Danmark Tekniske Universitet

02461 - Introduktion til intelligente systemer

Instrument Audio Classification

Authors:
Adrian Valentin Kragh-Hillers s201390
Johanne Christine Arboe Franck s204088
William Henrik Klingsten Peytz s204145

January 22, 2021

Abstract

This paper analyses whether a convolutional neural network (CNN) or a long-short term memory network (LSTM) is better at classifying monophonic sounds in the form of instruments. It further compares the classification abilities of these networks against human intuition as a control group. We conducted an experiment where both of the networks trained and tested on the same dataset consisting of 3456 one second audio files spread across 15 different musical instruments. We made sure both models used the optimal number of epochs to get the best results. As a control experiment we also asked 9 people to train and test on a smaller dataset consisting of 15 training files and 50 test files. It was found that the CNN model had a higher accuracy between 96.27% and 98.62%, as opposed to the LSTM model, which had an accuracy of 94.51% to 97.44% both with a confidence level of 95%. We further concluded that when trained on this specific dataset both of these models outperform humans whom had an accuracy of 20.3% to 29.5% with a confidence level of 95% on our specific dataset.

I. Introduction

Musical instrument classification is a difficult task, especially when needed applicable to polyphonic data (data with multiple sounds or instruments played at the same time). But the foundation to solve polyphonic problems, could be to solve monophonic (one sound) problems with a high accuracy first.

To solve this, there are two essential neural networks which need to be mentioned, when talking about audio classification and the development of an audio classifier. The Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN), specifically the Long Short-Term memory (LSTM) model. CNN is primarily used for classifying images, but when sound is converted to a spectrogram, the model has been proved to achieve fine classification results. An LSTM model varies slightly from the general RNN model, with the implementation of forget gates, which sorts out irrelevent data.

In this paper we focus with a monophonic approach on classifiyng instrument audio data, through a deep learning pipeline. Our research question is: Which of the two types of neural networks, the CNN or the LSTM, is better at classifying instrument audio data?

As an LSTM based model is typically seen as the better network when classifying audio, we hypothesise that an LSTM based model is slightly better than a CNN based model at classifying musical instrumental data, hence we operate on sequential data. Furthermore we also expect both networks to outperform humans who have no profound knowledge of musical instruments.

II. Methods

In order to answer the research question and test our hypothesis, we have found a pre-existing code [1] which includes both a CNN (in 2-dimensions) and an LSTM Network both operating on the audio data using the Kapre and Tensorflow library [2]. We briefly mention the architecture and parameters of the two models, to state the difference[1]:

The CNN model has 14 layers, with the first layer as a normalization layer to standardize our data with zero mean, afterwards comes six activation functions; firstly a convolutional -tanh, four -ReLU and one softmax layers, with four maxpooling layers inbetween the activation functions. The last layers are two dense-, one dropout- and one flatten layer.

The LSTM model is constructed with a total of 12 layers, starting as the CNN model with a normalisation layer, then two timedistributted wrapper layers, and a bidirectional layer. The LSTM model has one less activation function, then four regular densely-connected NN layer activation functions, one with -tanh, three -ReLU and also a softmax layer. Finally the LSTM also got a concenate-, maxpooling- and a dropoutlayer. For optimization, both of the models use the Adam algorithm.

We initialise our experiment with the same hyperparameters: 128 mel frequency spectogram bins, a sampling rate at 16 kHz, 512 FFT samples, a window length of 25 ms and a step size of 10 ms [6].

We operate on a monophonic dataset consisting of 15 different musical instruments; cello, clarinete, contrabass, flauta, guitar, hi hat, oboe, alto saxophone, soprano saxophone, trombon,

[1] See appendix, fig. 2 and 3 for more information.

trompa, trompet, tuba, viola and violin.[2] Our dataset consists of 2511 files of different time lengths.

In order to train our models to be able to classify these 15 instument classes, we first remove noise via a signal envelope. The signal envelope removes low magnitudes below the threshold we set equal to 100. Afterwards, we downsample the audio files and split on delta time one second. This gives us 3456 audio files.

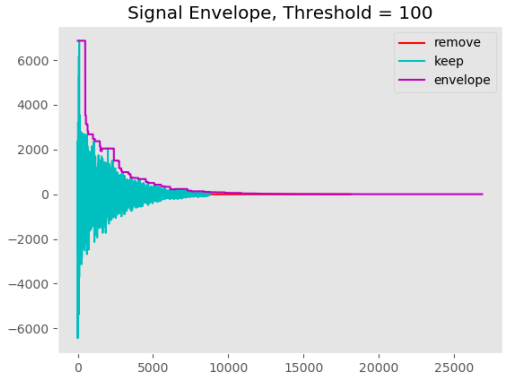


Figure 1: *Signal Envelope visual of a hi hat.*

First we compute the sample size we need to test our models on when we want a confidence level of 95% and 2% margin of error.[3] Via our pilotstudies we have a proportion of approximately $p = 92\%$.

$$n = 1.96^2 \cdot \frac{0.92 \cdot (1 - 0.92)}{0.02^2} \approx 707 \qquad (1)$$

Based on this sample estimate we randomly[4] remove 20% of the 3456 audio files, meaning that we test our two models on the same 692 audio files, which neither of the models have encountered before.

