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

First author[1] **Second author**[1] **Third author**[2]
Fourth author[3] **Fifth author**[2]
[1] Department of Computer Science, University , Country
[2] International Laboratories, City, Country
[3] Company, Address
CorrespondenceAuthor@ismir.edu, PossibleOtherAuthor@ismir.edu

ABSTRACT

In this paper we will do a preliminary exploration on how neural networks can be used for the task of target-based computer-assisted musical orchestration. We will show how it is possible to model this musical problem as a classification task and we will propose two deep learning models. We will show, first, how they perform as classifiers for musical instrument recognition by comparing them with specific baselines. We will then show how they perform, both qualitatively and quantitatively, in the task of computer-assisted orchestration by comparing them with state-of-the-art systems. Finally, we will highlight benefits and problems of neural approaches for assisted orchestration and we will propose possible future steps.

1. INTRODUCTION

The development of computational tools to assist and inspire the musical composition process constitutes an important research area known as *Computer-Assisted Composition (CAC)* [1, 2]. Within CAC, target-based computer-assisted orchestration is an important example of how machine learning can be used for enhancing and assisting music creativity [3].

Target-based computer-assisted orchestration can be thought of as the process of searching for the best combinations of sounds in a database of instrumental samples, to match a target sound under a specific similarity metric and specific sets of constraints. A solution to this problem is a set of orchestral scores that represent the mixtures of audio samples in the database, ranked by similarity with the target sound. Valid orchestral scores may contain several instruments sounding simultaneously, selected out of a large number of possible combinations of sounds from the database.

The approach studied in [4] consists in finding a good orchestration for any given sound by searching combinations of sounds from a database with a multi-objective optimization heuristics and a constraint solver that are jointly optimized. Both the target sound and the sounds in the database are embedded in a feature space defined by a fixed feature function and each generated combination of sounds is evaluated by using a specific metric. This method has been substantially improved in [8, 20] and is implemented in the *Orchidea* toolbox for assisted orchestration (`www.orch-idea.org`), currently considered the state-of-the-art system for assisted orchestration.

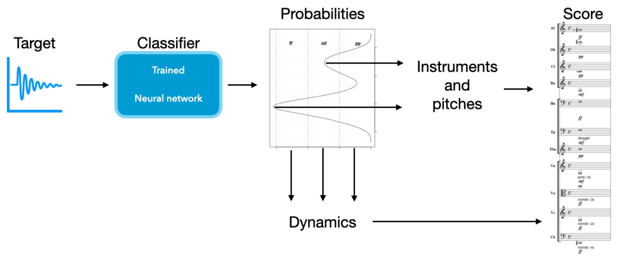

Figure 1. An overview of the proposed method for assisted orchestration with neural models. Instruments and pitches are determined as peaks of the output probability distribution, while the dynamics are computed by quantizing the probabilities.

In this paper, we try a different approach to this problem by experimenting with deep neural architectures. The main idea is to train a model to classify combinations of real instruments and use it then for orchestration. A typical solution for assisted orchestration is a set of triples *instrument-pitch-dynamics* such as {`Flute C6 pp, Bassoon C4 mf, Bassoon G4 ff`}. By training a neural network with real combinations of instrumental notes, it will acquire the ability to identify the presence of each instrument and its associated pitch by building the appropriate latent representation. Thus, when an unknown target sound is given as input, the network will identify which are the best instruments to match the target sound, and it will be able to deconstruct a complex mixture of timbres into individual instrument notes.

This method is motivated by the good results obtained in previous research on musical instruments identification [5, 14] and the more recent use of deep neural networks for musical classification [7, 15].

