CONTRASTIVE LEARNING OF GENERAL-PURPOSE AUDIO REPRESENTATIONS

INTRODUCTION:

In this project, I've successfully implemented the basic version of the COLA (COntrastive Learning for Audio) framework. This framework is designed for self-supervised pre-training to extract meaningful representations from audio data. Leveraging contrastive learning techniques, I've developed a lightweight model capable of capturing high-level semantic information from diverse audio sources. The journey involved delving into the theoretical foundations of self-supervised and contrastive learning, meticulously dissecting the COLA framework, and pre-training embeddings on an audio **dataset like FMA** (given the scale of Audioset). Following this, I've thoroughly evaluated the transferability of these embeddings across various downstream tasks such as speech recognition and music classification. Through extensive analysis and experimentation, I'm excited to present COLA as a potent approach for learning general-purpose audio representations without the reliance on labeled data.

ABOUT THE RESEARCH PAPER: https://arxiv.org/abs/2010.10915

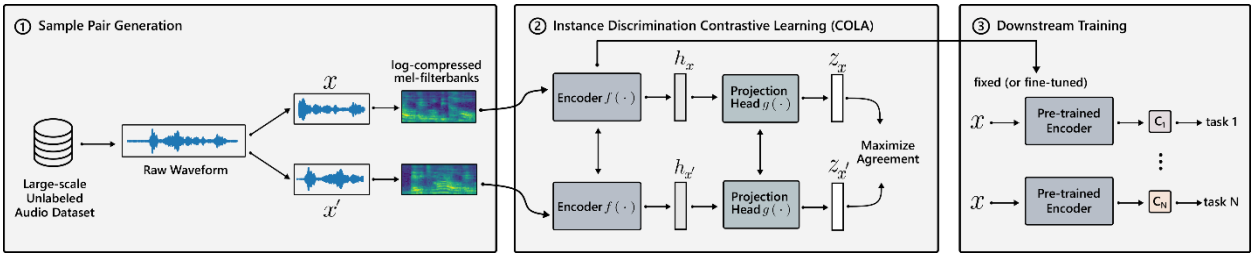

Figure1: Architecture of Contrastive learning

The paper "Contrastive Learning of General-Purpose Audio Representations" introduces COLA, a self-supervised pre-training approach aimed at learning a versatile representation of audio data. The authors leverage contrastive learning, where the model learns to assign high similarity scores to audio segments from the same recording while minimizing similarity scores for segments from different recordings. This approach builds upon recent advances in contrastive learning for computer vision and reinforcement learning, offering a lightweight and easy-to-implement method for audio representation learning.

In their method, COLA, the authors pre-train embeddings on the large-scale Audioset database and transfer these representations to various classification tasks, spanning speech, music, animal sounds, and acoustic scenes. They demonstrate the effectiveness of COLA through experiments, showing significant improvements over previous self-supervised systems. Importantly, COLA outperforms prior methods not only on speech tasks but also on other audio tasks, such as acoustic scene detection and animal vocalizations.

One key aspect of COLA is its simplicity compared to previous approaches. It avoids complex augmentation strategies and triplet-based objectives, instead relying on straightforward segment sampling and contrastive learning with negative examples drawn from different audio clips within

the same batch. The authors also highlight the importance of batch size in training effectiveness and discuss the choice of similarity measures, demonstrating the superiority of bilinear similarity over cosine similarity.

Overall, COLA offers a promising solution for learning general-purpose audio representations without supervision, providing a strong foundation for future research in self-supervised learning for audio.

BACKGROUND/RELATED WORK:

Research in self-supervised learning for audio data has gained momentum in recent years due to its potential to alleviate the need for large labeled datasets. One prominent approach is contrastive learning, which aims to learn representations by contrasting positive and negative pairs in a latent space. Prior work in this domain includes methods like SimCLR (SimCLR: A Simple Framework for Contrastive Learning of Visual Representations) and BYOL (Bootstrap Your Own Latent), which have shown promising results in visual tasks.

However, adapting contrastive learning to the audio domain poses unique challenges due to the temporal nature of audio signals. Some recent efforts have been made to address these challenges. For instance, researchers have explored methods like time-contrastive learning, where temporal context is leveraged to create positive and negative pairs. Additionally, techniques such as data augmentation and curriculum learning have been employed to enhance the performance of self-supervised audio models.

