NEURAL MODELS FOR COMPUTER-ASSISTED MUSICAL ORCHESTRATION: A PRELIMINARY STUDY

First Author
Affiliation1
author1@ismir.edu

Second Author
Retain these fake authors in submission to preserve the formatting

Third Author
Affiliation3
author3@ismir.edu

ABSTRACT

In this paper we will explore preliminary neural models for the task of computer-assisted musical orchestration. After an introduction on the problem, we will show how we decided to model musical orchestration as a classification task. In this context, we will propose two deep learning models and we will show how they perform as classifiers for musical instruments recognition by comparing them with specific baselines. We will the show how they perform, both qualitative and quantitative, in the task of computer-assisted orchestration by comparing them with state-of-the-art systems. We will highlight, finally, benefits and problems of neural approaches for orchestration and we will propose possible future steps.

1. INTRODUCTION

Computer-assisted orchestration is an important problem ... **Carmine can you add more?**.

We attempt to use neural networks to solve the assisted orchestration problem. Our idea is to frame this as a classification problem, and then use these classifiers to perform orchestrations. Our classifiers will be trained on sounds that are combinations of multiple instruments and notes, leading to a model that can take a complex mixture of timbres and deconstruct it into individual instrument-note pairs. In our setting of the problem, each class is an instrument-note pair. When a new sound, one that is not simply a mixture of notes, is fed to the model, the predictions are used to create an orchestrated solution.

We found that even on a simplified version of this classification problem, parametric classifiers were unable to meaningfully identify instrument-note pairs as the number of instruments used in the combinations increased. Therefore a need for neural networks arose. The two architectures we experimented with are a Convolutional Neural Network with a Long Short-Term Memory unit, and ResNet, a well known residual network. After training these models on combinations of 10 instruments, we orchestrated 15 target sounds by feeding them to the networks. We then compared our orchestrations to OrchIdea's solutions for the same targets.

2. DATA

2.1 Data Source and Generation

We used the TinySOL database to create our input data. TinySOL is a subset of the Studio On Line (SOL) database created by IRCAM **Reference needed**. TinySOL contains 1,529 samples from 12 instruments. The instruments come from different orchestral families: strings, woodwinds, and brass. Each sample is one instrument playing a single note in the ordinario style, with one of three dynamics: *pp*, *mf*, or *ff*. The instruments and ranges over which they were recorded is summarized in Tab.1.

We generated all the data we used to train our models. For a chosen N, each input to our model is a combination of N TinySOL samples. An orchestra of N instruments is chosen and the data is generated in the following way:

- For each instrument in the orchestra, a random TinySOL sample for that instrument is chosen. This leads to a random combination of both notes and dynamics.
- The chosen samples are combined to be played simultaneously and normalized by the number of instruments.
- The mel spectrogram of the mixture is computed to be used as the features input to the model.

The choice of using the mel spectrogram as input features of the model is very commmon in music information **{Insert reference}** and can be considered to be an accurate representation of the way humans perceive sound and music.

Samples had a sample rate of 44100Hz and were all padded or chopped to be exactly 4 seconds long. The Mel spectrograms were then generated using an FFT hop length of 2048 samples (e.g. the window of each FFT was 46ms wide), and a number of Mel bins equal to 128. Therefore, all the features fed to the model were matrices of size 128×345. We used Librosa package in Python to compute the features, and more details on the exact computations can be found in **{Reference to Librosa}** The choice of the hop length was made by doing a compromise between the amount of information extracted by each FFT window, and the ability to capture changes in the dynamic, assuming that no change faster than 10ms could be perceived by the human ear.

Instrument	Abbreviation	Range
Violin	Vn	G3-E7
Cello	Vc	C2-C6
Viola	Va	C3-C7
Trumpet C	TpC	F#3-C6
Trombone	Tbn	A#0;A#1-C5
Oboe	Ob	A#3-A6
Horn	Hn	G1-F5
Flute	Fl	B3-D7
Clarinette Bb	ClBb	D3-G6
Contrabass	Cb	E1-C5
Bass Tuba	BTb	F#1-F4
Bassoon	Bn	A#1-D#5

Table 1. Table containing the pitch ranges in the TinySOL database, used for training. Hyphen means all the notes between the two explicited notes.

2.2 Data Augmentation

In order to increase variability in our generated data, we used two methods of data augmentation.

