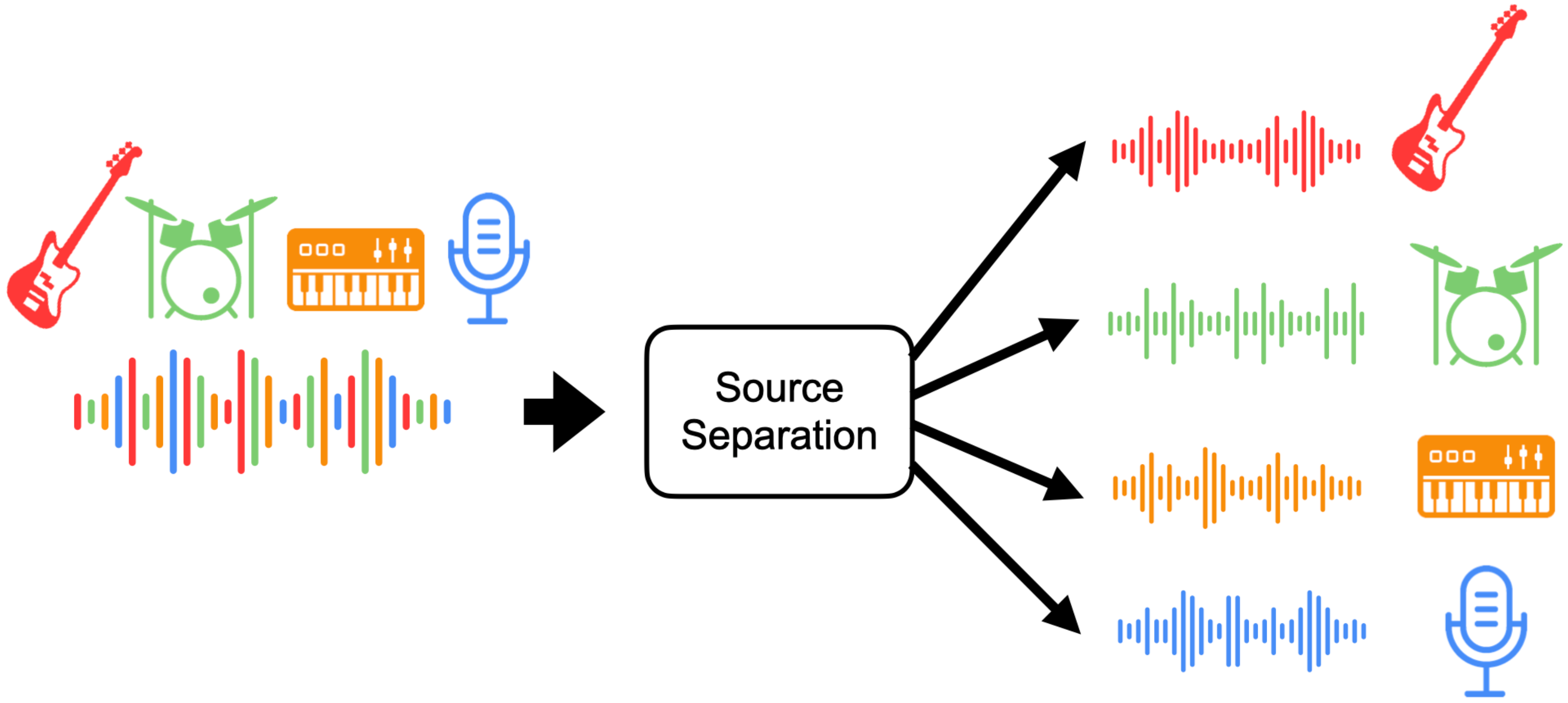
Speech Source Separation

The University of Texas at Dallas

Group 12

Task Description

The process of separating a mixture into isolated sounds from individual sources.



Motivation and Use-cases

- Audio Denoising
- Speech Separation and Isolation
- Audio Editing and Mixing
- Speech recognition

