MATCHA-TTS IMPLEMENTATION

Team Work

Standalone `model.py` – Matcha-TTS Skeleton Guide

This document describes how to create a **standalone `model.py` file** for Matcha-TTS, without relying on the original codebase.

1. Utility Functions Section

git copied **exactly** from `matcha/utils/model.py`.

Required utility functions (Arezki)

- `sequence_mask(length, max_length=None)` ✅
- `fix_len_compatibility(length, num_downsamplings_in_unet=2)` ✅
- `convert_pad_shape(pad_shape)` ✅
- `generate_path(duration, mask)`✅
- `duration_loss(logw, logw_, lengths)` ✅
- `denormalize(data, mu, std)` ✅

Alignment fallback(Arezki)

Add a simple monotonic alignment fallback:

- `maximum_path(value, mask)` ✅

---

2. Text Encoder Components Section

Write the following classes **in this exact order**.

2.1 Basic Building Blocks (Lounes)

- `LayerNorm(nn.Module)` ✅
- `ConvReluNorm(nn.Module)` ✅
- `DurationPredictor(nn.Module)` ✅

2.2 Attention Components(Lounes)

- `RotaryPositionalEmbeddings(nn.Module)` ✅

- `MultiHeadAttention(nn.Module)` ✅
- `FFN(nn.Module)` ✅
- `Encoder(nn.Module)` ✅

2.3 Main Text Encoder(Lounes)

- `TextEncoder(nn.Module)` ✅

---

3. Decoder / UNet Components Section

Implement the UNet-based decoder in the following order.

3.1 UNet Building Blocks (fayçal)

- `SinusoidalPosEmb(nn.Module)` ✅
- `Block1D(nn.Module)` ✅
- `ResnetBlock1D(nn.Module)` ✅
- `Downsample1D(nn.Module)`✅
- `Upsample1D(nn.Module)`✅
- `TimestepEmbedding(nn.Module)`✅

3.2 Transformer Components for UNet (mira)

- `SnakeBeta(nn.Module)`✅
- `FeedForward(nn.Module)`✅
- `BasicTransformerBlock(nn.Module)`✅

3.3 Main UNet (mira)

- `Decoder(nn.Module)` ✅

---

4. Flow Matching Section (faycal)

- `BASECFM(nn.Module)` ✅
  Base class for conditional flow matching
- `CFM(BASECFM)` ✅
  Concrete CFM implementation using the UNet estimator

---

5. Main Model Class (Arezki)

Create the main model class:

- `MatchaTTS(nn.Module)` ✅

Required methods (Arezki)

- `__init__()`
  Initialize all components
- `update_data_statistics()`
  Set mel-spectrogram normalization statistics
- `synthesise()`
  Main inference-only synthesis pipeline
- `load_from_checkpoint()`
  Static method to load pretrained weights

---

7. Configuration Constants

```
# Default model configuration (hardcoded from matcha.yaml + encoder/decoder/cfm configs)
DEFAULT_CONFIG = {
    'n_vocab': 178,
    'n_spks': 1,
    'spk_emb_dim': 64,
    'n_feats': 80,
    'encoder': {
        'encoder_type': 'RoPE Encoder',
        'encoder_params': {
            'n_feats': 80,
            'n_channels': 192,
            'filter_channels': 768,
            # ... etc
        },
        'duration_predictor_params': {
            # ... etc
        }
    },
    'decoder': {
        'channels': [256, 256],
        'dropout': 0.05,
        # ... etc
```

```
    },
    'cfm': {
        'solver': 'euler',
        'sigma_min': 1e-4
    }
}
```
