Deep Learnin 046211, Fall 2022

Adaptive STFT: Classify Music Genres with a learnable spectrogram layer

Final Project

Rotem Idelson 206056053 and Noam Elata 204806095
22/01/2022

Introduction

Expanding on existing application of image processing networks to audio using STFT, we propose an adaptive STFT layer that learns the best DFT kernel and window for the application.

The task of audio-processing using neural networks has proven to be a difficult task, even for the state-of-the-art 1-D processing network. The use of STFT to transform an audio-processing challenge into an image-processing challenge enables the use of better and stronger image-processing networks. An example of such uses can be found in following paper: Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks[1]. Because STFT is in essence a feature extractor, base on applying 1-D convolutions, we propose a method to simplify the translation of 1-D sequences into 2-D images. We will also improve the vanilla STFT by learning task-specific STFT window coefficients and DFT kernel coefficients, using pytorch's build in capabilities.

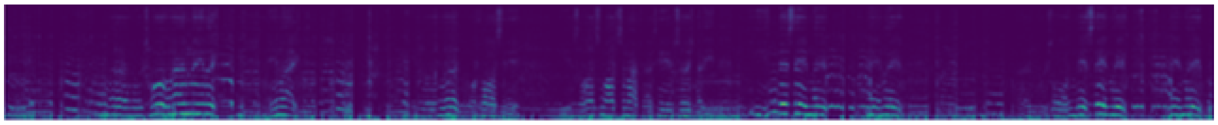
Figure 1: An Example of a STFT spectrogram produced by our module

In this project, we implemented a toy example of an audio-processing problem - music genre classification - to show the advantages of Ada-STFT. We have tried to classify the genre of an audio part from the GTZAN dataset[2]

The music classification task is based on a project done in the technion in 2021, and can be found on Github[3].

A link to our open source implementation can be found on Github.

[1]https://arxiv.org/abs/1706.07156
[2]https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification/code
[3]https://github.com/omercohen7640/MusicGenreClassifier

Previous work

We have based the project on a project done last semester on this course, MusicGenre-Classifier. There are multiple previous works on using STFT spectrograms for music genre classification:

- Music Genre Classification using Machine Learning Techniques[4] - Music classification on a large data-set with 7 classes, using VGG architecture.
- Music Genres Classification using Deep learning techniques[5] - Music classification on a GTZAN data-set with mobilenet, using both STFT spectrograms and wavelet images.
- Music Genre Classification Using CNN[6] - Music classification on a GTZAN data-set with a MLP (despite the name), using many visual features extracted from the data.

As we can see from the multitude of research projects and from the GTZAN data-set source it self that exists in both audio and spectrogram form [7], using STFT spectrograms is quite common. Most implementations using GTZAN and STFT features alone achieved final test-set accuracy of around 50-60%, so we know that the task is relatively difficult, and therefore has room for improvement using our Ada-STFT idea.

We have only found one work on the optimization of STFT itself:

- Optimizing Short-Time Fourier Transform Parameters via Gradient Descent[8]

This paper proposes methods for optimizing the window length and hop-size of the STFT calculation. These STFT parameters are more similar to most neural network architecture hyper-parameters than the window and kernel coefficients, therefore these parameters are not as easy to derive explicitly. We have decided to focus on the window and kernel coefficients, as they are simple for the network to learn, while in essence simplifying the classification method to a 1-D sequential data problem.

STFT Layer implementation

The Short Time Fourier Transform - STFT - of a discrete signal is defined as[9]:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{1}$$

As can be seen from the equation, STFT is in essence a 1-D convolution of input signal $x[n]$ with a kernel defined by $w[n-m]e^{-j\omega n}$. $w[n]$ is a predefined window function, and $e^{-j\omega n}$

[4]https://arxiv.org/abs/1804.01149

[5]https://www.analyticsvidhya.com/blog/2021/06/music-genres-classification-using-deep-learning-techni

[6]https://blog.clairvoyantsoft.com/music-genre-classification-using-cnn-ef9461553726

[7]https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification/code

[8]https://arxiv.org/abs/2010.15049

[9]https://en.wikipedia.org/wiki/Short-time_Fourier_transform

are the DFT coefficients. In our pytorch implementation of an STFT layer, we calculate the kernels $w[n-m]e^{-j\omega n}$ and convolve the input signal with these kernels to generate the spectrogram. To make our window function and window function coefficient and DFT coefficients learnable parameters, we save them as `torch.nn.Parameter` in the STFT layer. These parameters are initialized to vanilla STFT (as is formulated in equation 1), and can be set to `requires_grad = True` at any point in the training process to optimize them. In this project, we did not try to optimize over other STFT hyper-parameters like the window length or hop-size, although some of these hyper-parameters can be implicitly learned using our existing optimizations (for example, shorted window length can be achieved by zeroing the window length.