The first one was done on the soundfile resulting from the mixture of the samples, before the mel spectrogram was taken. We performed a slight pitch shift by shifting the frequency of the signal. We did this by ... **Alejandro, can you add how you did the pitch shift (and maybe explain why changing the SR leads to a pitch shift)**In practice, the sample rate used to compute the Mel spectrogram was modified to be within 5% of the actual 44100Hz.

The second augmentation was performed on the feature matrix itself, the mel spectrogram. We chose random columns and rows of the matrix to zero out. For a given matrix, each column and each row had individually a 1% chance to be set to 0, which yielded an average of 1.28 columns and 3.45 rows being zero-ed out. This method of data augmentation aimed to be more resilient to the possible errors of recording, due either to a setup that induces interferences and cancels specific frequencies, or to a default in the microphone that fails to record a small temporal portion.

3. OUR MODEL

Our idea is to perform orchestration using deep neural networks trained for classification. In our setting, each class is an instrument-note pair. Our model takes in the mel spectrogram of a soundfile that is a combination of different instruments and pitches, and attempts to identify which instrument-pitch pairs are present. The network learns how to take a sound that is a complex combination of notes and timbres and deconstruct it into its original parts. Then, when a network is given a target sound as input, it will attempt to apply the same classification rules, outputting a vector of the probabilities of each sample being in the target sound. By taking the samples that have the highest probability, an orchestrated solution can be created. Both of our deep models use the Adam optimizer, Binary Cross-Entropy loss, and were written using PyTorch.

We did not train our model to classify the dynamics of a sample despite TinySOL having pianissimo, mezzo forte, and fortissimo recordings for each sample. If we had, then each class would be an instrument-pitch-dynamic tuple and would have only one data point. Instead, when the model is used for orchestration, the dynamic is determined by the probability of that sample as output by the model. If the model output a probability higher than 0.66 for a sample, the fortissimo version of the sample was used. If it was between 0.33 and 0.66, then the mezzo forte version, and if less than 0.33 the pianissimo dynamic. The idea behind this is that samples that are the most represented in the target should be the loudest in the orchestrated solution. **{I am not sure this hypothesis can be used accurately}**

3.1 Baseline

In order to have a baseline to compare our results against, we attempted to solve the classification problem using various parametric classifiers. The classifiers we tested were Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors. We used the implementations provided in the scikit-learn library for Python. For SVM, we used SVC with an RBF kernel. For Random Forest, we set the maximum depth of each tree to be 15. Each classifier was wrapped in a MultiOutputClassifier to achieve the multi-label nature of this problem. We found SVM to have the highest accuracy of the three classifiers across all experiments. All of the following baseline experiments used 50,000 generated samples with a train-test split of 60/40. Each sample is a combination of one or more instruments and is four seconds in length. The features used are the mel frequency cepstral coefficients (MFCCs) of the resulting combination, with a total of 100 coefficients per sample. We chose to use MFCCs for this setting because using the mel spectrogram leads to a number of features that is too high for the parametric classifiers.

We started by simplifying the problem to classifying only the instruments and not the pitch. This had the benefit of both reducing the number of classes and increasing the number of samples per class. We found that SVM was able to very accurately identify the instrument given an input that had only one instrument present; for this case the accuracy was 99.8%. However as soon as multiple instruments were present in the input, the accuracy dropped significantly. With two instruments, accuracy was 55.4%, with three it was 19.6% and with 10 instruments the accuracy was 0.5%.

To better approximate the problem of identifying instrument and pitch, we then attempted to classify the instrument and pitch class. That is, which octave the pitch was in did not matter, only the pitch class. The input was a combination of two instruments drawn from a possible twelve instruments, and the classifier attempted to identify which instruments were present and for two of those instruments, say Flute and Violin, which pitch classes were present. If another instrument was present that was not Flute or Violin, the classifier would attempt to identify that instrument, but not its pitch class. The best results from this setting of the problem was SVM with 30.2% accuracy, which was

Instr. 1	Instr. 2	SVM Accuracy	RF Accuracy
Violin	Flute	38.8%	9.8%
Violin	Trumpet	33.8%	9.1%
Violin	Cello	34.8%	6.3%
Cello	Viola	32.1%	5.8%
Oboe	French Horn	**39.9%**	**17.5%**

Table 2. Comparison of results between SVM and Random Forest. Each data point is a combination of two TinySOL samples, where at least one of the samples is from one of the two instruments specified for that experiment. For the samples drawn from one of the two instruments, the pitch class of that sample is identified. For a sample not from one of the two instruments, the classifier simply attempted to identify that a sample from one of the non-specified instruments was present. In this setting, there are 25 classes: 24 from the 12 pitch classes from 2 instruments, and 1 class that specified whether an instrument that is not one of the two specified is present.

a result of classifying the pitch class of Flute and Violin. Depending on which two instruments had their pitch class identified, the accuracy varied greatly. For a Violin and Cello accuracy was 20.9%, and for Trombone and Bass Tuba accuracy was 2.2%.