We divide the remaining traning data so that we train on 90% and validate on 10% of the data. To determine when to stop training our models we plot the training loss and validation loss as functions of number of epochs. We found that the CNN model starts overfitting after approximately 100 epochs. The LSTM model starts overfitting after approximately 70 epochs.[5] However, we see unusual spikes in these plots; sometimes the loss is unusually high (and the accuracy unusually low) for a single epoch. Then the curve continues the decreasing tendency. The spikes are not periodical, and we suspect them to be a result of using the Adam algorithm. However, our algoritm is designed to only update the models when the loss is decreasing. This means that our model "ignores" the spikes.

However, we seek to find the optimal number of epochs to train each of the models on, which is difficult to read off the loss curve that is frequently abrupted by spikes. The Adam algorithm optimizes by loss so we continue to do this. Thereby, we take the average of 10 runs of both the CNN and LSTM to find an average loss curve.[6] On average the optimal number of epochs for the CNN model is around 100 epochs. For the LSTM the optimal is around 61 epochs. This is the estimated number of epochs to optimaly train our models on.

Setting up the experiment with humans

To check if our two neural networks outperform humans, we test how well humans without profound musical knowledge classify instruments. First, we calculated the needed sample size, with a confidence level of 95% , and with 2% margin of error

$$n = 1.96^2 \cdot \frac{0.95 \cdot (1 - 0.95)}{0.02^2} \approx 457 \qquad (2)$$

To estimate this sample size we created a survey with 50 randomized instrument audio files [5], and made 9 people fill out the survey[7], which gives us a sample size of 450, which is close to the calculated sample size of 457. We chose our participants among our families and friends. We know that none of the

[2] 14 of the 15 categories are from Kaggle[3]. We added the hi hat category[4] to have a more differentiated dataset, hence the existing data set mainly consists of instruments of the brass family (trompets etc.) and the string family.

[3] All statistical formulas are from course material.

[4] We have build this into the existing code ln 90 train.py. See appendix, tab. 1 for another random state that gives the same final accuracy of the models (within the margin).

[5] See appendix, fig. 4.

[6] See appendix, fig. 6 and 7.

[7] See Sporgeskema.pdf

asked participants have any musical education, however their musical instrumental knowledge may vary.[8] To give the participants the same understanding of these 15 instruments, they get to train on one audiofile from each category before we test them.

Methods used when analysing the results
We test our final models on the same 692 files[9] and record which files the two models classify as which instrument. Thereby, we collect data of the total number of files classified correctly, along with which instruments the two models often misclassify. We visualise this with confusion matrices.

We find that our proportion, p, is greater than 90% for both the LSTM and the CNN model. Therefore, we calculate the confidence level with a correction, using the Agresti-Coull interval when we compute the accuracy of the models:

$$\widetilde{p} \pm 1.96 \cdot \sqrt{\frac{\widetilde{p}(1-\widetilde{p})}{n+4}} \tag{3}$$

Where

$$\widetilde{p} = \frac{p \cdot n + 2}{n+4} \tag{4}$$

We calculate the confidence level of the human survey with the formula:

$$p \pm 1.96 \cdot \sqrt{\frac{p(1-p)}{n}} \tag{5}$$

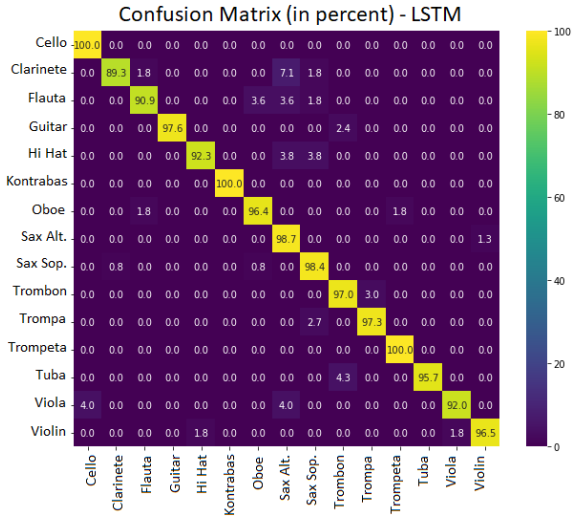
III. Results

When the LSTM network and the CNN network classify the same 692 audio files we get the test accuracy of the two networks. The accuracy and the number of correctly classified files are listed for the models in the following table:

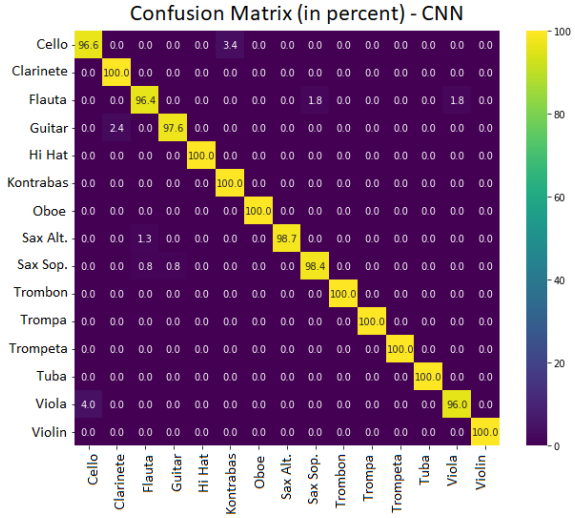
	Correct files	Interval Accuracy
LSTM	666	94.51% - 97.44%
CNN	684	96.27% - 98.62%

[8] See appendix, fig. 5.
[9] See predict.py

For the LSTM model
The LSTM model classifies the 15 instruments correctly with an accuracy of 94.51% to 97.44% with a confidence level of 95%.[10] Below we see a normalized confusion matrix for the LSTM network.