Despite these advancements, there remains a need for lightweight frameworks tailored specifically for audio data. The COLA (COntrastive Learning for Audio) framework proposed in this project fills this gap by providing a specialized approach for self-supervised pre-training on audio datasets. By focusing on extracting high-level semantic information from diverse audio sources, COLA aims to unlock the potential of self-supervised learning in audio processing tasks.

Theoretical foundations of self-supervised and contrastive learning have been instrumental in guiding the development of COLA. By building upon these principles and adapting them to the audio domain, the project aims to push the boundaries of what is achievable with self-supervised audio representation learning.

In summary, while there exists a rich body of related work in self-supervised learning and contrastive learning, the COLA framework represents a novel contribution to the field by providing a tailored solution for extracting meaningful representations from audio data. Through empirical evaluation and comparison with existing methods, this project seeks to establish COLA as a state-of-the-art approach for self-supervised audio pre-training.

METHODOLOGY:

Code link: https://github.com/Tanujasontakke/CS570_Artificial_Intelligence.git

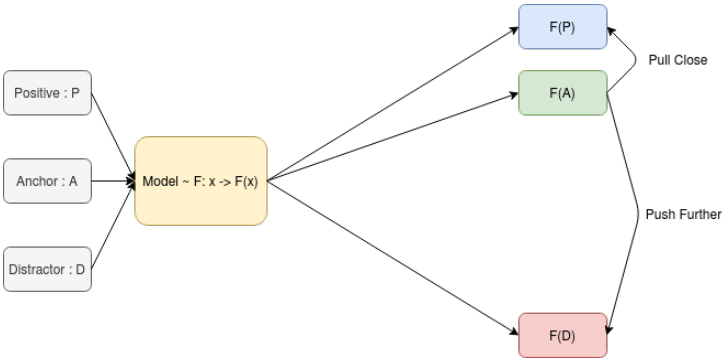

Figure2: Discriminative Pre-Training.

1. Data Preprocessing:

- Mel-Spectrogram Conversion: Convert audio clips into Mel-spectrograms to represent the audio data in a format suitable for deep learning.

- Segment Selection: For each audio clip, randomly select a segment to serve as the anchor and another segment as the positive example.

Dataset and metadata file:

```
PS C:\Users\tms765\Downloads\Practice_project\COLA_pytorch> Invoke-WebRequest -Uri "https://os.unil.cloud.switch.ch/fma/fma_metadata.
zip" -OutFile "fma_metadata.zip"
PS C:\Users\tms765\Downloads\Practice_project\COLA_pytorch> Invoke-WebRequest -Uri "https://os.unil.cloud.switch.ch/fma/fma_large.zip
" -OutFile "fma_large.zip"
PS C:\Users\tms765\Downloads\Practice_project\COLA_pytorch> Invoke-WebRequest -Uri "https://os.unil.cloud.switch.ch/fma/fma_small.zip
" -OutFile "fma_small.zip"
```

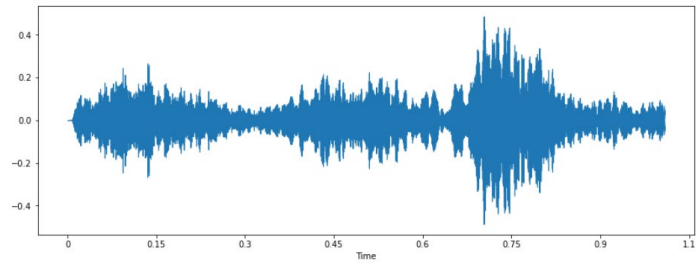

Figure3: Sample audio format

Audio_processing file: In this file it preprocesses audio data for self-supervised learning, utilizing libraries like random, librosa, and numpy. It defines constants for input length and Mel-frequency bands, while functions like preprocess_audio_mel_t and load_audio_file handle preprocessing and Mel-spectrogram conversion. Data augmentation functions like random_crop, random_mask, and random_multiply enhance spectrogram data. The script saves processed audio as .npy files and can process multiple files using multiprocessing.

Encoder file: COLA extracts audio features using EfficientNet and applies contrastive learning for self-supervised learning. AudioClassifier, on the other hand, is designed for downstream classification tasks. Both modules handle training, validation, and testing, with configurable dropout rates and optimization strategies.