After preforming the STFT itself, it is common practice to modify the spectrogram to Mel scale and apply a log function to the result. Our STFT layer also supports these operations (these operation do not have learnable parameters).

Music Genre Classifier Implementation

0.1 Data-Set

We have use the GTZAN data-set[10], a labeled data-set that contains a thousand 30 seconds long audio samples from 10 different music genres. In order to enlarge the training data-set, the data-loader splits each 30 second audio file into smaller parts; For training, a random sample of the 30 seconds is fed into the classifier to add variability to the data. For evaluation (validation or test) the 30 second are split into equal parts that are evaluated simultaneously, and a voting system between the results determines the final prediction (by the form of major vote). We have also used augmentations to prevent over-fitting.

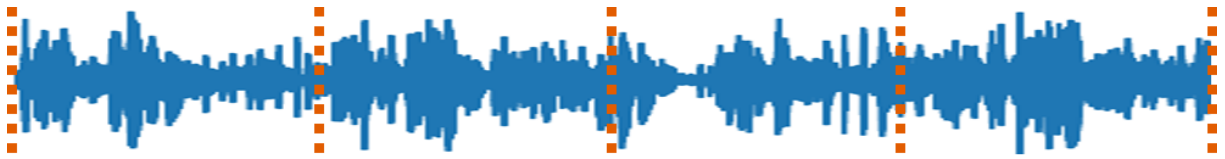

Figure 2: Sample is split in to parts to enlarge the data-set

0.2 Network Architecture

We have based our architecture on the classifier architecture proposed by the MusicGenreClassifier[11]. The classifier is a model composed of an STFT layer and a Resnet image classifier. The classifier has additional function to change the data shape (for example, duplicating the spectrogram to Resnet's 3 input channels). We train with resnet18, the smaller Resnet available.

[10]https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification/code
[11]https://github.com/omercohen7640/MusicGenreClassifier

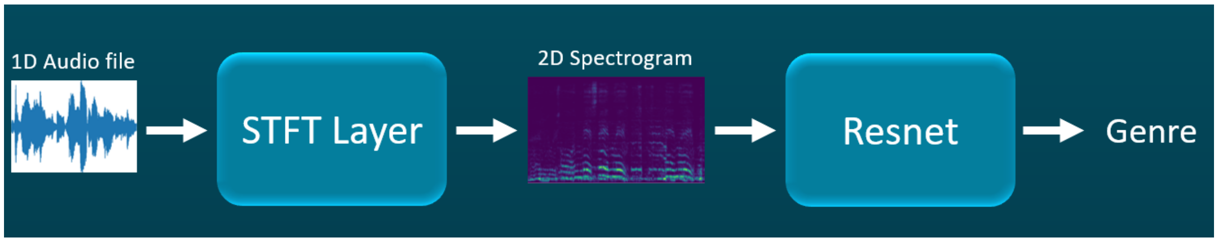

Figure 3: An overview of the proposed architecture

We have also conducted tests using 3 parallel STFT layers for each of Resnet's 3 input channels.

The network was trained using Cross Entrophy Loss.

0.3 Hyper-Parameter search

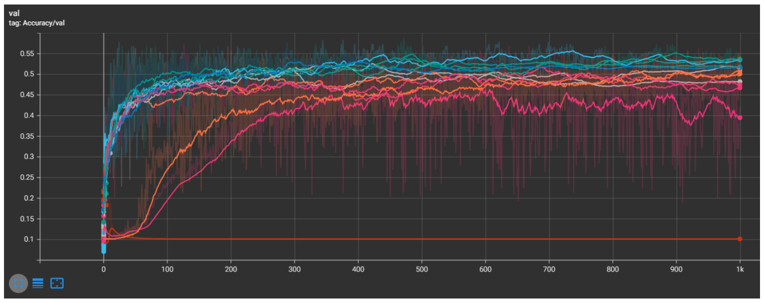

Figure 4: Optuna optimization accuracy results

At first, we have tried to imitate the hyper-parameters used on the Music Genre Classification project. After many trials with different hyper-parameters, and in order to further improve our results, we decided to use Optuna python package to preform a hyper-parameter search over some of the hyper-parameters such as:

- optimization algorithm
- learning rate
- number of parts to split the data to
- whether to use augmentations
- exponential decay scheduler's gamma parameter

We also Used Median Pruning and set the trial number to 100.

