Audio Analysis Using Gramian Angular Field (GAF) and FAST

GitHub: https://github.com/dallasai/csce5222-project

People

- Andrew Fausak (andrewfausak@my.unt.edu)
- Andy Fausak (andyfausak@my.unt.edu)
- Andre Sharp (asharpasharp99@gmail.com)
- Mica Haney (micahaney@my.unt.edu)

Introduction

Motivation

Motivation to create this work is based on review of a pollution analysis method that utilizes GAF to provide images representing normalized discrete readings in summary, thereby enabling tools to perform similarity tests to qualify conditions needing to be monitored. Having a solution whereby similarity of works can be compared is useful and readily consumed by AI tools needing features to isolate desired inferences. Experimentation is required to determine resolution needed to uniquely identify or classify a work using NLP methods.

Converting data from one domain of information representation to another can provide useful information that would not otherwise be noticeable, such as in spectrograms. People have hidden "text" and images in sound files that can be identified only by using specific graphical tools. Using graphical processing techniques to find hidden information in audio is an interesting topic with the potential to improve the current methods of working with audio.

Significance

Having a means to identify works quickly in terms of classification and/or origin is very useful. Moreover, this may provide alternate means of tagging documents for referencing. With this technique more information may be extracted from the audio, which will allow for better handling of audio tasks. If the proposed method does extract information from sound that is not usually or easily picked up by current audio processing techniques, then adding such features and procedures could notably improve results involving audio, particularly in machine learning models.

Objectives

The overall objectives of this project are to A) extract features from audio files that allows them to be converted to an image format, B) extract secondary features from the converted images that are typical of image-based feature engineering, and C) run models with the converted images and secondary features.

Ideally we would like to apply these above objectives in two fashions. First we would like to compare results of the converted features to results obtained by the same or a very similar architecture that handle the original audio data. Second, we would like to attempt to create a speech recognition tool based on these methods.

Features

The completed project will work with three main features, GADF, GASF, and spectrograms.

Spectrograms are visual representations of the frequencies of an audio signal. Currently our implementation of spectrogram generation uses the pre-built function in the matplotlib.pyplot library.

GADF and GASF are types of Gramian Angular Fields (GAFs). GAFs are created from a time series, and each point in the image represents the correlation between two points within the time series where the time series sits on both the x and y axis. Gramian Angular Difference Fields (GADFs) use the sine function to compute values, while Gramian Summation Fields (GASFs) use the cosine function.