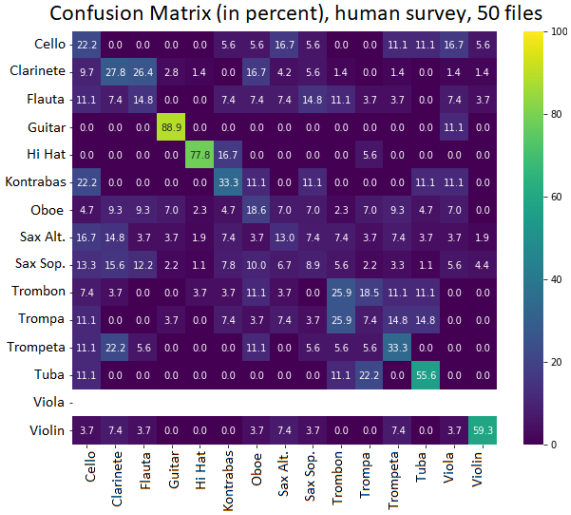
For the CNN model
The CNN model classifies correctly with an accuracy of 96.27% to 98.62% with a confidence level of 95%.



Results from survey: 9 participants answered 450 questions in total, with 112 correctly

[10] Please notice that the confusion matrices are normalized, and therefore the number of files in each category varies - however the distribution between the two networks are the same for each category.

classified instruments. Hereby the sample estimate of the accuracy is nearly 25% right answers. When we compute the confidence level (formula 5) we get that with a confidence level of 95% its possible to say, that a human answers between 20.3% and 29.5% correctly when classifying one second audio clips of instruments.[11]



When the CNN and LSTM classify the 50 audio files, both models get 100% accuracy, confusing no instrument with another.[12]

IV. Discussion

Both of the networks have a remarkably high accuracy on this dataset. The CNN network have an accuracy of 96.27% to 98.62%, whereas the LSTM model have an accuracy of 94.51% to 97.44% with a confidence level of 95%. The CNN network turned out to be better than the LSTM network, in almost all instances, but when looking at the intervals it is important to realize that there is a probability that the LSTM network peforms better, as the higher bound of the LSTM network, 97.44%, is greater than the lower bound of the CNN network at 96.27%. To summarise, the CNN model is still generally better, differing from our hypthesis expecting the LSTM model to be better.

[11] Notice, no viola in test-set cause of randomization.

[12] See appendix, fig. 8.

The CNN and the LSTM

The CNN model classifies 9 of 15 categories 100% correctly. On the contrary the LSTM model classifies 3 of 15 categories 100% correctly. Both models are unable to classify the flauta, guitar, both saxophones and the viola correctly with 100% accuracy. However the accuracy is still above 90%. It would be understandable for humans to have a hard time differentiating between the alto saxophone and the soprano saxophone, but both the CNN and LSTM confuse the saxophone sound for completely different instruments; flauta and the guitar.

In our hypothesis we state that we expect the LSTM model to perform better on instrument classification because former studies have shown that the model performs better on sequential data. This experiment indicates the opposite. By our results, it is shown that the CNN has a sligthly higher accuracy interval than the LSTM, therefore to be able to conclude that the CNN is in fact the optimal model for other kind of similar data, other perspective needs to be accounted for. The CNN model needs nearly twice the amount of iterations/epochs before reaching optimal perfomance than the LSTM, and it takes approximately double the time to train the the CNN model.[13] Both problems need to be addressed, because time and computational power are finite ressources. The architecture of the models might have an impact as well even though the models have approximately the same parameters, however the construction of the two different types of network differ.

Data handling

We have only trained the models on this specific dataset, so it is possible to question whether our two models would fit quite as accurate on another test-set or another dataset. Another issue with our data handling, is that the trained models have a chance of containing our 50 test files in the training set, because of the way we first subtract the 50 files, and afterwards decide how to split in train/validation

[13] See appendix, tab. 2.

and test. This could be an explaination to why the CNN and LSTM correctly classify all 50 test files. Therefore the comparison might not be equal.[14] As shortly mentioned in methods, we observe "spikes" in some epochs when training our models. The spikes seem to be unavoidable when using Adam in our models, because of the Mini-Batch Gradient Descent (MBGD) [7]. MBGD creates a random validation set, and sometimes it indicates with spikes, that it gets a really bad randomization, hence an epoch where e.g. no hi hats are in the training but all put aside for validation. We still account for the spikes, when we train on another dataset [1].

