Deep Learning project final report: Single label instrument classifier

By:

Ori Hefetz - 208783100

Ofek Dahan – 313598385

Introduction

In this project, our goal was to identify multiple musical instruments from a single audio file.

We wanted to use our innovative idea of multiple spectrograms generated for each sound sample, and passing each of them through a different convolutional neural network.

These CNNs would then be combined into a fully connected layer, which will aggregate the results from each network and derive conclusions for each possible instruments we want to detect.

We have little experience with signal processing from different courses, but this is our first attempt at utilizing those tools for a real life application. This is also our first model planned and built, not including homework in this very course.

By the end of the project, we were humbled and found many obstacles. We decided to move into a single label classification, because we could not converge into a high enough accuracy.

We also ended up stripping up the model to preform better calculations on specific data. We reduced the amount of spectrograms, using the ones were most influential on the final differentiation between different instruments.

We also attempted to use a more niche concept, of wavelet transformations - these are transformation kernels designed for specific types of signals, to better the results of the transformation. We ran into many problems on that end, and decided to let this idea go at the end of the day.
We believe that if we had more time and knowledge, this could be an interesting and useful concept to explore.

Our end result left us somewhat unsatisfied. But going from the expected accuracy of random guessing (~9%) to 15-20 % in most our runs, and ending up at 51% accuracy, we can still call this project a success.

Method

The short time Fourier transform:
The STFT, is a tool in digital processing that allows us to produce an image of the signal's frequencies over time.

For the transform, we choose a time window W of certain width. And then preform a Fourier transform for each position along the grid for the window, on the windowed signal.

Continuous STFT

$$STFT = X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t-\tau) e^{-i\omega t} dt$$

Discrete STFT

$$STFT = X(m, \omega) = \sum_{n=-\infty}^{\infty} x_n w_{n-m} e^{-i\omega t_n}$$

This results in the image of every frequency snapshot, for each of the windows.

The issue with STFT, is that for a shorter window, you may increase the accuracy in the time domain, but you lose on accuracy in the frequency domain- by the nature of a shorter signal being transformed.

Our solution: create multiple variants of the STFT: three possible time windows, and three possible frequency scales. Doing so resulted in 9 different spectrograms, each containing different information about the signal.

Now, that we have all this information, we wanted to process it in the most efficient neural network.

We decided that for each spectrogram, there would be a different convolution path. This way, the convolutions can focus on the most important features from each time and frequency windows. Finally, to merge these features and process a classification, we would converge the CNNs into a FC layer, which will reach a final decision.

We also implemented various methods of data processing, from inserting empty or noisy samples, to combining and shifting the samples in different ways.

Eventually we recognized that the dataset may already be noisy, and upon inspection we realized combining signals for multi label training samples was making such noisy data, we couldn't get results.

Thus we focused on improving the accuracy of a single label classifier.

IRMAS dataset:

IRMAS dataset is an audio dataset of approximately 6700 training samples and 2900 testing samples.

Each sample is one of 11 classes:

```
LABELS = ["cello", "clarinet", "flute", "acoustic_guitar", "organ", "piano", "saxophone", "trumpet", "violin", "voice", "other"]
```

Training set:

This set, was divided into 80-20 (train validation).

The set is of 3 seconds audio samples, multi instrument (1-3), single label.

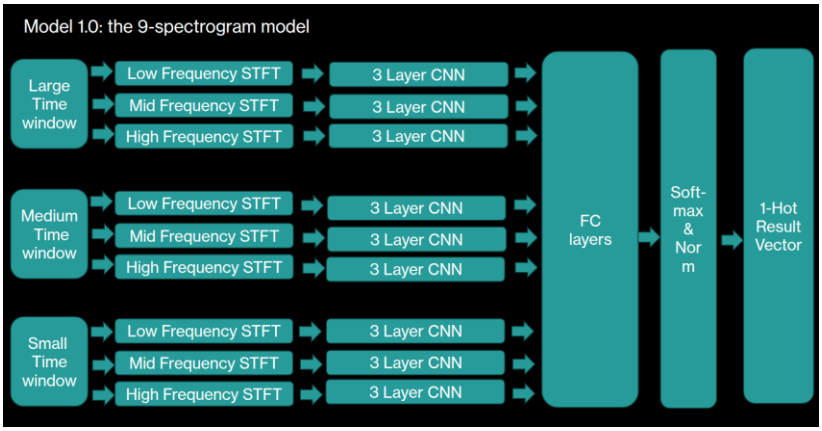
The label would be of the most dominant instrument out of the 3.

Testing set:

The set is of 20 seconds audio samples, multi instrument (1-3), multi label.

Experiments and results

Model 1:



We have started with the model above, 9 spectrograms, 3 in time, and 3 in frequency.

This was basically our renovation with this project. Use signal processing methods in deep learning.

So we had 3 windows in time to learn different "time" related information (comparing drum hit to a violin strum), and we had 3 windows in the frequency domain to learn "frequency" related information (vocal compare to violin).