In this paper we do preliminary experiments with two

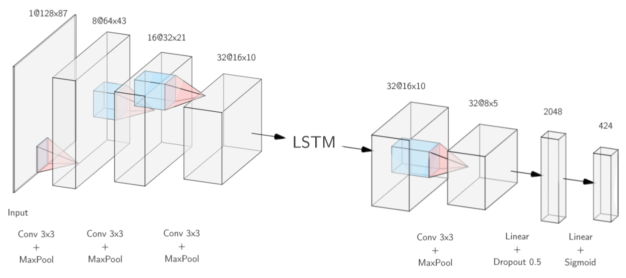

Figure 2. Diagram of the CNN with LSTM architecture.

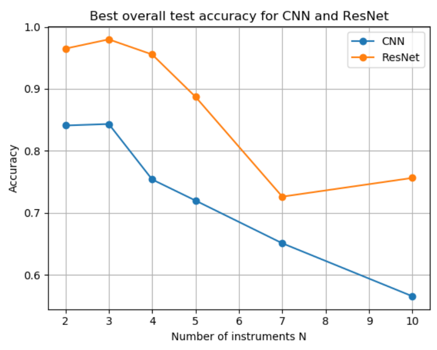

Figure 3. Best overall accuracy for CNN with LSTM and ResNet depending on the number of instruments in the combinations used for training.

deep architectures: a convolutional neural network (CNN) with a long short-term memory (LSTM) unit in the middle and *ResNet*, a well known residual architecture that already yielded good results for image classification [11]. We chose to use a CNN because of its success in audio classification [12] and we decided to include an LSTM unit in it because of its ability to learn long term dependencies in data [13], which is important given the temporal nature of audio.

After training these models on varying combinations of 10 instruments, we used them for orchestration. More specifically, we orchestrated 15 target sounds by feeding them to the networks and synthesizing a solution from the most likely samples output from the system. We then compared our orchestrations to Orchidea's solutions for the same targets, through both qualitative and quantitative means. For quantitative comparison, we computed a specific distance metric on the FFTs of the target and solution, comparing the distance between Orchidea's solution and the target and our model's solution and the target.

Next sections will be as follows: section 2 will describe the dataset we used for experiments and the models we trained as classifiers, showing the classification results. Sections 3 and 4 will present, respectively, the results of orchestration experiments using the trained models and the conclusions of our preliminary study on neural approaches for target-based computer-assisted orchestration.

The codebase for this paper and some audio examples can be found at: `https://github.com/dzluke/DeepOrchestration`.

2. NEURAL MODELS

2.1 From orchestration to classification

In this paper, we model assisted orchestration as a classification problem. The general methodology is as follows:

1. we train specific models to classify the instruments present in combinations of sounds from a database of instrument notes, up to ten simultaneous instruments;

2. we then pick the best classifier and we feed into it an unknown sound to be classified;

3. since the output of the classifier will be in the form of the probability that specific instruments are present in the sound, we use this information to synthesize an orchestration for the target sound;

4. finally, we evaluate the generated orchestration against state-of-the-art systems for computer-assisted orchestrations.

In other words, the classifiers learn how to take a complex combination of pitches and timbres and deconstruct it into its original parts.

A complete orchestration solution, however, would normally be made by triples of instrument-pitch-dynamics. It is difficult to frame this problem as classification, since we would need to have a very high number of classes. Moreover, for the nature of the samples we use (a typical sample is in the form `Flute-C4-pp`, as described in 2.2), each class would be represented by a single sample. For these reasons, our models have been trained on simplified versions of the original problem and we used different strategies to provide the remaining information needed to generate orchestrations.

In order to have a baseline to compare our classification results and to get a sense on the complexity of the problem, we used various parametric classifiers along with the neural models.

Three baseline classifiers have been trained to recognize either the instrument only for a total of 12 classes (the instruments in TinySOL) or the pair instrument-pitch *class* (a pitch class is the pitch of a note without the corresponding octave, e.g. C, D#, G, etc.) for a total of 25 classes (we will classifiy combinations of 2 instruments for 12 pitch classes each, plus one extra *trash* class, see section 2.4).