Human questionnaires
With only few ressources available, the survey test for our human test group, contains potential biases and issues. Though the survey contains issues, we thought it would give a good indication of human capabilities to categorize these audio clips, even though our results are questionable. In the survey, we assume that the same 50 randomly chosen test files on 9 different people, are able to count as $9 \cdot 50 = 450$ samples. For future research we would prefer to have random chosen files for each person, so that we test on a wider aspect of the full dataset, and avoid especially difficult files each time as a bias. Furthermore, we would want to have at least 30 participants taking part in the survey, because 9 participants are not enough for a broader statistical saying. Other preferences would be that each participant had fewer files to classify, because of fatigue. This would result in at more realistic view on the generel person. Last but not least, the participants had a hard time recognizing the instruments with only one second audio files, and therefore a longer audio file might haven given the humans a better chance versus our models. Even though our participants did not get close to the models score, they did not classify randomly hence we do not see that they classify with evenly percentages (figure 4). However for some instruments it seems that some of the participants have been guessing randomly e.g. the saxophone has a slightly evenly guessing rate distributed over all 15 instruments. We wonder if the participants find it easier to classify e.g. the guitar (88.9% accuracy) and the hi hat (77.8% accuracy) as these are more common and recognisable instruments. Perhaps this could be the reason why it indicates guessing in the saxophone category, simply because the participants were unfamiliar with this sound (even though they could compare with the training file).

Why audio classification is important
In this experiment we classified monophonic sounds using an LSTM and a CNN network, but in the real world polyphonic sounds are more common. Audio classification in more general terms is actually in use in many different technologies today. Spotify generates playlists based on music genres and when Siri recognises our voice on our phones. For technologies such as these, accuracy is imperative, and using the right models and neural network is key for getting the best user experience. An interesting further experiment could be to test how well the LSTM and CNN model would perform on a polyphonic sounds. Therefore experiments such as this one, that at a first glance can seem quite limited, actually have more real life applications than what one might first think.

V. Conclusion

By conducting this experiment and after analysing the results, we can conclude that when it comes to classifying instrument audio data the CNN model had an accuracy of 96.27% to 98.62%, whereas the LSTM model had an accuracy of 94.51% to 97.44%, both with a confidence level of 95%.

Thereby this study shows that the CNN model on average is slightly better than the LSTM model at classifying instruments, despite our initial hypothesis. Of course, this is only based on the experiments we performed on this specific dataset.

[14] See appendix, fig. 8.

We can furtherly conclude with statistical certainty that AI is better than non-musical educated humans at classifying instruments, independent of which of the two models we used, as the humans answered correctly with an accuracy between 20.3% and 29.5% with a confidence level of 95%.

VI. Learning Outcome

During this project we have experienced the hard workflow when designing an experiment, as we had to change our research question and hypothesis multiple times throughout the designing process. We have encountered new kinds of neural networks and methods. Overall we have gotten a better understanding of how to implement code in practice. It has been difficult handeling all the aspects of this project, as this process is still new to us. It has also at times been difficult for all of us to work efficiently at the same time, as we have different skillsets that are useful in different parts of the process. Some of us have been working hard on getting the code to work, designing the experiment with all the methods and statistics it entails, whereas others have participated more during the writing process. There has, unfortunately, been a difference in the amount of time and energy put into the project.

References

[1] *author*, Seth Adam. *title*, Audio-Classification (Kapre Version) *GitHub*, https://github.com/seth814/Audio-Classification.

[2] *author*, Choi, Keunwoo and Joo, Deokjin and Kim, Juho. *title*, Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Kera *documentation*, https://kapre.readthedocs.io/en/latest/ *booktitle*, Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning *GitHub*, https://github.com/keunwoochoi/kapre. *year*, 2007 *organisation*, ICML

[3] *author*, Dibakar. *title*, Music Instruments and 2D figures *Kaggle*, https://www.kaggle.com/dibakarsil/music-instruments-and-2d-figures.

[4] Hi hat databases *author*, Tenfingers, LuxrayProd, rs. *title*, Accent Perc Hats, Trap Loop Open Hats, Closed Hi-Hat Pattern - Loop. *SampleFocus*, https://samplefocus.com/samples/accent-perc-hats,https://samplefocus.com/samples/trap-loop-open-hats,https://samplefocus.com/samples/closed-hi-hat-pattern-loop.

[5] Randomizing python code. comment from THE PHOENIX 777 TDW *author*, THE PHOENIX 777 TDW. *title*, Best way to choose a random file from a directory. *StackOverflow*, https://stackoverflow.com/questions/701402/best-way-to-choose-a-random-file-from-a-directory.

[6] Tutorial on he's github code. *author*, Seth Adams. *title*, Deep Learning for Audio Classification (kapre version). *Youtube.com*, https://www.youtube.com/playlist?list=PLhA3b2k8R3t0SYW_MhWkWS5fWg-BlYqWn.

[7] Adam optimizer and spikes *author*, Abdul Fatir *title*, Explanation of Spikes in training loss vs. iterations with Adam Optimizer *stackexchange*, https://stats.stackexchange.com/questions/303857/explanation-of-spikes-in-training-loss-vs-iterations

[8] LSTM networks *author*, Colah *title*, Understanding LSTM Networks *stackexchange*, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[9] Understanding CNN *author*, Vincent Tatan *title*, Understanding CNN *stackexchange*, https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd6

[10] Should We Abandon LSTM for CNN? *author*, Geoffrey So *title*, Should We Abandon LSTM for CNN? *stackexchange*, `https://medium.com/ai-ml-at-symantec/should-we-abandon-lstm-for-cnn-83accaeb93d6`.

