MERT: Acoustic Music Understanding Model with Large-Scale Self-supervised Training

Nikol Toda, Matricola: 2047236

January 9, 2024

Contents

1 Methodology **3**
1.1 Custom Transformer Encoder: TransformerEncoderExtend . . . 3
1.1.1 Architecture Design: . . . 3
1.1.2 Feature Enhancement: . . . 3
1.1.3 Layer Customization: . . . 3
1.2 Enhanced Attention Mechanism: MultiheadAttentionExtend . . . 3
1.2.1 Attention Mechanism Refinement: . . . 3
1.2.2 Improvement in Contextual Understanding: . . . 3
1.2.3 Custom Implementation Details: . . . 3

2 Detailed Exploration of Technological Stack and Implementation Details **4**
2.1 Core Technologies and Libraries . . . 4
2.1.1 Python and PyTorch Framework: . . . 4
2.1.2 Transformers Library: . . . 4
2.1.3 Additional Libraries - torchtext, datasets, torchaudio: . . . 4
2.2 Advancing NLP: Customization and Innovation . . . 4
2.2.1 Custom Transformer Encoder - TransformerEncoderExtend: . . . 4
2.2.2 Enhanced Attention Mechanism - MultiheadAttentionExtend: . . . 4
2.2.3 Advanced Techniques and Novel Concepts: . . . 4

3 Detailed code examples: **5**

4 Summary **5**
4.1 Data Representation: . . . 5
4.2 Analysis of Output Values: . . . 5
4.3 Potential Interpretations: . . . 5

List of Figures

1 Transformer . . . 5
2 Multihead . . . 5
3 MERT . . . 5
4 Result . . . 5

Abstract

Self-supervised learning (SSL) has recently emerged as a promising paradigm for training generalisable models on large-scale data in the fields of vision, text, and speech. Although SSL has been proven effective in speech and audio, its application to music audio has yet to be thoroughly explored. This is partially due to the distinctive challenges associated with modelling musical knowledge, particularly tonal and pitched characteristics of music. To address this research gap, we propose an acoustic Music undERstanding model with large-scale self-supervised Training (MERT), which incorporates teacher models to provide pseudo labels in the masked language modelling (MLM) style acoustic pre-training. In our exploration, we identified an effective combination of teacher models, which outperforms conventional speech and audio approaches in terms of performance. This combination includes an acoustic teacher based on Residual Vector Quantization - Variational AutoEncoder (RVQ-VAE) and a musical teacher based on the Constant-Q Transform (CQT). Furthermore, we explore a wide range of settings to overcome the instability in acoustic language model pre-training, which allows our designed paradigm to scale from 95M to 330M parameters. Experimental results indicate that our model can generalise and perform well on 14 music understanding tasks and attain state-of-the-art (SOTA) overall scores.

1 Methodology

1.1 Custom Transformer Encoder: TransformerEncoderExtend

1.1.1 Architecture Design:

The 'TransformerEncoderExtend' class represents a significant modification of the standard Transformer encoder. This custom encoder is designed to increase the model's adaptability and performance in processing complex language data.
Key architectural elements include an adjustable number of encoder layers, each capable of being individually tuned. This flexibility allows for more nuanced modeling of language structures, adapting to varying complexities found in different NLP tasks.

1.1.2 Feature Enhancement:

Each encoder layer in 'TransformerEncoderExtend' is built to enhance certain features of the Transformer model. Notably, the embedding dimensions and dropout rates are configurable, enabling fine-tuning for specific datasets and applications.
This customization extends the model's capability beyond standard applications, allowing it to handle more challenging NLP tasks that require a deeper understanding of context and nuance.

1.1.3 Layer Customization:

The layers within the encoder are not just replicas of the standard Transformer layer but are re-engineered to incorporate advanced NLP concepts. This includes modifications to the attention mechanism and feed-forward networks, catering to the specific demands of your NLP application.

1.2 Enhanced Attention Mechanism: MultiheadAttentionExtend

1.2.1 Attention Mechanism Refinement:

The 'MultiheadAttentionExtend' class is a refined version of the traditional multi-head attention mechanism found in Transformers. It's tailored to provide a more granular attention process, which is crucial for understanding complex linguistic patterns and dependencies.
This enhanced attention mechanism is likely to improve the model's ability to distinguish and prioritize different segments of the input data, a crucial aspect in tasks like contextual understanding and semantic representation.

1.2.2 Improvement in Contextual Understanding:

The modifications in the attention mechanism aim to enhance the model's contextual understanding. By adjusting how attention is distributed across different parts of the input, the model can potentially gain a more nuanced understanding of language, improving its performance in tasks such as sentiment analysis, context-sensitive translation, or abstract summarization.

1.2.3 Custom Implementation Details:

The implementation includes specific adjustments to the attention calculation, which may involve scaling the attention scores or altering the way these scores are combined with the input embeddings. Such changes are indicative of an experimental approach to improve the Transformer's inherent capabilities.

The methodology employed in this project reflects a deep engagement with cutting-edge NLP research and practices. By customizing key components of the Transformer architecture, your project pushes the boundaries of what these models can achieve, particularly in understanding and processing complex, nuanced language data.
This detailed focus on methodology indicates a project that is not just applying existing models but actively seeking to innovate and contribute to the advancement of NLP technology. The project's success could have significant implications for the field, potentially leading to more effective and sophisticated language processing tools.