For a slight modification on this setting, we no longer attempted to identify which instrument was present if it was not one of the two instruments whose pitch class was being identified. Instead, the classifier would simply identify that an instrument that was not one of the two was present. Since this is a simpler problem, it lead to increased accuracies. The accuracies from this experiment are outlined in Table 1. Since Random Forest performed worse than SVM is every experiment, we stopped testing with Random Forest and used SVM from that point on.

We then performed this same experiment with three instruments having their pitch class identified. Each input was a combination of three instruments drawn from a possible twelve instruments. For three instruments specified in that experiment, pitch class was identified. Flute, Oboe, and Violin reached an accuracy of 11.1%, and Bass Tuba, Trumpet, and Trombone was 0.5%. As we increased the number of instruments whose pitch classes was being identified, the accuracy continued to drop. For classifying the pitch class of four instruments: Oboe, French Horn, Violin, and Flute, the accuracy was 2.7%.

This was still a simplified version of the problem, as we were identifying only the pitch class of a few instruments. However, the parametric classifiers were unable to achieve accurate results as the number of instruments increased. Therefore, we did not attempt the full setting of the problem, in which individual pitches are classified for all instruments, with parametric classifiers.

3.2 CNN with LSTM

The first deep model we tested was a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM) unit. CNNs show good performance on classification problems for their ability to extract spatial features. **Reference needed**. Long Short-Term Memory (LSTM) units provide a way to learn long term dependencies in the data [**?**], which is relevent given the sequential nature of audio.

Our architecture contains four convolutional layers and two fully connected layers. Each convolutional layer is followed by a BatchNorm layer, a ReLU activation layer and a 2×2 MaxPool layer with a stride of 2. The kernel size is 3×3 with a stride of 1 and a padding of 1. The number of filters are 8, 16, 32, and 32.

Following the first three convolutional layers, there is an LSTM layer with 32 outputs. After the LSTM layer, there is a final convolutional layer yielding a tensor of dimensions $32 \times 8 \times 21$. We flatten the outputs and feed them into fully a connected layer with Dropout, then another another fully connected layer. Finally, the sigmoid function is applied to the final layer.

Since each class is independent, we are able to take the sigmoid activation and use binary classification for each class. The model architecture is shown in Figure 1.

3.3 ResNet

The second deep model that we used was the well known deep residual network ResNet.

To make our setting more reasonable, we took the well-known deep residual network (ResNet) as backbone in our experiments. Specifically, we used 18-layer ResNet, which allows information to pass directly from the input to the final layer of each block. Besides, to make the model more suitable to our problem, we reduce the number of parameters and reset the output channel numbers of each block with 32, 64, 32, 32 respectively.

3.4 Classification results

During training, the loss function used to optimize the inner parameters of the model was binary cross entropy, as it is the common choice for multiclass multilabel classification frameworks such as this one. However, the value of the loss function alone is difficult to interpret. For this reason we created a complementary function f to be used only for evaluation. This function compares a vector of expected outputs $\overline{X}$ with the estimated output from the model $\hat{X}$ by using the following function

$$f(\overline{X}, \hat{X}) = \frac{1}{N} < \overline{X}, M_N(\hat{X}) > \quad (1)$$

where

$$M_N(\hat{X})_i = \begin{cases} 1 \text{ if } i \in I_N(\hat{X}) \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

and $I_N(\hat{X})$ is the set of indices containing the N first maximums of the vector $\hat{X}$.