[11] Information about statistic from course 02461 *title*, Statistic *directory fil name*, Statistics

VII. Appendix

i. Architecture

```
def LSTM(N_CLASSES=15, SR=16000, DT=1.0):
    input_shape = (int(SR*DT), 1)
    i = get_melspectrogram_layer(input_shape=input_shape,
                                 n_mels=128,
                                 pad_end=True,
                                 n_fft=512,
                                 win_length=400,
                                 hop_length=160,
                                 sample_rate=SR,
                                 return_decibel=True,
                                 input_data_format='channels_last',
                                 output_data_format='channels_last',
                                 name='2d_convolution')
    x = LayerNormalization(axis=2, name='batch_norm')(i.output)
    x = TimeDistributed(layers.Reshape((-1,)), name='reshape')(x)
    s = TimeDistributed(layers.Dense(64, activation='tanh'),
                        name='td_dense_tanh')(x)
    x = layers.Bidirectional(layers.LSTM(32, return_sequences=True),
                             name='bidirectional_lstm')(s)
    x = layers.concatenate([s, x], axis=2, name='skip_connection')
    x = layers.Dense(64, activation='relu', name='dense_1_relu')(x)
    x = layers.MaxPooling1D(name='max_pool_1d')(x)
    x = layers.Dense(32, activation='relu', name='dense_2_relu')(x)
    x = layers.Flatten(name='flatten')(x)
    x = layers.Dropout(rate=0.2, name='dropout')(x)
    x = layers.Dense(32, activation='relu',
                         activity_regularizer=l2(0.001),
                         name='dense_3_relu')(x)
    o = layers.Dense(N_CLASSES, activation='softmax', name='softmax')(x)
    model = Model(inputs=i.input, outputs=o, name='long_short_term_memory')
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

Figure 2: *LSTM model*

```
def Conv2D(N_CLASSES=15, SR=16000, DT=1.0):
    input_shape = (int(SR*DT), 1)
    i = get_melspectrogram_layer(input_shape=input_shape,
                                 n_mels=128,
                                 pad_end=True,
                                 n_fft=512,
                                 win_length=400,
                                 hop_length=160,
                                 sample_rate=SR,
                                 return_decibel=True,
                                 input_data_format='channels_last',
                                 output_data_format='channels_last')
    x = LayerNormalization(axis=2, name='batch_norm')(i.output)
    x = layers.Conv2D(8, kernel_size=(7,7), activation='tanh', padding='same', name='conv2d_tanh')(x)
    x = layers.MaxPooling2D(pool_size=(2,2), padding='same', name='max_pool_2d_1')(x)
    x = layers.Conv2D(16, kernel_size=(5,5), activation='relu', padding='same', name='conv2d_relu_1')(x)
    x = layers.MaxPooling2D(pool_size=(2,2), padding='same', name='max_pool_2d_2')(x)
    x = layers.Conv2D(16, kernel_size=(3,3), activation='relu', padding='same', name='conv2d_relu_2')(x)
    x = layers.MaxPooling2D(pool_size=(2,2), padding='same', name='max_pool_2d_3')(x)
    x = layers.Conv2D(32, kernel_size=(3,3), activation='relu', padding='same', name='conv2d_relu_3')(x)
    x = layers.MaxPooling2D(pool_size=(2,2), padding='same', name='max_pool_2d_4')(x)
    x = layers.Conv2D(32, kernel_size=(3,3), activation='relu', padding='same', name='conv2d_relu_4')(x)
    x = layers.Flatten(name='flatten')(x)
    x = layers.Dropout(rate=0.2, name='dropout')(x)
    x = layers.Dense(64, activation='relu', activity_regularizer=l2(0.001), name='dense')(x)
    o = layers.Dense(N_CLASSES, activation='softmax', name='softmax')(x)
    model = Model(inputs=i.input, outputs=o, name='2d_convolution')
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Figure 3: *CNN model*

ii. Overfitting and spikes

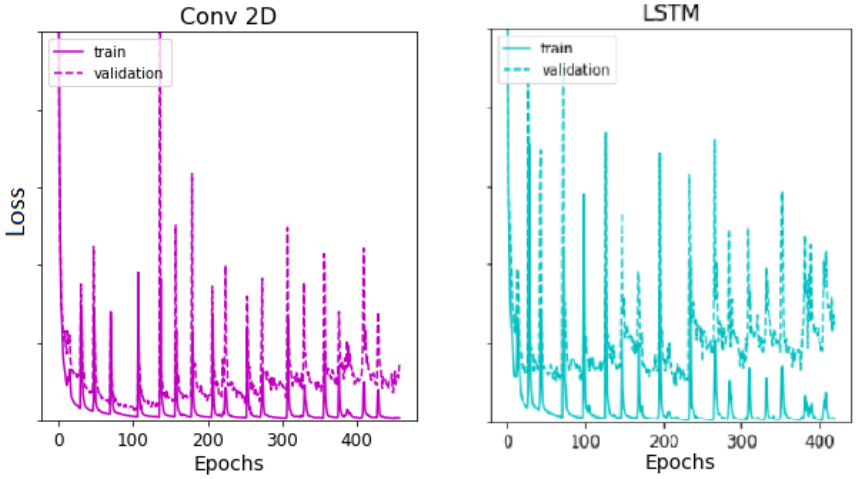


Figure 4: *Overfitting both CNN and LSTM. Notice the spikes in both plots as well.*

iii. Accuracy Table for different random state

Threshold = 100			
		Accuracy	
	Epochs	Random state = 0	Random state = 1
LSTM	61	p = 95.26% 93.18% - 97.34%	p = 95.98% 94.51 % - 97.44%
CNN	100	p = 98.42% 97.20% - 99.64%	p = 98.56% 96.27% - 98.62%