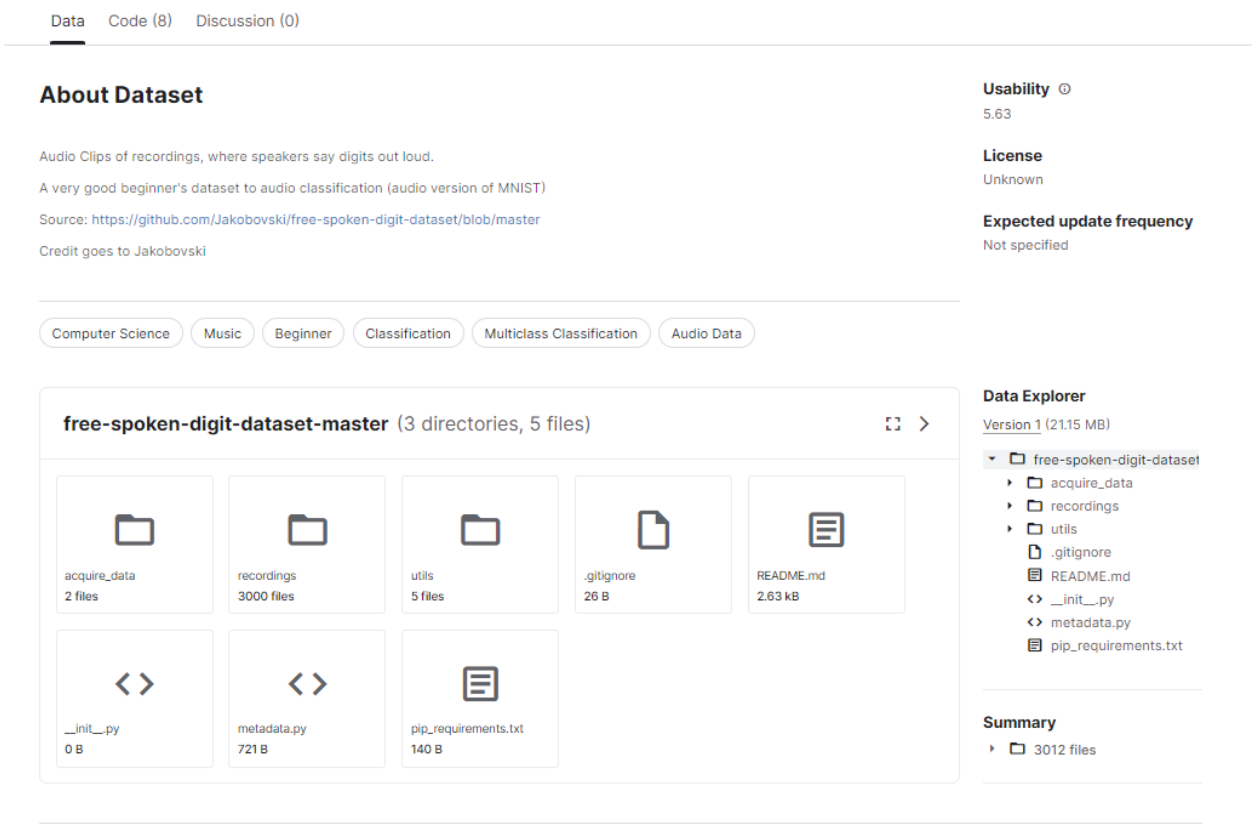
Related Work

Shah, S. K., Tariq, Z., Lee, J., & Lee, Y. (2021). Event-Driven Deep Learning for Edge Intelligence (EDL-EI). *Sensors*, *21*(18), 6023.

Wang, Z., & Oates, T. (2015, June). Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Dataset

The Audio MNIST dataset consists of audio recordings of individuals saying numbers in the zero to nine range in English. There are six individuals speaking, with 50 recordings of each digit being spoken for each individual for a total of 3,000 recordings. These recordings are .wav files.



Design of Features

The initial choice of features was to use GAFs, but was quickly expanded to look at using more features. The GAF was split into two, with two different computation methods being used to produce the GAF images for comparison. Spectrograms were chosen as supporting image features to feed into a CNN in comparison to or alongside the GAF images.

One of the biggest design considerations for the features was actually what size to save the GAF and spectrogram images at. Audio files are information-dense, so the higher file sizes allow for more data points and therefore finer granularity of data, as well as less data loss. However, larger image sizes pose a multitude of issues. The main issues in concern were the time it would take for scripts to process the images and the memory space required to handle that much data. Ultimately it was decided to save larger files to keep as much data from the original audio as possible, which should improve results.

Implementation

The GAF file generation code uses custom code to generate the GAF values, with matplotlib, scipy, pandas, and numpy as supporting libraries for loading the audio signal and plotting the GAF values.

```
class GAF:

    def __init__(self):
        pass

    def __call__(self, serie, gasf=True):
        min_ = np.amin(serie)
        max_ = np.amax(serie)
        scaled_serie = (2 * serie - max_ - min_) / (max_ - min_)

        scaled_serie = np.where(scaled_serie >= 1., 1., scaled_serie)
        scaled_serie = np.where(scaled_serie <= -1., -1., scaled_serie)

        phi = np.arccos(scaled_serie)
        r = np.linspace(0, 1, len(scaled_serie))

        gaf = tabulate(phi, phi, cos_sum if gasf else sin_sum)

        return (gaf, phi, r, scaled_serie)
```

For the MNIST classifier, each audio file had a spectrogram created from it via the raw audio values and framerate using pylab. Each spectrogram was saved as a .png file of the dimensions 256 by 256. Each spectrogram was normalized to the 0 to 1 range. Some of the spectrograms were randomly flipped along an axis, and some were randomly rotated. The spectrograms that were rotated or flipped were not replaced with the original scaled spectrogram.

```
# Function to prepare our datasets for modelling
def prepare(ds, augment=False):
    # Define our one transformation
    rescale = tf.keras.Sequential([tf.keras.layers.experimental.preprocessing.Rescaling(1./255)])
    flip_and_rotate = tf.keras.Sequential([
        tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
        tf.keras.layers.experimental.preprocessing.RandomRotation(0.2)
    ])

    # Apply rescale to both datasets and augmentation only to training
    ds = ds.map(lambda x, y: (rescale(x, training=True), y))
    if augment: ds = ds.map(lambda x, y: (flip_and_rotate(x, training=True), y))
    return ds

train_dataset = prepare(train_dataset, augment=False)
valid_dataset = prepare(valid_dataset, augment=False)
```

Preliminary Results

The GAF file generation code is up and running, and 180,000 of the Audio MNIST files have been processed.