As you can see in figure 4, most of the trials where early pruned, and most of them converge around 0.53 validation accuracy score.

0.4 Testing the performance of the Ada-STFT module

0.4.1 Selected hyper-parameters

We selected hyper-parameters using an Optuna trial with good performance:

- optimization algorithm: AdamW
- learning rate: 004217059205126061
- exponential decay scheduler's gamma parameter: 0.9974547047513221
- number of parts to split the data to: 12
- whether to use augmentations: 1 (True)

0.4.2 Trial configurations

Using the best hyper-parameters from the Optuna search, we will compare the performance of networks with the following 7 trials:

1. basic run, with no STFT learning
2. learning the STFT window coefficients
3. learning the STFT DFT kernel coefficients
4. learning both the DFT kernel coefficients and window coefficients
5. learning 3 different STFT: window coefficients only
6. learning 3 different STFT: DFT kernel coefficients
7. learning 3 different STFT: both DFT kernel coefficients and window coefficients

Results

0.5 Train Loss

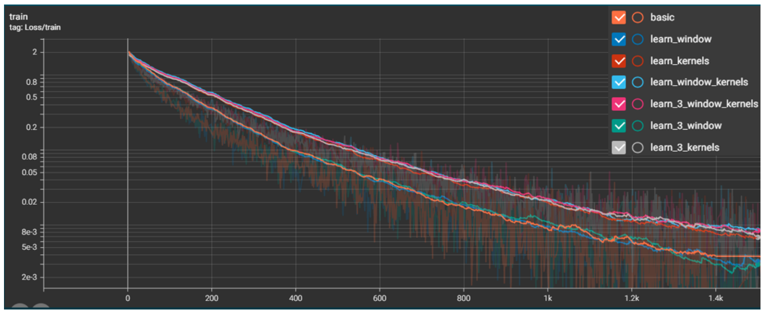

Figure 5: The train loss progress (logarithmic)

0.6 Validation Accuracy

As we can see, out of the following 3 combinations:

1. learning the STFT window coefficients
2. learning the STFT DFT kernel coefficients
3. learning both the DFT kernel coefficients and window coefficients

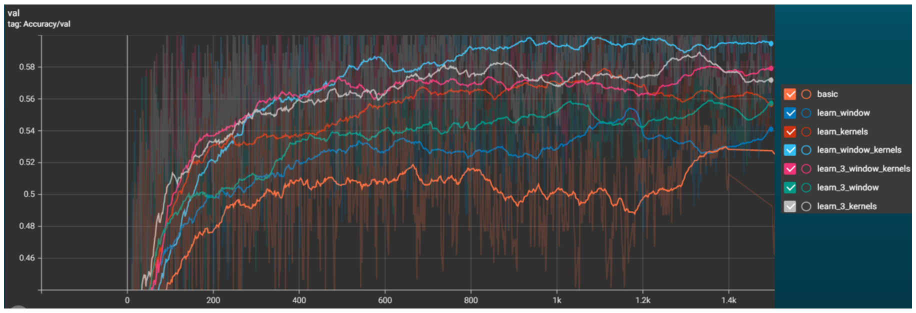

Figure 6: The validation accuracy progress

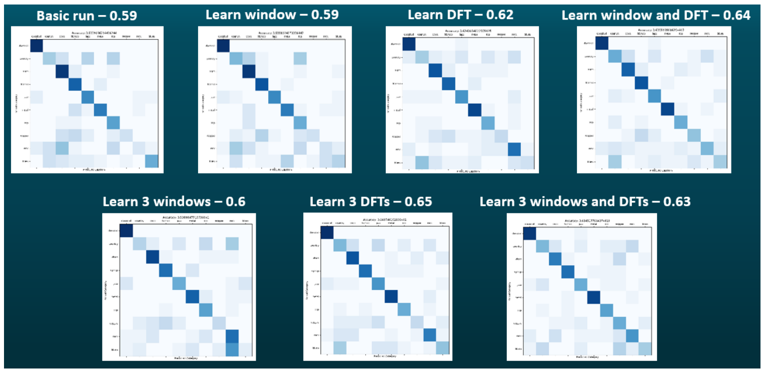

Figure 7: The best validation confusion matrices per trial

It appears that learning both the DFT kernel coefficients and window coefficients together has the best performance. Surprisingly, it seems that learning 3 different STFT modules (one for each of Resnet's input channels) does not improve the performance over learning 1 STFT module; It performs slightly better or slightly worse, depending on the trial configuration and chance. (Note: the 0.65% score of the 'Learn 3 DFT' trial is an outlier from the beginning of the run, and therefore ignored).