More specifically, the function $M_N(\hat{X})$ takes as an input a vector of probabilities and outputs a vector where only the positions of the N first maxima are set to 1. This new vector would be the orchestration of N instruments given by the model. Thus, the function f simply outputs

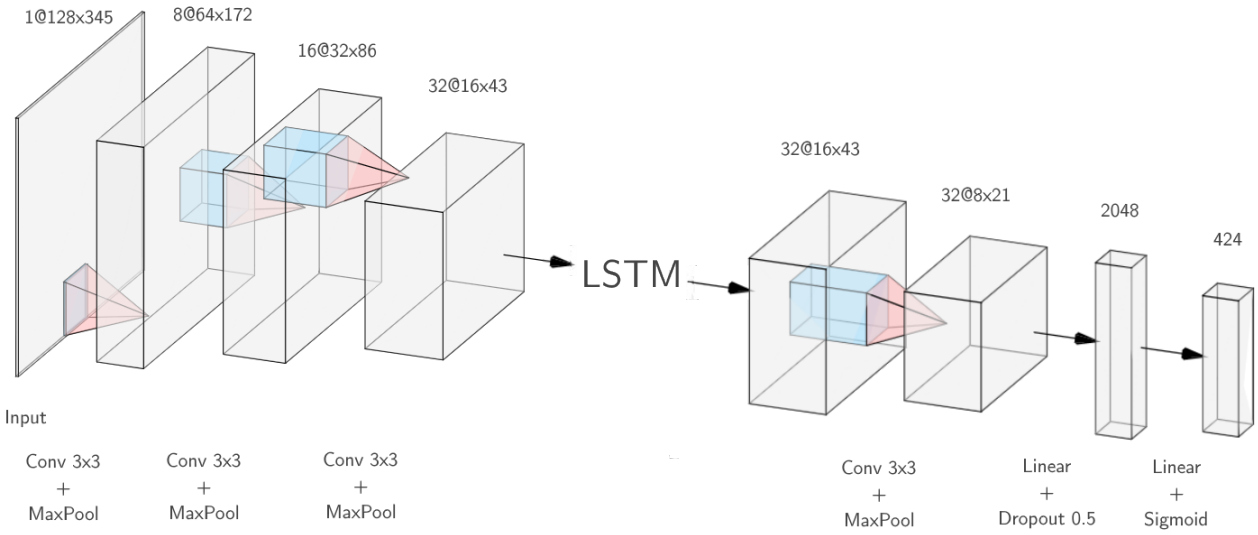


Figure 1. Diagram of the CNN with LSTM architecture.

N	Orchestra
4	Hn, Ob, Vn, Va
5	Hn, Ob, Vn, Va, Vc
6	Hn, Ob, Vn, Va, Vc, Fl
7	Hn, Ob, Vn, Va, Vc, Fl, Tbn
8	Hn, Ob, Vn, Va, Vc, Fl, Tbn, Bn
9	Hn, Ob, Vn, Va, Vc, Fl, Tbn, Bn, TpC
10	Hn, Ob, Vn, Va, Vc, Fl, Tbn, Bn, TpC, ClBb

Table 3. Orchestra for each N used in the experiments.

Model	N	Nb epochs	Nb samples
CNN-LSTM	4-7	50	100000
CNN-LSTM	8-10	50	200000
ResNet	4-8	20	200000
ResNet	9-10	20	400000

Table 4. Number of epochs and number of samples per epoch used for training each model with a given value of the size of the orchestra N.

the proportion of the estimated orchestration that matches the expected one.

Different experiments were made by varying the number of instruments in the orchestra N but always matching this number with the number of instruments in each mixture sample. Tab. 3 shows the orchestra for each different value of N used in the experiments.

(Alejandro, add how there were certain instruments that performed poorly and some that were great. Add how ResNet was awful for Flute and always output Flute C7)

Then, for both CNN with LSTM and ResNet, we computed the maximum accuracy over the epochs. Tab. 4 compiles the number of samples per epoch and the total number of epochs used for each model and each value of N. Fig. 2 shows the best overall test accuracy achieved by both models across the number of instruments N to detect simultaneously.

From this plot, we see that ResNet achieves better performances regardless of the number of instruments used in the combination, with

For more details on the accuracy Fig. 3 and Fig. 4 show the maximum accuracy computed using the function f explicited in (1).