Table 1: *Accuracy of random state 0 and random state 1, with a 95% confidence level.*

iv. Questionnaries

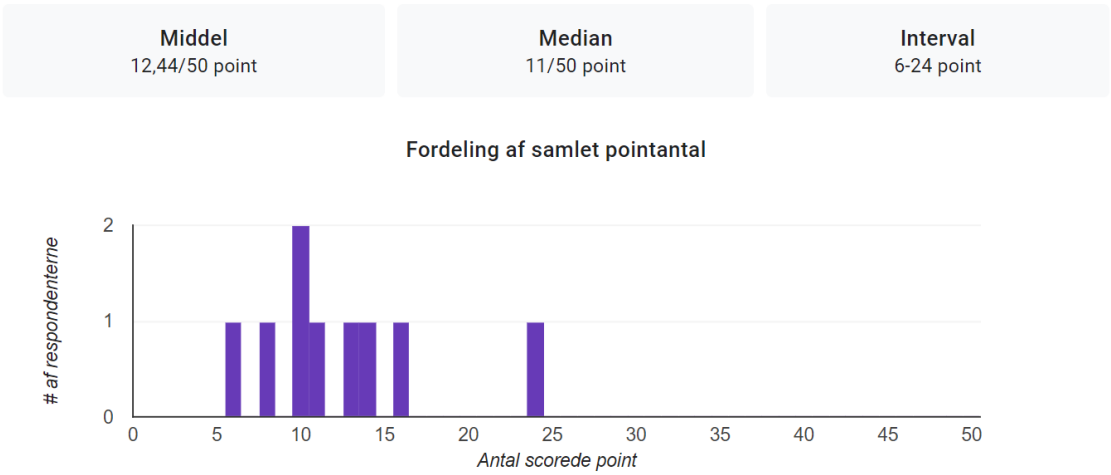


Figure 5: *Results from questionnarie. The 9 participants, Interval, mean, median, correct answers.*

v. Computational time

We measure the computational time on computer1 and on computer2.

Computational time		
	61 epochs	100 epochs
LSTM	9 min 35 sec 18 min 32 sec	15 min 55 sec 30 min 56 sec
CNN	20 min 30 sec 42 min 10 sec	33 min 30 sec 1 h 9 min 50 sec

Table 2: *Computational time. Training time of the LSTM and the CNN models for two different computers.*

