

WAV Rendering

Spectrograms are decoded into waveforms via a **DiffWave** or **EnCodec** vocoder:

```
import diffwave

waveform = diffwave.decode_spectrogram(spec)
torchaudio.save("alien_house.wav", waveform, sample_rate=32000)
```

Alternatively, using OpenVINO-optimized components:

```
audio_values = audio_decoder_wrapper.decode(output_ids, audio_scales)
audio_values_np = audio_values.audio_values.numpy()
sf.write("alien_house_openvino.wav", audio_values_np, sampling_rate)
```

High-fidelity rendering employs dynamic-range scaling and multi-band processing to emphasize alien textures. Continuous evaluation with **Frechet Audio Distance** (FAD) and **CLAP-based** alignment metrics ensure both audio quality and text adherence across generated tracks.

References (6)

1. *MUSIC CONTROLNET: Multiple Time-varying Controls for Music Generation*. https://arxiv.org/pdf/2311.07069
2. *Musical genre classification of audio signals - Speech and Audio ...*. https://dspace.library.uvic.ca/server/api/core/bitstreams/d7457cdf-e42f-4772-b9ee-801adf43f949/content
3. *A Q -D BASED EVALUATION STRATEGY FOR SYMBOLIC MUSIC GENERATION*. https://ml-eval.github.io/assets/pdf/A_Quality_Diversity_Based_Evaluation_Strategy_For_Symbolic_Music_Generation.pd
4. *Google Colab*. https://colab.research.google.com/github/magenta/magenta-demos/blob/master/colab-notebooks/MusicVAE.ipynb
5. *Audio Feature Extractions - Torchaudio 2.7.0 documentation*. https://docs.pytorch.org/audio/stable/tutorials/audio_feature_extractions_tutorial.html
6. *Pipeline Structure* . https://deepwiki.com/ace-step/ACE-Step/2.1-pipeline-structure