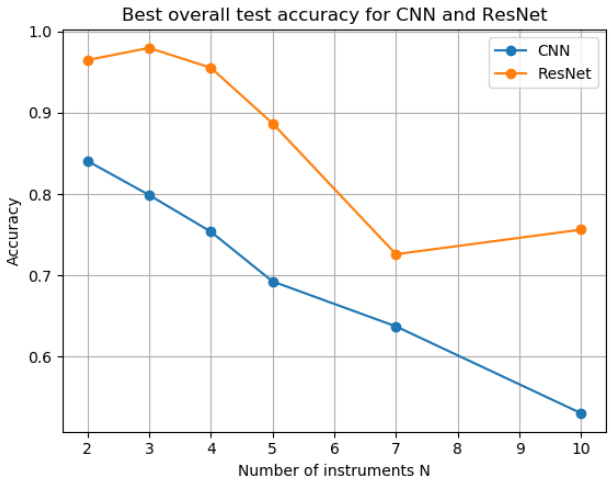


Figure 2. Best overall accuracy and for CNN and ResNet depending on the number of instruments in the combinations used for training.

4. ORCHESTRATION EXPERIMENTS

Our final model was trained on data where each input was a combination of ten instruments, but we performed experiments with varying numbers of instruments used in combination. (insert table of data showing the results of CNN with 2,3,4 etc instruments) An arbitrary number of samples can be used in the solution, since the n highest probabilities can be taken from the output, leading to n samples in the solution.

We tested our models using 15 targets for orchestration. Two of the targets were made from TinySOL samples, but the rest are not combinations of input data, and some are not even recordings of instruments. Among the targets are recordings of bells, a car horn, a gong, and a recording of a boat docking. By passing these samples into a model, an orchestration of the target is created from the TinySOL samples that were used to train the network. For orchestrating these targets, we used both the CNN and ResNet models, each trained on combinations of 10 instruments from a possible 10 instruments. The CNN was trained for 49 epochs and the ResNet for 24.

Although the data we trained the model on was fixed to four seconds of audio, the target samples can be of arbitrary length. By changing the mel hop length, which is the distance between the frames of the melspectrogram, an audio signal of any length can be represented as a matrix of the same size as the training data.

4.1 Evaluation

We evaluated our orchestrations both qualitatively and quantitatively. Qualitative inspection was done through an acoustic inspection of the solution, paying close attention to timbre and pitch. For targets that had harmonic content, it was noted if the partials present in the target were also represented in the orchestrated solution. For example, one of the samples of a bell had partials that represented a C sharp minor chord. For a target that contained specific notes, the solution from ResNet contained both the note in the target and its partials: a sample that was two octaves higher, the fifth an octave up, and a minor third two octaves up (mix_ObA4_BnC3).

For quantitative evaluation, we used the distance metric defined in (3) to calculate the distance between the target samples and our orchestrated solutions. We then orchestrated the same targets with OrchIdea and calculated the distance for OrchIdea's solutions. A comparison of these results is in Tab. 4.1.

In order to have comparable solutions, we did not allow OrchIdea to create sparse solutions, meaning that OrchIdea was forced to use all 10 instruments in each solution. This creates a more fair comparison since our model is optimize solutions by using fewer than 10 samples.

$$d(x,\tilde{x}) = \lambda_1 \sum_k \delta_{k1}(x_k - \tilde{x}_k) + \lambda_2 \sum_k \delta_{k2}|x_k - \tilde{x}_k| \quad (3)$$

$$\delta_{k1} = \begin{cases} 1, \text{if } x_k > \tilde{x}_k \\ 0, \text{otherwise} \end{cases} \quad (4)$$

Model	Average Distance
CNN with LSTM	
ResNet	
OrchIdea	

Table 5. Comparison of the orchestration solutions of our two models, CNN with LSTM and ResNet, against OrchIdea's solutions. The distance between the target and solution are calculated for each target using Eqn. (3) and then averaged.

$$\delta_{k2} = \begin{cases} 1, \text{if } x_k < \tilde{x}_k \\ 0, \text{otherwise} \end{cases} \quad (5)$$

This metric takes in the FFT of the target, x and the FFT of the solution $\tilde{x}$ and computes the difference between them at each bin.

5. CONCLUSION

- the approach seems promising for orchestration - many things used in Orchidea are not implemented here (symbolic constraints, sparsity,...) - CNN seems better for timbre - ResNet seems better for pitch (what are timbre and pitch??)

Based on our accuracies from training the model and the results of orchestrating 15 targets, our approach seems promising for orchestration. When comparing our method with OrchIdea, it is important to note that many of processes OrchIdea uses to optimize its solution were not used in our model. This includes important aspects such as sparsity, partial filtering, **(Carmine any more to add?)**. Our model also lacks the implementation of symbolic constraints, which is an important part of assisted orchestration.