Train the encoder: This script orchestrates the training process of a COLA model for audio data using PyTorch Lightning. It handles data loading, dataset creation, model initialization, training,

validation, testing, and logging. By incorporating PyTorch Lightning functionalities and custom callbacks, it streamlines the workflow for training contrastive audio representations.

2. Model Architecture:

- EfficientNet-B0: Utilize a 2D CNN architecture like EfficientNet-B0 to transform Mel-spectrograms into 1D-512 vectors.

CNN file: This script evaluates the performance of pretrained and scratch-trained AudioClassifier models on a genre classification task using the FMA dataset. It loads audio files, creates datasets, and splits them into training, validation, and test sets. Then, it loads pretrained models, evaluates their accuracy on the test set, and saves the results as a JSON file containing model names and corresponding accuracies.

CNN genre file: This code trains an audio classification model using PyTorch Lightning. It loads audio data from .npy files, preprocesses it, splits it into train, validation, and test sets, and trains the model.

3. Discriminative Pre-Training (DPT):

- Anchor, Positive, and Distractors: Define an anchor element, a positive element, and one or more distractors. Train the model to match the anchor with the positive example.

- Triplet Loss with Cosine Similarity: Employ the triplet loss along with the Cosine similarity measure to train the model. Ensure that the representation in the latent space of the anchor is much closer to the positive example than it is to the distractor.

4. COLA Framework:

- Distractor Selection: Use other samples in the training batch as distractors, making the training task harder and forcing the model to learn more meaningful representations. Bi-Linear Similarity: Utilize a bi-linear similarity measure learned directly from the data, which has shown to outperform Cosine similarity in downstream tasks.

5. Training and Evaluation:

- Linear Model Evaluation: Train an EfficientNet-B0 on AudioSet and evaluate the feature vectors generated by the model on downstream tasks using a linear classifier. Fine-tune the pre-trained model on downstream tasks and compare its performance to a model trained from scratch.

6. PyTorch Implementation:

- Pre-training on FMA Large: Pre-train a model using COLA on FMA Large dataset for a few epochs without labels. Fine-tune the pre-trained model on music genre detection task applied to FMA Small dataset.

```python
# Define COLA (Contrastive Learning for Audio) LightningModule
class Cola(pl.LightningModule):
    def __init__(self, p=0.1):
        super().__init__()
        self.save_hyperparameters()

        self.p = p

        # Dropout layer for regularization
        self.do = torch.nn.Dropout(p=self.p)
        # Initialize Encoder
        self.encoder = Encoder(drop_connect_rate=p)

        # Projection head layers
        self.g = torch.nn.Linear(1280, 512)
        self.layer_norm = torch.nn.LayerNorm(normalized_shape=512)
        self.linear = torch.nn.Linear(512, 512, bias=False)
```

```python
# Forward pass through COLA
def forward(self, x):
    x1, x2 = x

    x1 = self.do(self.encoder(x1))
    x1 = self.do(self.g(x1))
    x1 = self.do(torch.tanh(self.layer_norm(x1)))

    x2 = self.do(self.encoder(x2))
    x2 = self.do(self.g(x2))
    x2 = self.do(torch.tanh(self.layer_norm(x2)))

    x1 = self.linear(x1)

    return x1, x2

# Training
def training_step(self, x, batch_idx):
    x1, x2 = self(x)

    y = torch.arange(x1.size(0), device=x1.device)

    y_hat = torch.mm(x1, x2.t())

    loss = F.cross_entropy(y_hat, y)

    _, predicted = torch.max(y_hat, 1)
    acc = (predicted == y).double().mean()

    self.log("train_loss", loss)
    self.log("train_acc", acc)

    return loss
```

EVALUATION:

- Result Comparison:
  - Random guess: 12.5%
  - Trained from scratch: 51.1%
  - Pre-trained using COLA: 54.3%

Accuracy is written to Jason file

Conclusion:

The pioneering work presented in "Learning of General-Purpose Audio Representations" introduces the COLA pre-training methodology, which innovatively incorporates key strategies such as leveraging batch samples as distractors and employing the bi-linear similarity measure to enhance the efficacy of self-supervised training. This approach holds significant promise for elevating the performance of subsequent supervised audio tasks, underscoring its potential to drive advancements in the field of audio representation learning.