We have seen that network was too big, but not deep enough, which caused really low accuracy.

Best accuracy was around 15%.

Multilabel training data:

So we thought to ourselves, maybe to learn multilabel information, we need multilabel datasets.

Reminder – The IRMAS training dataset is multi instrument – single label.

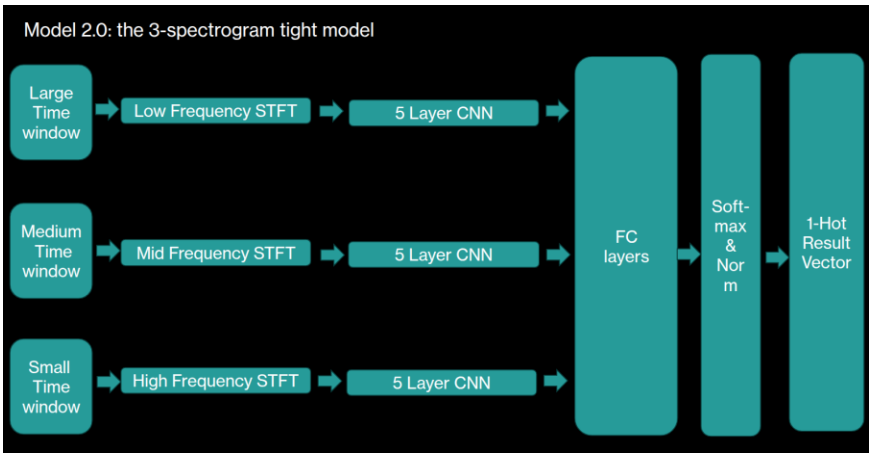
We then mixed up samples, in order to get multilabel samples, and also to increase our dataset size.

Eventually, we figured out, that having 6-9 instruments playing simultaneously was too much noise, so we decided to let this method go.

Switch to single label classifying:

After consulting Lior, we got to the conclusion, that our dataset and our resources are not enough to classify multilabel properly, so we decided to switch to classify single label instrument instead.

Model 2:



To get a deeper network, we had to decrease our net's width. In order to do so, we decided to remove some of the spectrograms, and keep also 3 of them. That factor, allowed us to deeper out network, which cause the network to increase its performance to around 30% accuracy.

Transfer learning:

After learning about transfer learning, we decided to try this approach.

We found a network called "PANN", and used its weights.

After a few iterations with different amount of "frozen" epochs, and different learning rates, we settles on 4 epochs with frozen backbone with LR of 0.003, and afterwards using LR of 0.0002.

Data augmentation:

We tried to use data augmentation like:

- Time strech
- Pitch shift
- Noise
- Volume change

We had to let them go, cause we only saw decrease with our accuracy.

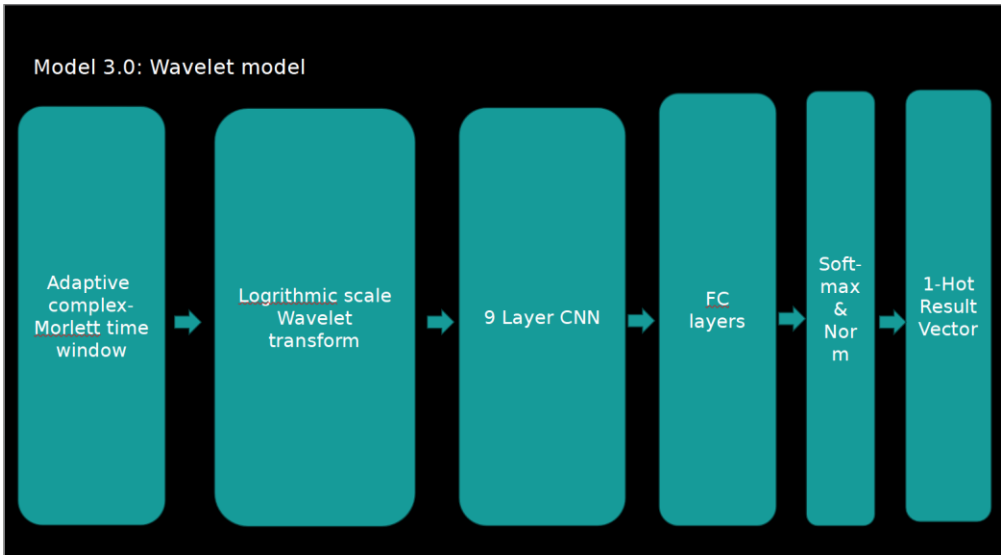
Increase network size:

So we decided to increase the network size, more params, more conv nets, bigger batch size.

This indeed helped a bit, to increase the accuracy from around 45% to 51%, but eventually we reached a barrier, where the collab's GPU, couldn't handle the size anymore, so we had to stick with the biggest amount it allowed.

Model 3:

Our final solution uses a variance of model 2. But before we settled on these results, we wanted to try one more architecture:



A wavelet transform, consists of a kernel wavelet, that imitates a specific signal. By stretching and moving this kernel over the signal in many permutations, we can produce a spectrogram that excels in identifying specific features.

