Development of convolutional neural networks for detection of beats in short music samples

Benjamin Banks, David Svane and Lucas Pedersen

19. January 2024

Abstract

This paper addresses the challenge of beat detection in music, a crucial first step towards live synchronization of visual elements to musical tracks. Previous beat detection methods based on expert systems are either too simple to account for the varieties in music or take a significant amount of development time and expertise. Utilizing that convolutional neural networks are good at finding their own patterns in data, we developed and assessed two convolutional neural network (CNN) systems for identifying beats in 10-second music snippets: one based on waveform (WM) analysis and the other on spectrogram (SM) analysis. Our evaluation shows that the spectrogram-based CNN significantly outperforms the waveform-based approach in beat placement accuracy on a 5% significance level. Our findings might suggest that spectrograms highlight certain features that are helpful for beat tracking, which might be used for further development in the area.

Introduction

An integral part of almost all music in the world is the concept of an underlying rhythm. When dancing, we move in sync to the beat, making entire crowds of people move as one. In this context, people such as concert coordinators and hip speaker designers are interested in tracking the beats of music to automatically add visually pleasing synchronized effects. Thus, improving the experience of the listeners.

Currently, this is done in two different ways: firstly, with models that take entire prerecorded songs as input to analyze them, and secondly with live beat detection, where beats are detected in a live audio feed. The analysis of full songs can now effectively be made by neural networks (NN), partly because of the advantage of NNs being able to find their own patterns in data, which is well suited for the problem domain of analyzing music, which is full of patterns that are hard to completely describe. NNs have led to improved models over old expert systems[1][2], and are also able to detect other components of music, like when vocals appear in a track[3], potentially allowing for complex understanding and separation of different musical parts. Full song analysis AI beat detection algorithms are already widespread, helping people on social media make fast and engaging video editing[4].

Instead, for live beat detection, we have found expert systems to be dominant in publicly available methods. In contrast, we are interested in utilizing the benefits of neural net technology for this challenge. As a first step in the process, we intend to develop a CNN model for finding the beats in 10 second music clips, envisioning that our model learns to detect previous beats without much context, which can later be used to predict future beats. We chose a CNN model, as they are more resistant to overfitting and work well for data with repeating patterns or features that need to be extracted. The simplest choice is to feed the model raw waveform data as an input, which is simply data of the oscillating air pressure over time which humans interpret as sound. However, in other beat tracking algorithms[5], the authors have found that transforming the data to spectrograms before analyzing it was critical for their algorithm's success. Spectrograms show sound as a 2D image, with the frequency range on the y-axis, and time on the x-axis, with the brightness of the color being the intensity of the specific frequency at the specific time. To make the best model possible, we decided to develop two neural networks, each based on the two sound representations respectively, the waveform model (WM) and the spectrogram model (SM) and compare their performance.

We hypothesize that the development of a model based on spectrograms would have an advantage, as the spectrograms separate the frequencies into different layers called bands, which consequently makes different instruments and vocals easier to identify. We theorize that since this preprocessing makes it much easier for us, as humans, to locate the beats by visual inspection, a neural network would learn to identify the beats more effectively from them.

Methods

All python-functions are used with default parameters for the specified versions unless otherwise specified. Relevant code, data and other information can be found on our git[17].

Song snippets were generated from the collection "Now That's What I Call Music!"[6]. Beat annotations were created on the full songs using the beat tracking feature of the python-library Librosa[7]. Then, for each song, the first 10 seconds and 2-4 randomly selected 10 second snippets were stored. Snippets not containing at least 5 annotations or containing the last annotation of the entire song, were excluded to eliminate inaccurate data. This resulted in 14.027 snippets of which a sample of 100 were manually inspected for the accuracy of annotations and determined to be satisfactory. We then divided the snippets into training and testing sets,

making sure all snippets from the same song ended in the same set. When generating snippets, we made it easy to later change the proportion of test and training data, in case we didn't have enough test data to conclude anything. However, changing this proportion wasn't necessary, and only our first split was used for all training and testing throughout.

Binary annotations were generated for all snippets by dividing them into 430 bins, corresponding to 23.3ms per bin. Each bin was given a value of 0 or 1 based on the presence of a beat, with 1 meaning a beat is present.