2 Detailed Exploration of Technological Stack and Implementation Details

2.1 Core Technologies and Libraries

2.1.1 Python and PyTorch Framework:

The project is developed in Python, a versatile and widely-used programming language in the field of data science and machine learning.
PyTorch is the primary framework used for constructing the neural network components. Its dynamic computation graph and comprehensive ecosystem make it a top choice for developing sophisticated machine learning models, particularly in the field of NLP.

2.1.2 Transformers Library:

he use of the transformers library, known for its extensive collection of pre-trained models and Transformer components, plays a crucial role. It provides a solid foundation for both leveraging and extending state-of-the-art Transformer architectures.
This library simplifies the implementation of complex models and allows for an efficient comparison with benchmark models in NLP tasks.

2.1.3 Additional Libraries - torchtext, datasets, torchaudio:

torchtext is employed for efficient text processing and data loading, which is crucial for handling linguistic datasets.
The datasets library aids in managing large-scale language datasets, ensuring smooth data handling and preprocessing.
torchaudio offers supplementary functionalities, although its specific role would depend on whether the project encompasses audio data processing in the context of NLP.

2.2 Advancing NLP: Customization and Innovation

2.2.1 Custom Transformer Encoder - TransformerEncoderExtend:

The implementation of TransformerEncoderExtend suggests a deep customization of the Transformer encoder.
Key adaptations include variable encoder layers, adjustable embedding dimensions, and tunable dropout rates. These allow the model to adapt to different linguistic datasets and tasks, potentially improving performance and accuracy.
The encoder's flexibility in handling various types of input data points to its potential application in diverse NLP scenarios, from text classification to language generation tasks.

2.2.2 Enhanced Attention Mechanism - MultiheadAttentionExtend:

he modified multi-head attention mechanism, MultiheadAttentionExtend, represents a significant enhancement over the standard attention process in Transformers.
This customized attention module could offer more refined control over the model's focus, allowing it to better capture contextual nuances and complex dependencies in language data.
Such an enhancement is particularly crucial in tasks like semantic analysis, where understanding the subtle context is key.

2.2.3 Advanced Techniques and Novel Concepts:

The code snippets indicate the use of advanced neural network techniques, including layer normalization, customized forward passes, and innovative adaptations of standard Transformer components.
The project appears to incorporate cutting-edge concepts in NLP, potentially contributing to the field through novel methodologies or improved performance metrics.

3 Detailed code examples:

```
class TransformerEncoderExtend(nn.Module):
    def __init__(self, args):
        super().__init__()
        self.args = args
        self.embedding_dim = args.embedding_dim
        self.dropout = args.dropout
        self.layer_type = args.layer_type
        self.encoder_layers = args.encoder_layers
        self.attention_relax = args.attention_relax

        self.layers = nn.ModuleList([
            self.build_encoder_layer(args) for _ in range(args.encoder_layers)
        ])
```

Figure 1: Transformer

```
class MultiheadAttentionExtend(nn.MultiheadAttention):
    def __init__(
        self,
        embed_dim,
        num_heads,
        kdim=None,
        vdim=None,
        dropout=0.0,
        bias=True,
        add_bias_kv=False,
        add_zero_attn=False,
        self_attention=False,
        encoder_decoder_attention=False,
        q_noise=0.0,
        qn_block_size=8,
        attention_relax=-1.0,
    ):
```

Figure 2: Multihead

```
class MERTConfigClass:
    def __init__(self):
        # Define your MERT configuration parameters here
        self.embedding_dim = 768
        self.encoder_ffn_embed_dim = 3072
        self.encoder_attention_heads = 8
        self.dropout = 0.1
        self.attention_dropout = 0.1
        self.activation_dropout = 0.1
        self.activation_fn = "relu"
        self.layer_norm_first = False
        self.attention_relax = -1.0  # Set to a positive value to enable attention relaxation
        self.encoder_layers = 6  # Number of encoder layers
        self.deepnorm = False
        self.layer_type = "transformer"  # or "conformer"
        self.subln = True
        self.checkpoint_activations = False
```

Figure 3: MERT

```
# Usage example:
args = MERTConfigClass()  # Replace with your MERT configuration
model = TransformerEncoderExtend(args)
input_tensor = torch.randn((10, 32, 768))  # Replace with your input shape
output, attn_weights = model(input_tensor)
```

Figure 4: Result

4 Summary

4.1 Data Representation:

The tensor output suggests that the model generates multi-dimensional data. Each sub-array might represent encoded features or outputs from a specific layer of your NLP model. This kind of output is typical in deep learning models where each dimension can represent different aspects of the data being processed.

4.2 Analysis of Output Values:

The values in the tensor vary both positively and negatively, indicating a diverse range of activations or responses from the model. This variation is a good sign, suggesting that your model is capturing a complex pattern in the input data.
The values do not seem excessively large or small, which implies that the model's outputs are stable and the network is likely behaving as expected.

4.3 Potential Interpretations:

Since these values are from an encoder in an NLP model, they represent some form of embeddings or transformed representations of input text. Such embeddings capture semantic and syntactic features of the input data.
In the context of NLP tasks like text classification, sentiment analysis, or language translation, these outputs would be further processed by additional model layers to make predictions or generate text.