We find that the CNN and ResNet give similar accuracies during training, but perform differently when tasked with orchestrating targets. Overall, CNN seems to better emulate the timbre in its orchestrations, where ResNet is better for recreating the pitch(es) of the target.

5.1 Interpreting the Latent Space

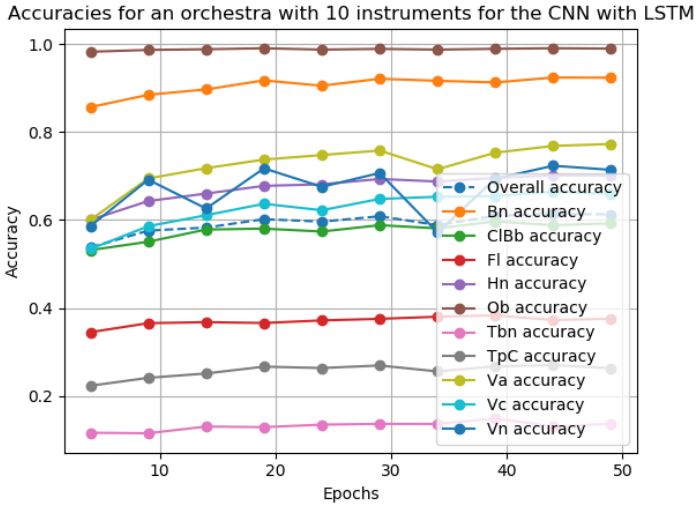
- the system finds filters like...

6. FUTURE STEPS

- using conditioning to impose symbolic constraints - variable size solutions - joint networks for orchestral size detection and orchestral families (see paper)

Future steps in this project include implementing various methods that are present in OrchIdea.

Our current model orchestrates all targets using the same number of samples, and this does not take into account the density of different targets. The solution to this is to allow sparse solutions in which the model decides how many samples should be used to best represent the target. This allows a small number of samples to be used for



sonically sparse sounds and many to be used for sonically dense sounds.

Partial filtering is a method that would aid our model in orchestrating the harmonics of a target. The dominant harmonic partials of the target are identified, and the search space is limited to only include samples of those pitches. For example, if the target is a recording of an instrument playing a C4, then the partials identified may be C4, C5, G5, and E6. The model would then only consider samples of these pitches to be used in the solution. This leads to a solution whose harmonics are much closer to the target, which is an important part of aural similarity.

7. CITATIONS

All bibliographical references should be listed at the end, inside a section named "REFERENCES," numbered and in alphabetical order. All references listed should be cited in the text. When referring to a document, type the number in square brackets [**?**], or for a range [**?**, **?**, **?**].

When the following words appear in the conference publication titles, please abbreviate them: Proceedings → Proc.; Record → Rec.; Symposium → Symp.; Technical Digest → Tech. Dig.; Technical Paper → Tech. Paper; First → 1st; Second → 2nd; Third → 3rd; Fourth/nth → 4th/nth.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of [**?**]," not "In our previous work [**?**]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous."