Waveform input for WM were generated by downsampling each 10 second song snippet to 27.520 data points using the "librosa.resample"[8], resulting in a sample rate of 2.752 Hz. This was done to match the total number of data points of the spectrograms described in the next section.

Spectrogram input for SM were generated using the "librosa.stft"[9] (short time Fourier transformation). Window lengths ("n_fft") of 512 samples and stride lengths ("hop_length") of 256 samples were used, see appendix "Spectrogram Parameter Selection", everything else set to the defaults of Librosa. Resulting spectrograms were then downscaled by a factor of 4 in each dimension using "np.mean"[10], yielding 1/16 the number of datapoints in total. To see a visual representation of both a waveform and a spectrogram see the results section.

Fine tuning of the models was done separately and in parallel. Both models are trained on the same 10 second clips of music and have roughly the same complexity. The main difference between the models is the inputs representations of the music. We tried different model architectures for each model, and the ones we use were the best found for each. Using multiple distinct rounds of grid search, see appendix "Hyperparameters Grid Search", we also found their individual optimal hyperparameters.

Final model architectures of SM and WM are very similar. The overall structure of the CNNs are four convolutional layers given 64x430 and 27.520 data points as inputs using max-pooling and the ReLU function between each convolution. Then, a fully connected linear layer maps to 430 output bins. During training, the dropout is applied before this linear layer. After the final linear layer, a sigmoid function maps the outputs to the range [0;1].

Post model peak detection was conducted on the model's post-training distribution of 430 probabilities, which represent the models' confidence that the corresponding bin contains a beat. This data needs to be converted from probabilities to binary predictions: beat or no-beat. At this stage we have 10% (1408) of the snippets put aside for evaluation. The peak detection is done by using "librosa.util.peak_pick"[17], see github for the used parameters. We found these optimal parameters for the function with a grid search on 10% of the 1408 snippets, afterwards removing this data from the evaluation set to account for potential overfitting. We did this for both SM and WM separately, but with the same snippets, to ensure optimal peak detection parameters for both models. This left 1268 snippets for the final statistical evaluation of the models.

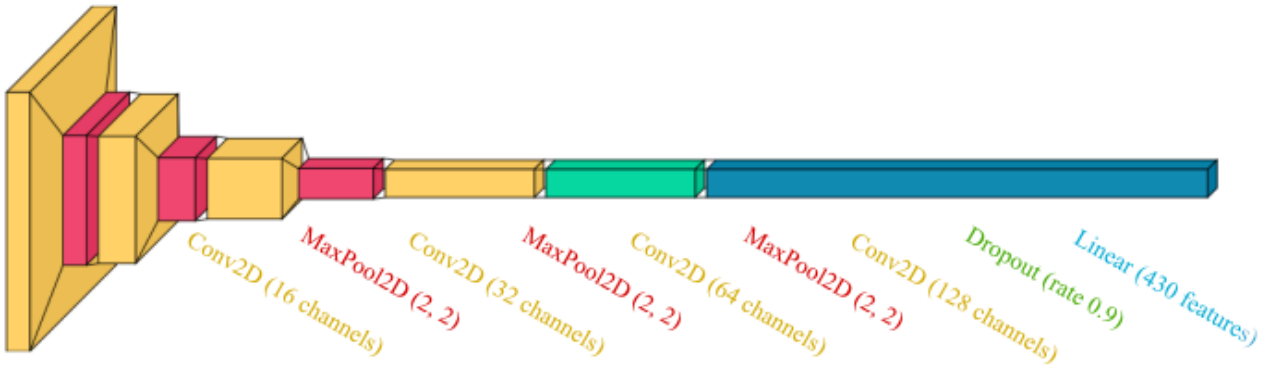

Figure 1: Model architecture visualization for SM using "visualkeras". The WM uses 1-dimensional functions instead and has an input size of 27,520. Yellow layers are convolutions, red are max pooling, green is our dropout as the last function before the linear output layer.