We believed that using a Morlet kernel, which excels in recognizing sound patterns, we could get better results with one single transform.

However, we ran into many efficiency problems, mostly around implementing the transform inside the model.

After working on this model for a while with not a single successful run, we decided to let that idea go. We still believe there is a great potential to this method, should we ever decide to return to this project.

Beyond the issues, this model has the potential to be very lightweight, and to recognize features easily by their frequency localization. This model reached 2 million paramaters at most.

Reference to previous papers- We did not consult previous papers or references. In hindsight this could have helped us a lot, but we wanted to make our innovative ideas work without help. This is a good lesson for us, by the end of the day.

Conclusions and Future Work:

As we stated, one big conclusion was to consult previous papers and other projects.

We wanted to create something completely original, but as students with little experience, we wasted dozens of hours trying to make this project succeed. This would have been easier if we based our network on other people's work.

Second, our important conclusions, are about the model itself:

1. When using conventional networks, we must have enough depth to reach the specific features we need to examine. And thus we require a minimal depth for a task.
2. Our method may had some good ideas, but some features may only be noticeable on the whole picture, perhaps our segregation of feature into different networks hindered the work of the model.
3. Using a large repository, threw us into unexpected bugs. Next time, we would be wiser in writing more compact code, and then building the utilities once it runs.
4. Our choice of data set may have been problematic, since although it was noisy, we couldn't label it properly (not even by using a teacher model)

For future work, we would love to find the opportunity to make model 3 work. Perhaps this System would make multi label recognition possible, but we lack the time and resources to progress in that direction right now.

This has been a valued experience, we learned a lot about teamwork, different models, and building something big together.

Ethics Statement by:

Ori Hefetz - 208783100

Ofek Dahan – 313598385

Our project aims to recognize musical instruments from a short audio file. Using our new idea around signal processing and CNNs.

2) LLM-assisted stakeholder analysis

2a) Stakeholders (3 types)

End users (musicians, music educators, librarians/archivists, creators who tag audio):

This tool classifies short audio clips by the *most likely* instrument (single-label "dominant instrument"). It was trained primarily on the IRMAS dataset and similar audio; accuracy varies by instrument, audio quality, and whether multiple instruments play at once. Probabilities are confidence scores, not guarantees; in mixtures, the top class can be the loudest/most characteristic timbre rather than the only source. The model is not intended for surveillance or rights enforcement. For best results, use ≥5–10 s clips with moderate SNR, and treat low-confidence outputs as "needs review." A model card documents per-class performance, common failure cases (e.g., sax vs. clarinet, strings overlap), and known dataset biases.

People represented in the audio datasets (artists/rights holders whose recordings are used for training/eval):

We use publicly available recordings (e.g., IRMAS) under their respective licenses solely for research/education. Audio may be resampled and converted into time–frequency features (e.g., spectrograms/scalograms) and learned weights; we do not attempt to identify individuals. We attribute datasets and respect their terms; if you are a rights holder with concerns about inclusion, licensing, or downstream use, contact the maintainers for removal or clarification. No personal data beyond the audio content is collected. We publish a data statement listing sources, licenses, and any preprocessing performed.

Deployers/maintainers (your project team, course staff, or any platform that integrates the model):

You are responsible for presenting the model card, communicating limits in the UI (e.g., "dominant-instrument only," "not for polyphonic attribution"), choosing sensible defaults (clip length, thresholds), and monitoring performance drift. Provide an appeal/contact path, log model versioning, and retrain or recalibrate when data distribution changes. Ensure your use

complies with dataset licenses and local policy; disable uses that could enable surveillance or rights enforcement without due process. Keep documentation synchronized with each model update and record known issues.

2c)

Who gives each explanation? (roles, timing, and channels)

End users → Explanation authored by the project team and reviewed by course staff/deployer, delivered at onboarding (readme/model card link), in-UI tooltips near outputs (e.g., “dominant-instrument probability”), and in a FAQ. Updates are the deployer’s responsibility whenever the model or thresholds change.

Dataset subjects/rights holders → Explanation authored and published by the project team in the model card & data statement (repo/docs). The deployer includes the data statement link in public materials and handles takedown/queries via a visible contact channel; course staff provide oversight.

Deployers/maintainers → Explanation provided by the project team during handoff (technical docs: training recipe, metrics, limits, change log). Course staff/supervisor ensures compliance and approves the communication plan. The deployer keeps release notes and versioned model cards current.

3. Our reflection:

We don’t see much use for your model by right holders, this model is not smart enough to recognize specific songs. This is meant as more of a casual tool for end users to hone their knowledge and practice music.

Niche uses by curious people is something that could be a generic role for our model. If you indeed hear some noisy music and want to recognize the instrument, this is your model to do so. As usual, we believe ChatGPT is going over the top with his hallucinations, but it is cool to see the ideas it is prompting. The model is indeed raw currently, it would require much adjustment to become a tool for clients.