Two neural classifiers, instead, have been trained on the more complex problem of recognizing instrument-pitch pairs, for a total of 424 classes (i.e. the neural classifiers would classify instruments and pitches with respective octaves, hence 12 instruments for 12 pitch classes and three dynamics; note that not all the samples in TinySOL have all dynamics). We want to point out, however, that this is still a simplified problem compared to assisted orchestration. Even in the case of instrument-pitch pair classification, indeed, we would miss the dynamics of the note. To cope with this problem, in the orchestration phase, we quantize the probability functions generated by the models. 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. Since the classes only encode instrument-pitch pairs and since we also need the dynamics, we quantize the probability function generated by the network into three echelons coresponding, respectively, to the dynamic levels *pianissimo*, *mezzoforte* and *fortissimo* (see section 3). By taking the instruments and pitches that have the highest probability and by using quantized probabilities to retrieve dynamics, a full orchestrated solution can be created. Figure 1 illustrates the described ideas.

2.2 Dataset

To create the input data for training the classifiers, we used the *TinySOL* database. TinySOL is a subset of the Studio On Line (SOL) database created by IRCAM [9]. 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* playing style, with one of three dynamics: *pp*, *mf*, or *ff* (for example `Flute-C4-pp` or `Clarinet-D5-mf`).

For a given number of instruments N, each input to our models is a combination of N TinySOL samples chosen among an orchestra of 10 instruments. The data is generated in the following way:

- N random TinySOL samples are selected, leading to a variety of instruments, pitches, and dynamics; we did not allow the same instrument to be chosen more than three times in order to ensure variety in the mixtures;
- the chosen samples are combined to be played simultaneously and normalized by the number of instruments;
- the Mel frequency cepstral coefficients (MFCCs) and the Mel spectrogram of the mixture are computed to be used as the input features for the baseline classifiers and the neural classifiers respectively.

The resulting combination has a sample rate of 44100Hz and is padded or trimmed to be exactly 4 seconds long.

The choice of using the Mel spectrogram and MFCCs as input features for classification models is common in music information retrieval [19] and can be considered to be an appropriate representation of sound and musical signals. The Mel spectrogram is generated using an FFT hop length of 2048 samples (the window of each FFT was 46ms wide), and a number of Mel bins equal to 128. Therefore, the Mel features fed to the model were matrices of size 128×345. We used the Librosa package in Python to compute the features; more details on the exact computations can be found in [16]. 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 would be allowed.

2.3 Data augmentation

In order to increase variability in the generated data for the neural models, we also used two methods of data augmentation as described in [6, 18]; more specifically, we used pitch shifting and partial feature dropout.

The former has been directly applied on the TinySOL samples each time they are selected to generate a new combination. We performed a small pitch shift by reading the samples with different sample rates: a small difference in sample rate will slightly modify the duration and the perceived pitch if played at the normal sample rate. In practice, the sample rates used for this data augmentation were within 5% of the actual 44100Hz.

The latter was performed on the feature matrix itself of input samples, 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 variations in the recording of the instruments.

2.4 Baselines

In order to get a better sense on the complexity of the problem, we tested three baseline classifiers: support vector machine (SVM), random forest (RF), and K-nearest neighbours (KNN). We used the implementations provided in the *scikit-learn* library for Python [17]. More specifically, for SVM we used a non-linear RBF kernel and for RF we set the maximum depth of each tree to be 15. All of the baseline experiments used 50,000 generated samples with a train-test split of 60/40. Each sample was made by a combination of one to ten instruments and was four seconds in length. In this case, differently from the neural models, the features used are the MFCCs of the resulting combination; we chose to use MFCCs for this setting, instead of the Mel spectrogram, to have a lower number of features, that is more manageable by parametric classifiers. We found SVM to have the highest accuracy of the three classifiers across all experiments.

The complete problem of classification of the pairs instrument-pitch has 424 total classes. This problem is very hard for parametric classifiers, so we initially switched to the simpler problem of 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%. KNN performed significantly worse than SVM, so we did not attempt any further testing with it.