Model statistics were made to quantify the performance of the models. We calculated the F-measure with an error margin of 25ms and the Kulback-Leibler divergence (KLd) with histograms of 50 bins as suggested in Plumbley et al[11]. The advantage of the F-measure being that it's a simple, easily interpretable evaluation, taking the harmonic mean of two proportions, namely the proportion of beats detected that are correct, and the proportion of correct beats that were detected by the algorithm. KLd has the advantage of measuring "information gain", most importantly meaning off-beats aren't punished as heavily, as they still imply that the model has learned something and may simply be because of an ambiguous metric level in the song. The KLd is calculated by computing all the beat errors, the distance from the predicted beat to the closest annotation, of all snippets, dividing them by the interval between the annotations on both sides to normalize, plotting a probability histogram of these beat errors, and finally calculating the Kulback-Leibler divergence between this histogram and a uniform distribution. The same is done but where the role of annotations and predictions are switched, and the lowest value of the two is chosen. A higher value thus indicates that more information about beat locations is gained, but not necessarily that they are in phase with the annotations. We calculated the 95% confidence intervals differently for the two values. For the F-measure we used a simple 95% confidence interval of the mean of a sample of size over 30, assuming normality of the distribution of such a mean. For the KLd-value we did a non-parametric bootstrap of all the beat errors, for all the snippets, together, with a simulation count of 1000.

Model qualitative assessment was also performed to further assess the difference in accuracy of the models. 11 random snippets were chosen from the final test set for our two models to predict beats in. A click sound was then overlayed at the detected beat locations for each model and we made a manual comparison by ear. These sound files can be found on our git.

Results

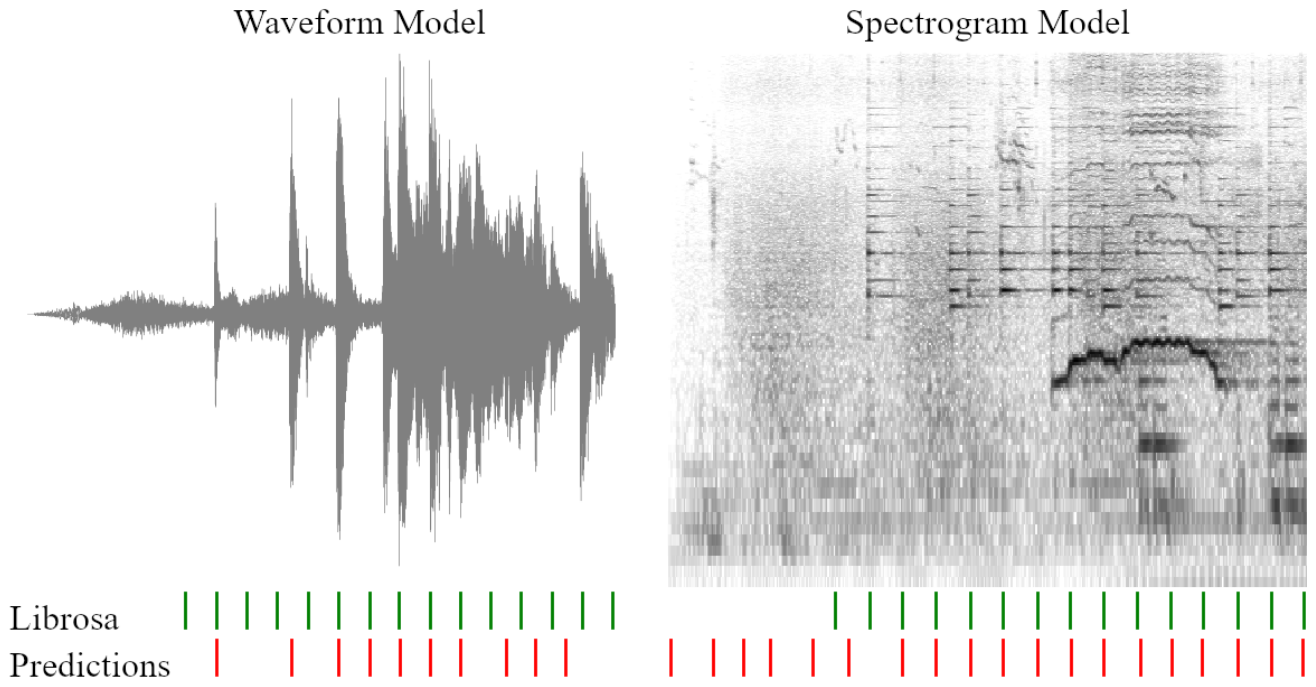

Figure 2: Comparison of the two models on the same song. The right is the spectrogram based model, the left is the waveform based model. The green lines represent the ground truth, beats detected by the Librosa library. The red lines represent the beats detected by the model.