vi. Average accuracy and loss

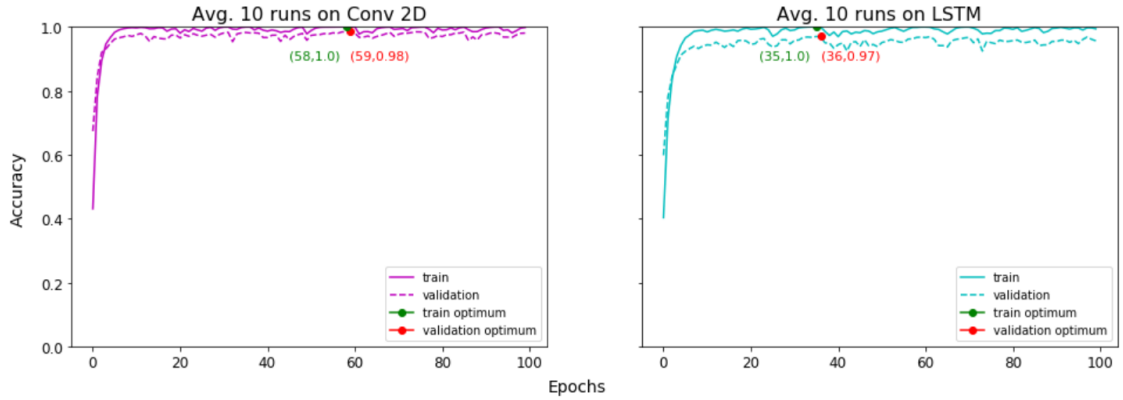


Figure 6: *Avg. over 10 runs, validation accuracy, both the CNN and the LSTM.*

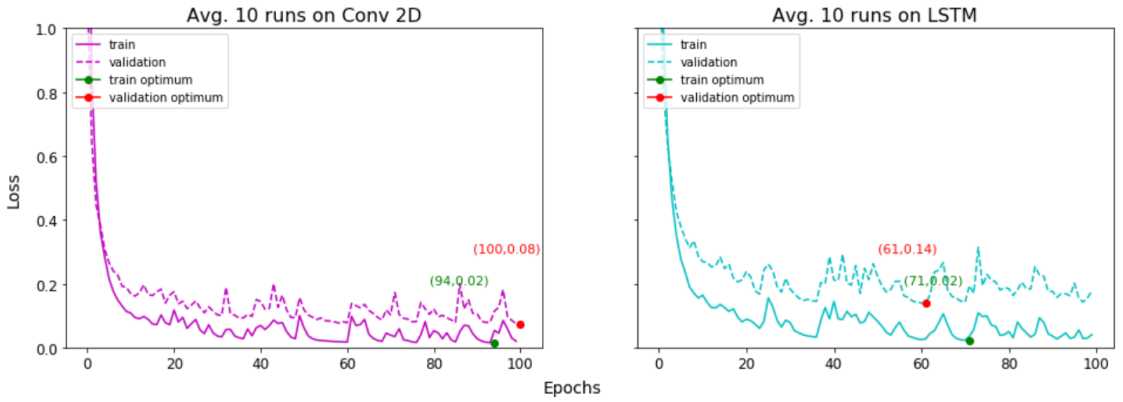


Figure 7: *Avg. over 10 runs, validation loss, both the CNN and the LSTM.*

vii. CNN and LSTM correctly classify all 50 files, confusion matrix

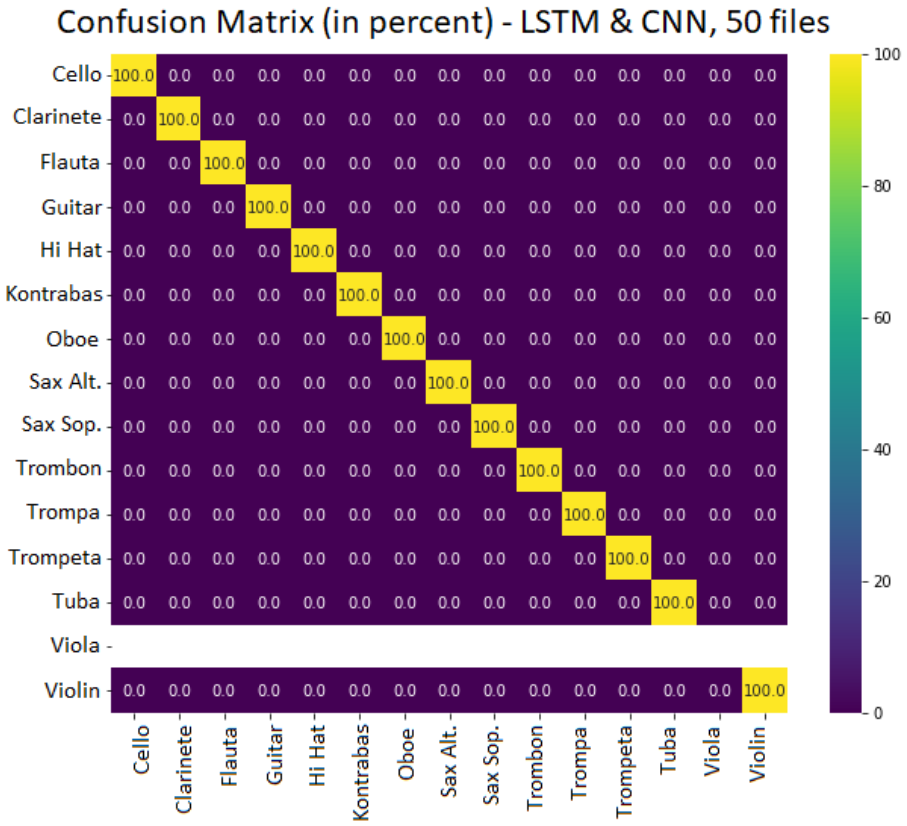