8. FIGURES

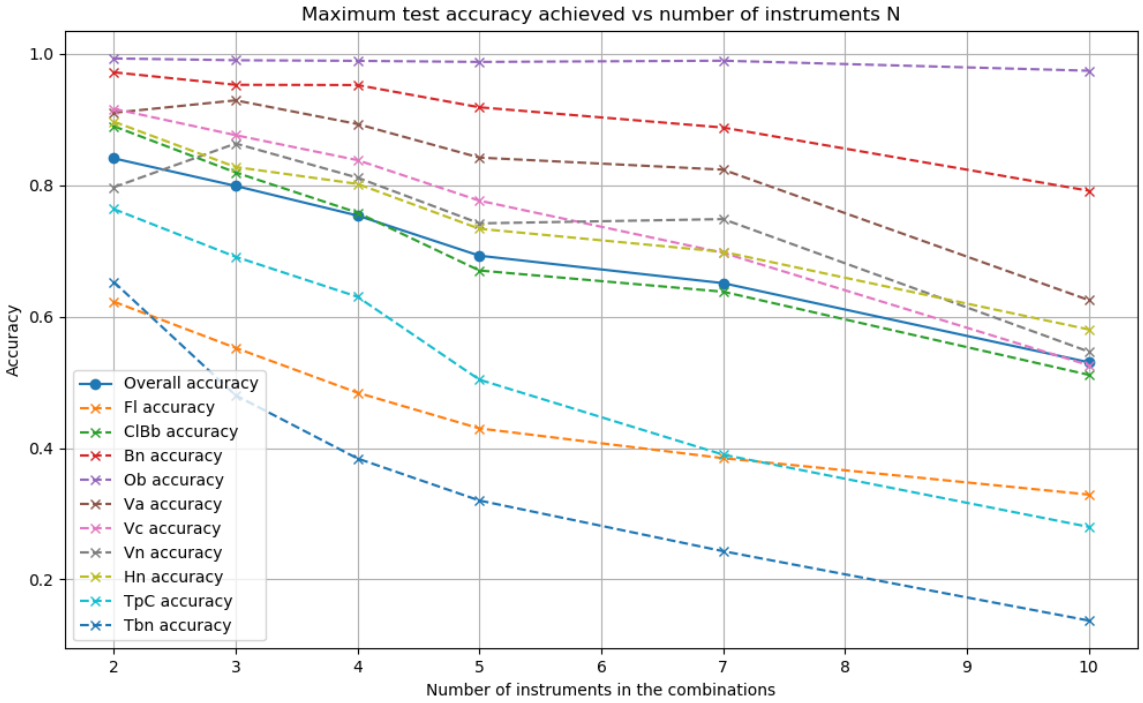


Figure 3. Best overall accuracy and for each instrument obtained by the CNN with LSTM depending on the number of instruments in the combinations.

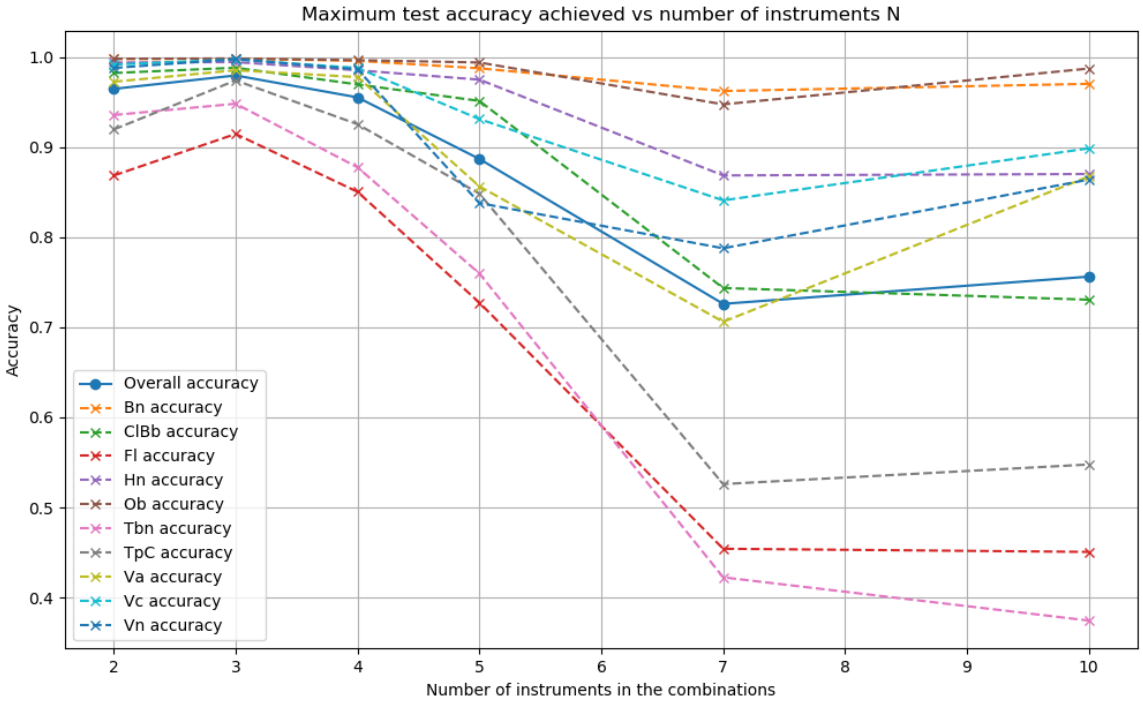


Figure 4. Best overall accuracy and for each instrument obtained by the ResNet depending on the number of instruments in the combinations.