The optimal hyperparameters for the models were 0.9 in dropout rate and 32 in batch size for both models. With regards to optimizer, the Adam optimizer with a learning rate of 0.001 gave the best results for the SM and Adadelta (standard settings) with a learning rate of 1 for WM. See appendix for hyperparameter gridsearch visualization.

It is statistically proven to a 5% significance level that the spectrogram model we developed and trained is a more accurate beat tracker, using the F-measure and the KLd evaluation methods, than our waveform model, for our dataset of western pop/rock music. This confirms our hypothesis that the model developed on spectrogram data would have a higher accuracy than the one developed on waveform data.

	WM	SM
KLd	1.11, CI: [1.08; 1.14]	1.71, CI: [1.68; 1.74]
F_measure	49%, CI: [48%; 51%]	61%, CI: [59%; 62%]

Table 1: Statistical results for the two models. KLd is the Kullback-Leibler divergence between the beat error histogram and a uniform distribution. F_measure is the F1 score between the ground truth and the model. CI is the 95% confidence interval. For both measures higher numbers corresponds to higher accuracy.

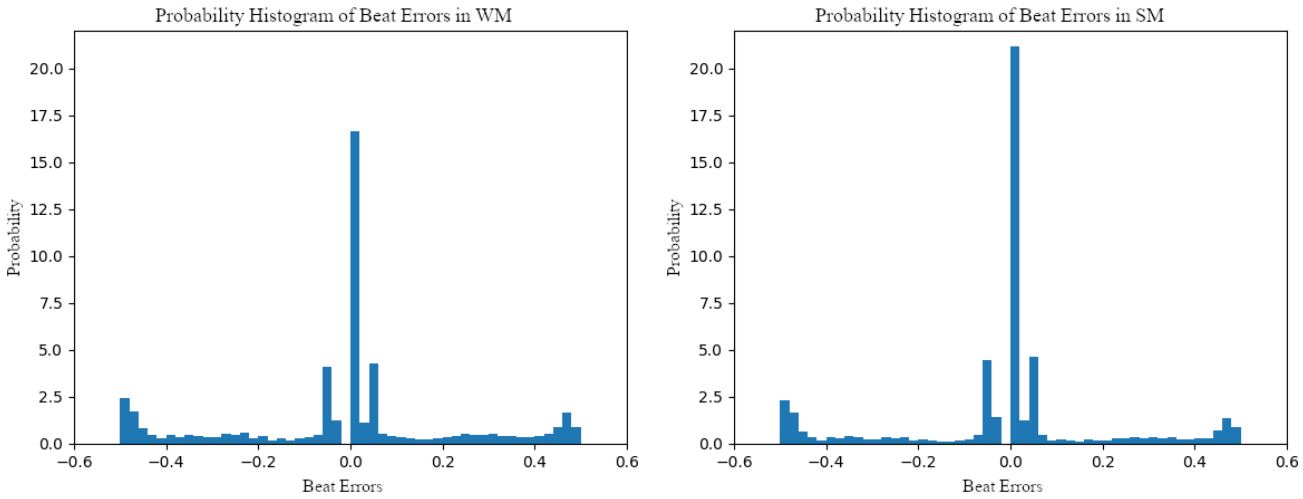

Figure 3: test

Discussion

Our results show that the SM was better in both performance measures. It especially outperformed the WM in the KLd measure, and less so in the F-measure. This is interesting as it may indicate that the waveform model is less likely to make false predictions in the beginning of a song. Approximately, one third of our snippets are at the beginning of songs, and when calculating the KLd, the false predictions in the beginning of a song actually get ignored because it only checks one beat interval before the first annotations for beat errors. However, the F-measure would punish these false positives, thus making the F-measure relatively smaller for the SM. This is further supported by examining specific examples of beat detections in snippets, like the one shown in figure 2. As we do not wish for false detections in the beginning of songs, we consider the F-measures more accurate representations of performance for our

project.

From the histograms of the beat errors, we can also see exactly how the two algorithms makes mistakes. The SM and WM actually make the same amount of offbeat predictions, however, the WM had more noise (random predictions) whereas the SM had more predictions exactly on-beat. From audial analysis of a small sample of the music, we noticed that the waveform model generally detected less beats overall. This all seems to confirm that, as we suspected, the WM has had a harder time learning the patterns that define beats.