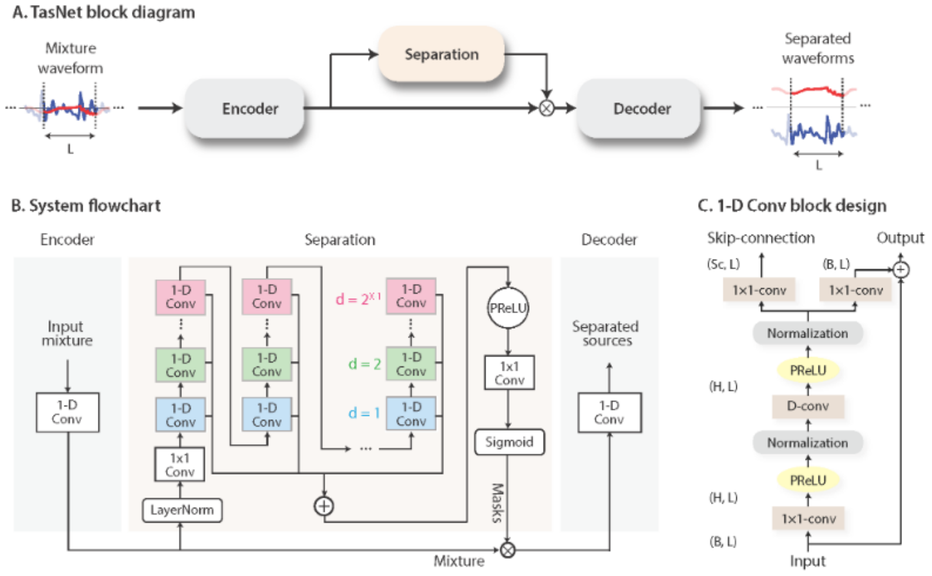
Dataset

- MUSDB18
  - 150 Full length music tracks (~10hrs) with their isolated instrument sources
  - Train: 100 tracks
    - Training Split: 87 tracks
    - Validation Split: 13 tracks
  - Test: 50 tracks

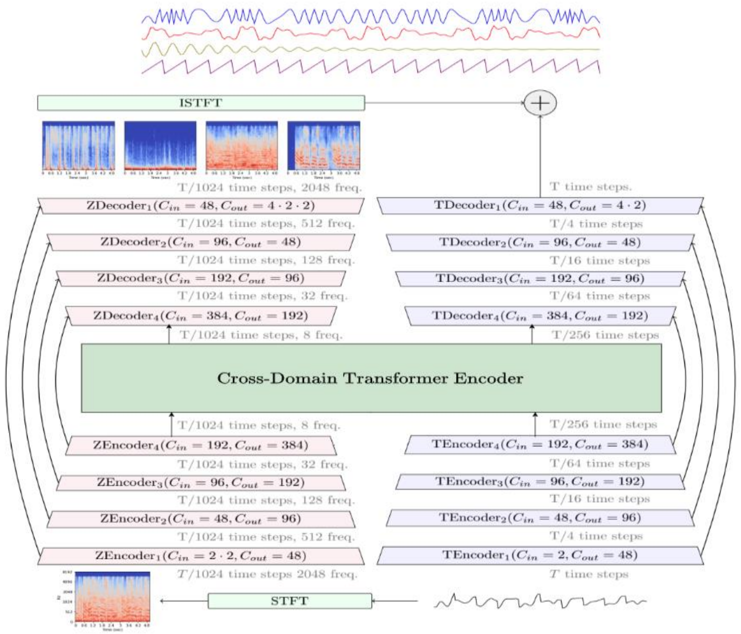
  - The dataset in total is of 5GB, however due to limitations of GPU memory and run time, we extracted random partitions from each track for training, validation and testing.

Approach:

Explore	Attempt	Modify
Explore the difference in effectiveness of various Spectrogram-based models.	Attempt to train 2 variants of Transformers. • 1st using Spectrogram Features • 2nd using MelSpectrogram Features	Modify a pretrained Conv-TasNet model (trained on Libri2Mix model) for MUSDB18 dataset.



https://arxiv.org/abs/1809.07454v3



https://github.com/facebookresearch/demucs

Implementation

- We have used Conv-TasNet BASE model and added several encoder-decoder layers to train on MUSDB18 dataset.

```python
class ConvTasNetWithTransformer(nn.Module):
    def __init__(self, num_sources = 2, hidden_size=128, num_heads=1, num_layers=1):
        super(ConvTasNetWithTransformer, self).__init__()

        self.num_sources = num_sources

        conv_tasnet = CONVTASNET_BASE_LIBRI2MIX.get_model()
        self.conv_tasnet_encoder = conv_tasnet.encoder
        self.conv_tasnet_maskGenerator = conv_tasnet.mask_generator
        self.conv_tasnet_decoder = conv_tasnet.decoder

        self.input_embed = nn.Linear(512, hidden_size)
        self.transformer = nn.TransformerEncoderLayer(hidden_size, num_heads, hidden_size, dropout=0.1)
        self.encoder = nn.TransformerEncoder(self.transformer, num_layers=num_layers)

        self.output_embed = nn.Linear(hidden_size, 512)

        for param in self.conv_tasnet_encoder.parameters():
            param.requires_grad = False

        for param in self.conv_tasnet_maskGenerator.parameters():
            param.requires_grad = False
```