Figure 8: *Confusion Matrix, the CNN and LSTM, 50 files.*

viii. Confusion matrixes already presented in the report

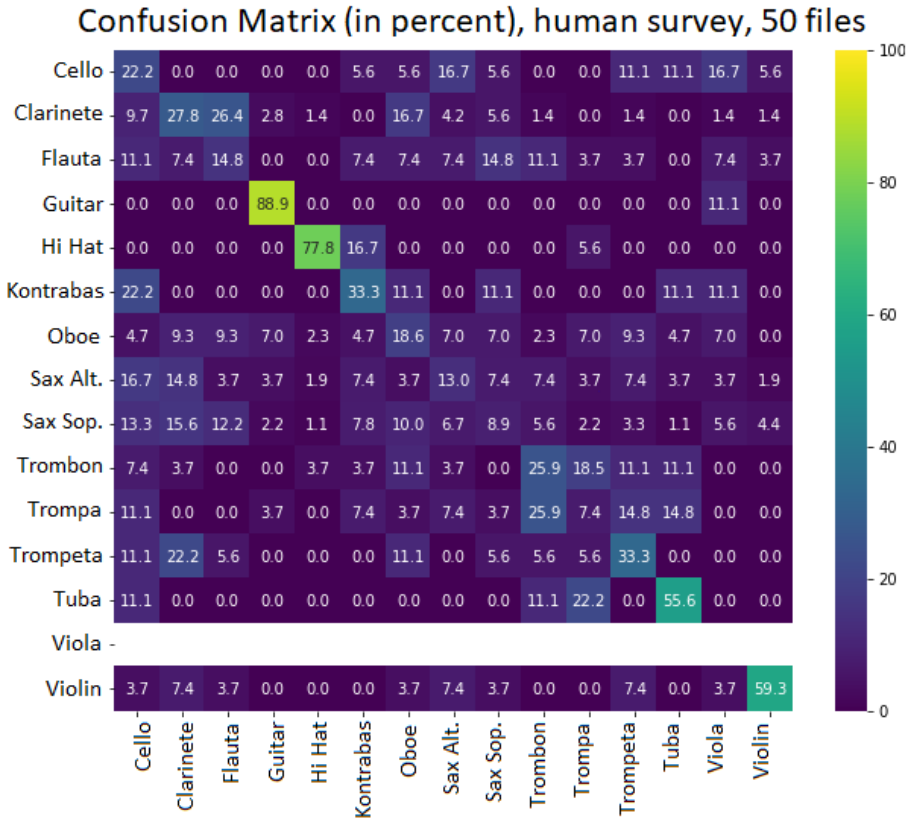


Figure 9: *Confusion Matrix, human survey, 50 files.*

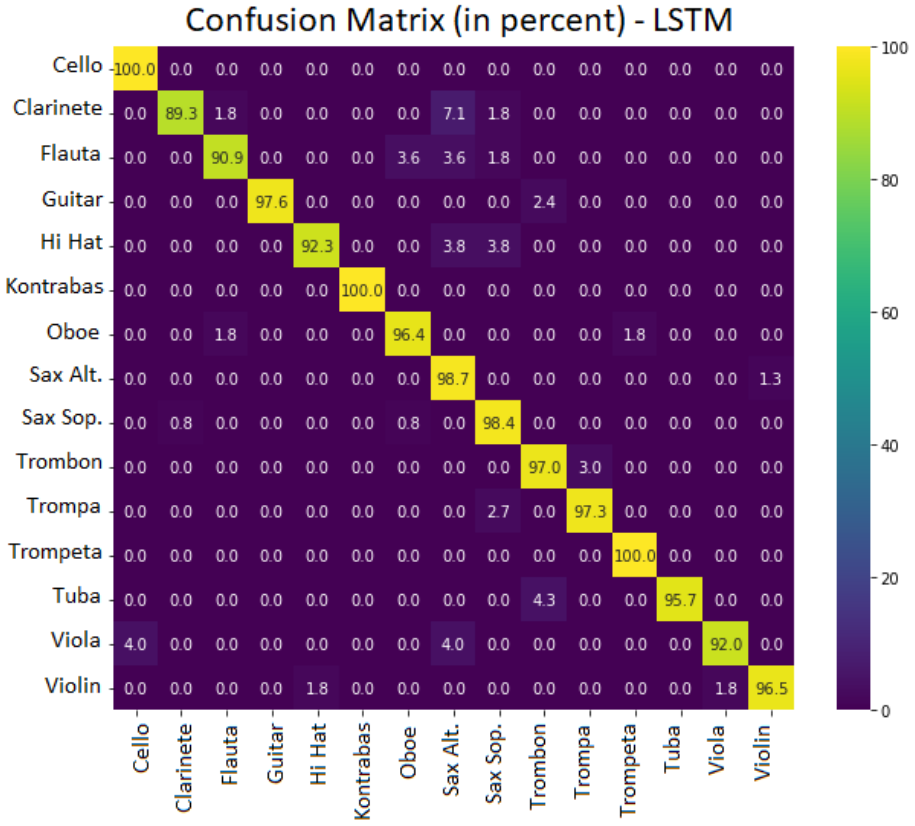


Figure 10: *Confusion Matrix, LSTM network, 692 files.*

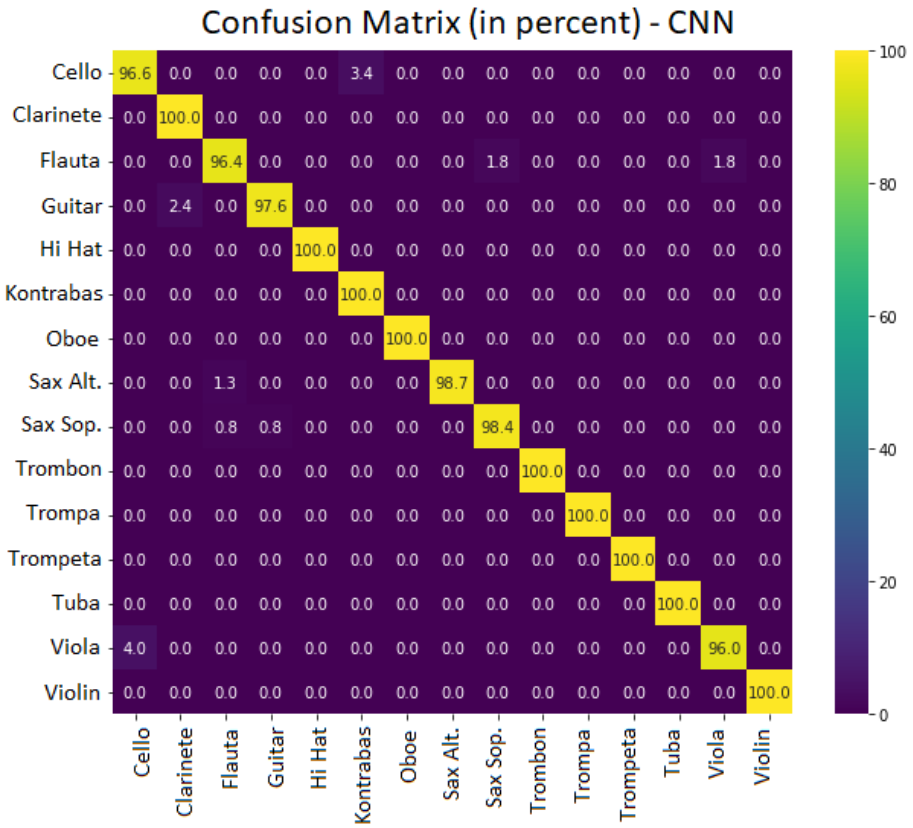


Figure 11: *Confusion Matrix, CNN network, 692 files.*