While it's important to keep in mind that one cannot with certainty generalize the results from this study to different models, both models had roughly the same complexity, e.g. number of layers, and were given the same amount of datapoints, and this led to a significant difference in favor of SM. Therefore one could theorize that a spectrogram does extract some useful features that makes beat tracking easier, as indicated in[5]. As explained in the introduction, this can be explained by how in spectrograms, different musical elements are separated and clearly visible to the human eye.

On that note, it is also possible that we may have inadvertently introduced a bias by limiting the maximum complexity of the models we tested. Both models are the most complex models we tested, due to computational limitations. It is possible that a more complex waveform neural network, with more layers and channels, could outperform a spectrogram-based model, as a waveform model might need more complexity to replicate the information retrieval done by a Fourier transform, but might end up with better features for detecting beats in the end.

Investigating when and how our models failed, we noticed that singing/rapping interfered with beat detection in both models. One reason for this could be that it overlaps with important information for the models. This is interesting, as we thought mainly the models would learn to identify the drums and possibly the bass of songs, and the frequency ranges of these mostly extend outside the range of the human voice. In the same vein, one of the snippets our models worked best on[16], which didn't have any drums, but rather had a pure violin segment. All in all, our algorithms have learned to utilize many more musical elements than the drums, which shows promise for generalization to different music genres.

It may, however, also be since singing is often highlighted, and therefore loud, in songs. We noticed that the models often misinterpreted the "rhythm" in other musical events, such as singing, as beats, even though they aren't necessarily happening in fixed time intervals. Loud short sounds in between beats can make the models falsely predict a beat, meaning the model's sense of cyclic intervals isn't as well developed as the model's recognition and weighting to parts that look like a beat. This also makes sense considering the model structure, which didn't include any parts to specifically learn temporal patterns, but relying on convolutional layers, which are good at learning to identify local patterns.

Despite the tendencies to prioritize certain sounds as beats and not accounting for the time in between beats, the model does show some temporal awareness. E.g. in the analysis of one snippet [16], there is a break with no instruments in the music where the algorithm still manages to place a beat. To enhance this awareness we suggest implementing Long Short-Term Memory, which is designed to capture long-term dependencies in sequential data, in the CNN. We believe that this might be one of the most important improvement pointers based on the mistakes the algorithms make.

Finally, we should mention that our dataset consisted of music specifically from the album collection "Now that's what I call music". This leads to a myriad of biases, including but not limited to: Genre, recording, compression levels and normalization levels. Since both training and test set came from the same source, these biases are included in our model and in the evaluation of it. For the scope of our project, it was not possible to include more varied music, however, obtaining varied datasets with confirmed annotations is essential for further development.

Safety and ethical concerns regarding our models are minimal. There is a risk that some may consider the bias of our beat tracker towards western pop and rock music to be unethical. It has been seen that Snapchat's filters, among other more prominent things, have been criticized for not looking as good or working as well on people of color[14]. The same would most likely be the case for our beat tracker and non-western music, as a result of our dataset. However, we note that the controversy regarding Snapchats filters are often based on how it may affect people of color's self-perception by 'whitewashing' their features, invalidating such a personal thing as how they look. We argue that our models won't hold enough power to change people's view of music and in the worst case, people can simply opt out of using our models.

In conclusion, in the context of western pop-rock music, SM had a higher accuracy in beat placements than WM. This might indicate that one can gain an advantage for beat tracking CNNs networks by first ap-

plying a Fourier transform to the data, but it is uncertain whether waveform models would outperform spectrogram models using different datasets or model architectures, especially ones with more layers. The biggest issue our models are facing is detecting all sudden loud noises as beats. We believe a large point of improvement for both of our models, besides obtaining a better dataset, would be the use of a temporal convolutional networks, e.g. by employing Long Short-Term Memory in conjunction with convolutional layers, for the model to learn to space out beats somewhat evenly, according to the cyclical nature of beats.