To better approximate our original problem of identifying instrument and pitch, however, we performed experiments in which two instruments were selected and for input data that contained samples from one of those two instruments, both the instrument and pitch *class* of the sample would be classified. The pitch class is the note name without the octave, e.g. C, D#, G, etc. 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 the specified instruments,

Instr. 1	Instr. 2	SVM Acc.	RF Acc.
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 1. Comparison of accuracies between SVM and RF. 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.

say Violin and Viola, which pitch classes were present. If another instrument was present in the input combination that was not Violin or Viola, the classifier would simply identify that an instrument that was not one of the two was present. The accuracies from this experiment are outlined in Table 1. Since RF performed worse than SVM in every experiment, we stopped testing with it and used SVM from that point on.

We then performed this same experiment with three instruments having their pitch class 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.Indeed, 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, with parametric classifiers, the full setting of the problem in which individual pitches are classified for all instruments.

In the rest of this section, we will show the classification results of the two neural models.

2.5 CNN with LSTM

The first deep model we trained as a classifier for musical instruments and pitches was a CNN with a LSTM unit.

Our architecture is made of 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

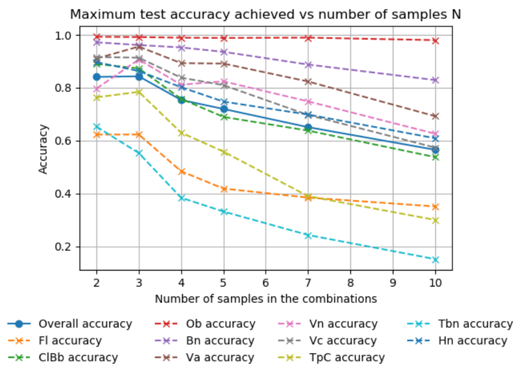

Figure 4. Best overall accuracy and for each instrument obtained by the CNN with LSTM depending on the number of instruments in the combinations.

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 2.

2.6 ResNet

The second and more deep model that we trained as classifier was the well known deep residual network *ResNet*. 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 decided to reduce the number of parameters and to reset the output channel numbers of each block with 32, 64, 32, 32 respectively.

2.7 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. However, the value of the loss function alone is difficult to interpret. For this reason we created a complementary function f to be used for evaluation only. This function compares a vector of expected outputs $\overline{X}$ with the estimated output from the model $\hat{X}$ by using the following function

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

where

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

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

Different experiments were made by varying the number N of samples in each mixture. We used an orchestra of 10 instruments containing Horn, Oboe, Violin, Viola, Cello, Flute, Trombone, Bassoon, Trumpet in C and Clarinet in Bb.

Then, for both CNN with LSTM and ResNet, we computed the maximum accuracy over the epochs. The CNN was trained on 200,000 generated samples over 50 epochs, and ResNet was trained on 400,000 generated samples over 20 epochs. Fig. 3 shows the best overall test accuracy achieved by both models across the number of samples N.

ResNet outperforms the CNN regardless of the number of samples used in the combination. This result is consistent with previous research [11], as residual networks usually perform well in classification problems.

Fig. 4 and Fig. 5 show the maximum accuracy computed using the function f from Eqn. (1). For ResNet, the variance in accuracy is much smaller until N reaches 5, at which point it becomes similar to the CNN. The results on both figures show consistency on the relative accuracy of instruments, which was for us the first step towards the validation of this method. Flute, Trombone and Trumpet yield the worst results for both models.

While it is not easy to explain these differences in accuracy, we hypothesize this being related to the nature of peculiar spectral and temporal morphology of each instrument. For example, flute notes tend to exhibit a steep spectral rolloff, with most of the energy captured by the first few partials. Moreover, the noisy nature of the transient portions of these note is not well represented by frequency-based descriptions such as Mel spectra. These two factors combined, could make the disentanglement of the flute from the analyzed combination more difficult.

Strings give similar results across both models. An interesting point to notice is the very high accuracy of Oboe on both models. This could indicate that there is an optimal spectral shape that maximizes the probability of being detected in such classification framework.