The MNIST classifier obtained a top validation accuracy of 96.17%, with a training accuracy of 99.42% for that epoch. The classifier was trained with sparse categorical cross entropy loss, root mean squared propagation optimization, and with a softmax output activation. There are no testing scores at this time.

```
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(N_CLASSES, activation='softmax'))

# Compile model
model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=tf.keras.optimizers.RMSprop(),
    metrics=['accuracy'],
)

# Train model for 10 epochs, capture the history
history = model.fit(train_dataset, epochs=10, validation_data=valid_dataset)
```

```
Epoch 1/10
75/75 [==============================] - 400s 5s/step - loss: 0.6488 - accuracy: 0.8067 - val_loss: 3.3265 - val_accuracy: 0.1417
Epoch 2/10
75/75 [==============================] - 371s 5s/step - loss: 0.2154 - accuracy: 0.9413 - val_loss: 3.7130 - val_accuracy: 0.1183
Epoch 3/10
75/75 [==============================] - 382s 5s/step - loss: 0.1324 - accuracy: 0.9708 - val_loss: 3.2742 - val_accuracy: 0.2283
Epoch 4/10
75/75 [==============================] - 375s 5s/step - loss: 0.0933 - accuracy: 0.9783 - val_loss: 2.6962 - val_accuracy: 0.3650
Epoch 5/10
75/75 [==============================] - 381s 5s/step - loss: 0.0646 - accuracy: 0.9867 - val_loss: 1.1913 - val_accuracy: 0.6117
Epoch 6/10
75/75 [==============================] - 382s 5s/step - loss: 0.0377 - accuracy: 0.9933 - val_loss: 0.4864 - val_accuracy: 0.8517
Epoch 7/10
75/75 [==============================] - 384s 5s/step - loss: 0.0313 - accuracy: 0.9942 - val_loss: 0.1887 - val_accuracy: 0.9433
Epoch 8/10
75/75 [==============================] - 386s 5s/step - loss: 0.0266 - accuracy: 0.9942 - val_loss: 0.1410 - val_accuracy: 0.9617
Epoch 9/10
75/75 [==============================] - 376s 5s/step - loss: 0.0190 - accuracy: 0.9971 - val_loss: 0.2720 - val_accuracy: 0.9250
Epoch 10/10
75/75 [==============================] - 376s 5s/step - loss: 0.0271 - accuracy: 0.9946 - val_loss: 0.2764 - val_accuracy: 0.9033
```

Project Management

Communication was done over Discord. Meetings were arranged as-needed, and occurred both in-person and virtually over Zoom.

Implementation Status Report

Work Completed

Description

For the Audio MNIST dataset, the spectrograms, GADFs, and GASFs have been successfully extracted from the files.

The Audio MNIST CNN classifier had been completed.

Responsibility

Andrew Fausak

Create the GAF generation code. Generate GAF files.

Andy Fausak

Create the GAF generation code. Generate GAF files.

Andre Sharp

Created the CNN classifier trained on the Audio MNIST dataset.

Mica Haney

Wrote the update report. Brainstorming.

Contributions

25% Andrew Fausak

25% Andy Fausak

25% Andre Sharp

25% Mica Haney

Work to be Completed

Description

We intend to create code that converts the GAF files back into the audio format so that a comparison can be done to evaluate how much data loss occurs in the conversion of audio to GAF files.

The classifier needs to be properly trained on the complete dataset, with various experiments done to compare how the different features and combinations of features affect scores.

If time permits, we would like to attempt dimensionality reduction techniques such as PCA and few-shot learning. Dimensionality reduction has the potential to allow a reduction in learning times with minimal data loss for the large images, which is of interest to us and the machine learning community. As few-shot learning looks at data similarities for predictions, we feel that this technique has the potential to broaden the applications of the base idea of the project.

Responsibility

Andrew Fausak

Convert GAF files back to audio. Analyze the information loss.

Andy Fausak

Convert GAF files back to audio. Analyze the information loss.

Andre Sharp

Run experiments. Analyze the results of the experiments. Write the final report.

Mica Haney

Train the classifier on the whole dataset. Run experiments. Analyze the results of the experiments. Write the final report.

Issues and Concerns

The main issues come from working with the large GAF images. They take a long time to generate, they require a lot of computational power to generate, and due to library crashes caused by the image size efforts in creating the GAF files is about 1/5th the speed it would be if working with smaller images. We have concerns about fully completing the project due to time concerns purely related to how long it takes to work with large images.

References

Choon, A. (2020). Audio MNIST (1) [https://www.kaggle.com/datasets/alanchn31/free-spoken-digits]. kaggle

Faouzi, J., & Janati, H. (2020). pyts: A Python Package for Time Series Classification. *J. Mach. Learn. Res.*, *21*, 46-1.

Matplotlib.pyplot. Matplotlib documentation. November, 2022, from https://matplotlib.org/stable/api/pyplot_summary.html

Shah, S. K., Tariq, Z., Lee, J., & Lee, Y. (2021). Event-Driven Deep Learning for Edge Intelligence (EDL-EI). *Sensors*, *21*(18), 6023.

Spectrogram. (2022, August 29). In *Wikipedia*. https://en.wikipedia.org/wiki/Spectrogram

Wang, Z., & Oates, T. (2015, June). Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.