References

[1] "Downbeat tracking with multiple features and deep neural networks", G. Richard et. al., 2016

[2] "Temporal convolutional networks for musical audio beat tracking", S. Böck et. al., 2019

[3] "Industry-first AI tech shows positions of vocals in tracks and Beatsource LINK is now supported", rekordbox, 2020

[4] "Beats Detection (version 3.6.0)", Effecthouse - TikTok, 2024

[5] "Analysis of the meter of acoustic musical signals", J.T. Astola et. al., 2009

[6] https://github.com/bforbanks/iBeat/blob/master/meta.csv

[7] "librosa.beat.beat_track" (version 0.10.1), Librosa, 2023

[8] "librosa.resample" (version 0.10.1), Librosa, 2023

[9] "librosa.stft" (version 0.10.1), Librosa, 2023

[10] "numpy" (version 1.25.2), NumPy, 2022

[11] "Evaluation Methods for Musical Audio Beat Tracking Algorithms", M.D. Plumbley et. al., 2009

[12] librosa.util.peak_pick" (version 0.10.1), Librosa, 2023

[13] "A probabilistic approach to simultaneous extraction of beats and downbeats", M. Omologo et. al., 2014

[14] "Snapchat's Insensitive Filters", P. Justin, 2021

[15] "visualkeras for Keras / TensorFlow" (version 0.0.2), 2021

[16] "Everybody Gonfi Gon New Atlantic Edit", Two Cowboys, 1994

[17] GitHub repository for iBeat, 2024

Appendix

Abbreviations

CI - "Confidence interval"
CNN - "Convolutional neural network"
KLd - "Kulback-Leibler divergence"
NN - "Neural network"
SM - "Spectrogram models"
WM - "Waveform models"

Responsibilities

Abstract - "Lucas Pedersen"
Introduction - "Lucas Pedersen"
Methods - "David Svane"
Results - "Lucas Pedersen"
Discussion - "Benjamin Banks"

All parts of the report was developed in collaboration.