0.7 Test accuracy score of the best model

The best model seems to be the one who learned 1 STFT; both DFT's and window coefficients. It's best validation accuracy was 0.64%. These result are on par or better than most other implementations of similar classifiers. Due to a bug, the best checkpoint was corrupted, therefore the test accuracy has been calculated using a different checkpoint from a previous epoch, with high validation score (0.63%) and a similar conclusion matrix. The results are shown in Figure 8.

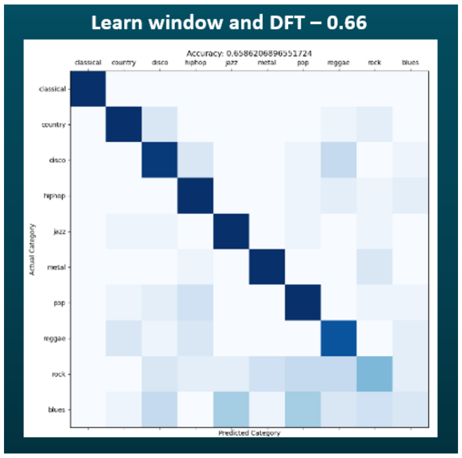

Figure 8: Test confusion matrix

0.8 Learned windows and DFT kernels

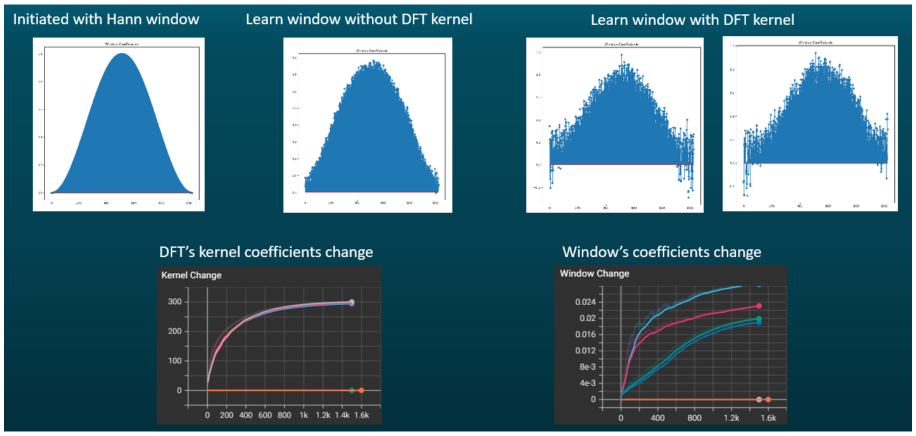

Figure 9: Examples of learned windows (top), graphs of coefficient change (bottom)

As shown in figure 9, trying to learn the window without the DFT's kernel changes the window coefficient only slightly, probably because Hann window is a strong prior. However, learning the DFT coefficients together with the window promotes it to change and lose the Gaussian shape. The window appears flatter, and sometimes a slightly shifted from it's initial symmetric axis. In contrast to this, in the trials the change in the DFT coefficients is not effected by whether the STFT is learning the window coefficients or not.

Future Work and Conclusions

0.9 Conclusions

From our results, we conclude that the application of Ada-STFT might be useful in any task where a spectrogram would be used to transform an audio-processing task into an image-processing task. This method can push the accuracy by a few extra percents, as well as grant insight into the network's chosen feature.

0.10 Future Work

1. In order to further explore the capabilities of Ada-STFT, we believe it should be tested on a bigger data-set, with more training resources. We would add more epochs to the trials to allow the networks to fully converge (something that might lead to over-fitting on our small data-set).

2. We believe Ada-STFT would better show its potential with an extensive hyper-parameter tuning. We would also recommend Optuna trials over all the many hyper-parameters we did not get to test, such as the window size of the STFT or the augmentation's hyper-parameters.

3. It would be interesting to try and apply Ada-STFT (combining with a well known good 2D network) to other sequential data domain problems, where spectrum's are traditionally used, and test it's performance.

4. It might also be worthwhile to test applying an activation function between the Ada-STFT and the image-processing module. A non-linearity that that does not remove information from the signal (like LReLU) could cause the Ada-STFT's to function better as a neural-network layer.

5. We believe it would be interesting to further analyze the learned window and kernel coefficients. It would be interesting to understand the mathematical meaning of the learned features of the window in the Fourier domain and the space that is spanned by the kernel coefficients.