Implementation

- Apart from this we also implemented a MelSpectogram based transformer model by adding encoder-decoder layers and trained on MUSDB18 dataset to analyze results.

```
class TransformerModel(nn.Module):
    def __init__(self, d_model = 512, num_heads = 4, num_layers = 4, dropout = 0.1):
        super().__init__()

        # input embedding layer
        self.input_embed = nn.Linear(128, d_model)

        # transformer encoder layers
        encoder_layers = nn.TransformerEncoderLayer(d_model=d_model, nhead=num_heads, dropout=dropout)
        self.encoder = nn.TransformerEncoder(encoder_layers, num_layers=num_layers)

        decoder_layers = nn.TransformerDecoderLayer(d_model=d_model, nhead = num_heads, dropout=dropout)
        self.decoder = nn.TransformerDecoder(decoder_layers, num_layers = num_layers//2)

        # output embedding layer
        self.output_embed = nn.Linear(d_model, 202)

        self.transform = MelSpectrogram(sample_rate = 44100, n_fft = 200, n_mels = 128)
        self.invtransform = InverseSpectrogram(n_fft = 200)
```

Evaluation Metric

- The Signal-to-Distortion Ratio (SDR) is a metric used to evaluate the quality of a separated audio signal by measuring the similarity between the separated signal and the true source signal, while accounting for distortion caused by the separation process.

- Higher SDR values indicate better separation performance.

```
def calculate_sdr(predicted_output, ground_truth):
    """
    Calculates the Signal-to-Distortion Ratio (SDR) metric between a predicted output and its corresponding ground truth.

    Args:
    predicted_output: A numpy array of shape (number of channels, number of frames, number of bins).
    ground_truth: A numpy array of shape (number of channels, number of frames, number of bins).

    Returns:
    sdr: A scalar representing the SDR value between predicted_output and ground_truth.
    """
    eps = np.finfo(np.float32).eps  # To avoid division by zero errors
    num_channels = predicted_output.shape[0]
    sdr_sum = 0

#     print(predicted_output.shape, ground_truth.shape)

    for c in range(num_channels):
        # Compute the power of the true source signal
        true_source_power = np.sum(ground_truth[c]**2)

        # Compute the scalar product between true source signal and predicted signal
        true_pred_scalar = np.sum(ground_truth[c] * predicted_output[c])

        # Compute the SDR for this channel
        sdr = 10 * np.log10(true_source_power / (np.sum(ground_truth[c]**2) - true_pred_scalar + eps) + eps)
        if not math.isnan(sdr):
            sdr_sum += sdr

    # Compute the average SDR across all channels
    sdr = sdr_sum / num_channels

    return -sdr
```

Experiment 1

- Getting familiar with the dataset and a simple Transformer-based Model using extracted Spectrogram features.
- Transformer Specifications:
  - 4 Attention Heads, 4 Encoder Layers, 2 Decoder Layers
- Results:

```
Training MSE: 0.008412279839500447
Validation MSE: 0.008107607452464955
Testing MSE: 0.006638711600488425
```

```
Training SDR: 4.369476915549454
Validation SDR: 2.867712731368163
Testing SDR: 3.4226250170195756
```

Experiment 2

- Modifying the Transformer by using Mel-Spectrogram features instead of Spectrogram features.
- Transformer Specifications:
  - 4 Attention Heads, 4 Encoder Layers, 2 Decoder Layers
- Results:

```
Training MSE: 0.008429285858503797
Validation MSE: 0.008108512631484441
Testing MSE: 0.006638769670389593
```

```
Training SDR: 4.369410692386922
Validation SDR: 2.940788830961262
Testing SDR: 3.505986264104757
```

Experiment 3

- Modifying a Conv-Tas-Net model trained on Libri2Mix dataset for music source separation to focus on Vocals Source Separation using MUSDB18 dataset.
- Modification Specifications:
  - 1 Attention Head, 1 Encoder Layer, 1 Decoder Layer added between the Mask Generation Module and the Decoder Module of ConvTasNet.
- Results:

```
Training MSE: 0.027541908520189198
Validation MSE: 0.026301669755152295
Testing MSE: 0.03057378761470318
```

```
Training SDR: 4.347351231621799
Validation SDR: 3.3966928381526555
Testing SDR: 4.090461655007675
```

Result and Analysis

- Transformer trained using Mel Spectrogram features provide a better performance than the transformer trained using Spectrogram based features.
- Can be attributed to the ability of Mel Spectrogram features to provide a more discriminative feature space and reduce the dimensionality of the input space.
- The modified ConvTasNet model provides a better Generalization on unseen data.

Relevant Papers

- HYBRID TRANSFORMERS FOR MUSIC SOURCE SEPARATION [current state of the art]
- AN EFFICIENT ENCODER-DECODER ARCHITECTURE WITH TOP-DOWN ATTENTION FOR SPEECH SEPARATION
- On Using Transformers for Speech-Separation [LibriMix Dataset]
- Music Source Separation with Band-split RNN
- https://arxiv.org/abs/1809.07454

QUESTION ?