Spectrogram Parameter Selection

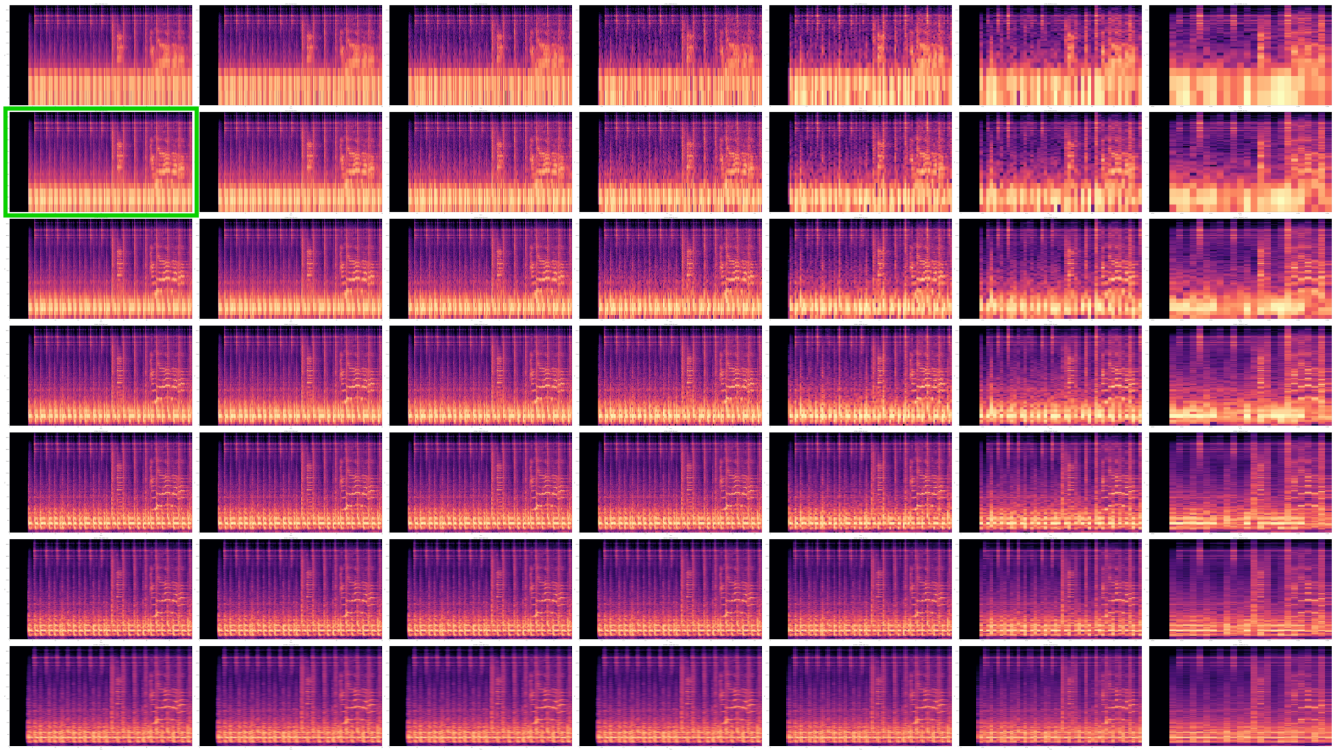

Figure 4: Combinations of tested spectrogram parameters. Columns represent stride lengths. Rows represent window lengths. The green box indicate the selected spectrogram parameters. It was chosen based on visual inspection of the balance between temporal and frequency resolution.

Hyperparameters Grid Search

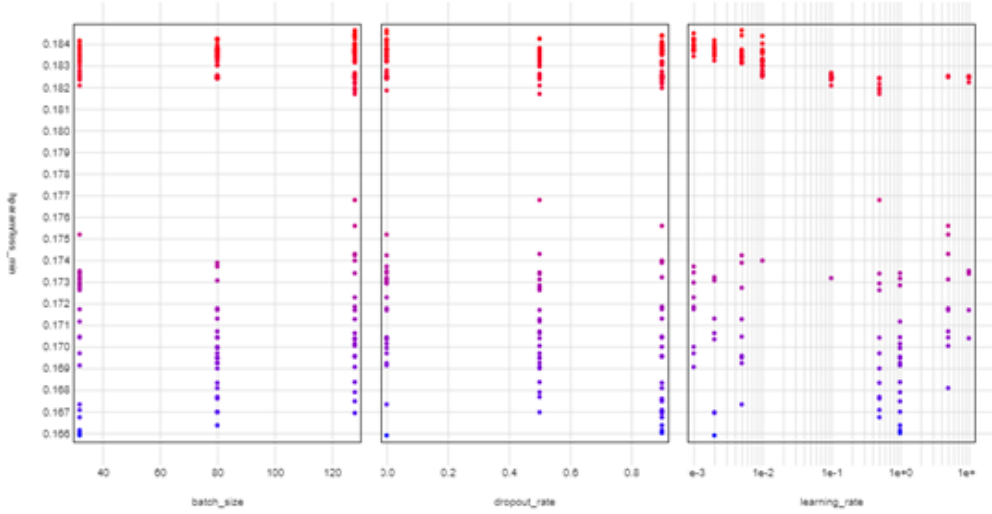

Figure 5: Hyperparameter data from the grid search on WM. (Sadly we didn't correctly log optimizers, but the optimal found was "adadelta")

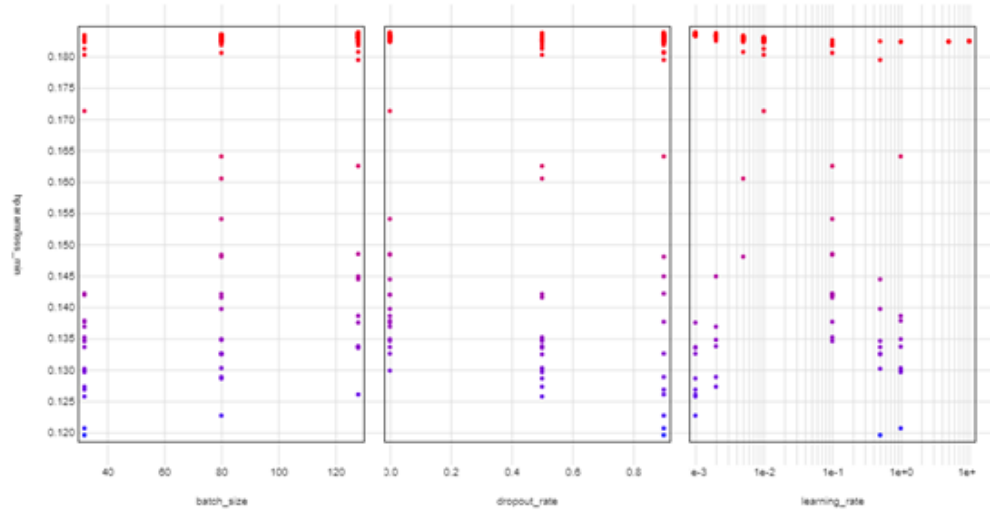

Figure 6: Hyperparameter data from the grid search on SM.