3. ORCHESTRATION EXPERIMENTS

After training the neural models for classification, we finally tested them for the task of target-based computer-assisted orchestration. For this purpose, we decided to use the models trained on combinations of 10 instruments, namely French Horn, Oboe, Violin, Viola, Cello, Flute, Trombone, Bassoon, Trumpet, and Clarinet. Each class, in this model, represents an instrument-pitch pair.

To orchestrate, a target sound is input to the model, and the 10 classes with the highest probability are extracted. These 10 classes are the instrument-pitch pairs that are most represented in the target, and can be from any combination of the 10 instruments.

Since we decided not to train our models to classify the dynamics of a sample (despite TinySOL having *pianissimo*, *mezzoforte*, and *fortissimo* recordings), the dynamics are determined by the probability of each sample as output

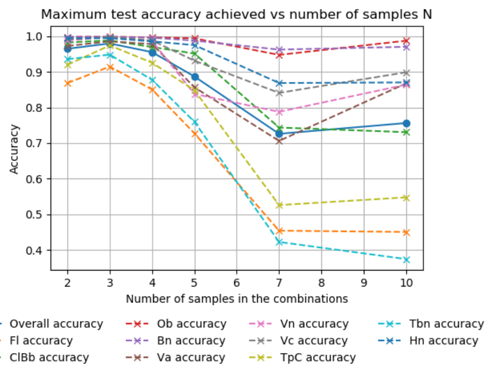

Figure 5. Best overall accuracy and for each instrument obtained by the ResNet depending on the number of instruments in the combinations.

by the model. If the model outputs a probability higher than 0.66 for a sample, the fortissimo version of the sample is used. If the probability is between 0.33 and 0.66, then the mezzoforte version is selected and if it is less than 0.33 the pianissimo version is used. The idea behind this quantization is that samples that are the most represented in the target should appear as the loudest in the orchestrated solution.

In order to test our models for orchestration, we used 15 targets from the Orchidea distribution. These targets represent a variety of signal types but are mostly static, in the sense that they do not change sensibly over time.

Some of the targets were made of instrumental samples: a sum of oboe and bassoon notes, single bassoon and bass clarinet notes, two multiphonics of bassoon and two symphonic chords. The other targets there are: the recordings of two bells, of a car horn, of a gong, of a screaming voice, of a wind harp, of the attack of a brass instrument and the recording of a boat docking.

Although the data we trained the model on was fixed to four seconds of audio, the target samples can be of arbitrary length since we adapt the representation to the size of the training data by changing the hop length during the computation of the Mel spectrogram.

3.1 Evaluation

We evaluated our orchestrations both qualitatively and quantitatively by comparing our solutions to the solutions generated by Orchidea, the state-of-the-art system for computer-assisted orchestration.

Orchidea implements many advanced features that are not supported by our models. For example, it is able to apply symbolic constraints to the search, hence allowing only specific instrumental combinations or playing styles. It is also able to reduce the search space by applying harmonic analysis on the target sound. 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

Model	Ob + Bn	Bn	Bass cl.	Bell 1	Bell 2	Multiph. 1	Car horn	Boat . . .
CNN with LSTM	0.17	0.28	0.70	0.55	0.26	**1.10**	0.68	**1.12** . . .
ResNet	0.34	0.50	0.48	0.59	0.45	0.90	0.49	**1.16** . . .

. . .	Wind harp	Chord 1	Multiph. 2	Chord 2	Gong	Scream	Brass	Average
. . .	0.55	0.79	0.70	0.57	0.73	**1.14**	0.79	0.71
. . .	0.61	0.86	0.51	0.37	0.71	**1.03**	**1.05**	0.66

Table 2. Quantitative comparison of orchestrations as ratios to Orchidea. Eqn. (3) was used with $\lambda_1 = 0.5$ and $\lambda_2 = 10$ to compute distances between orchestrations and targets. What is shown is the ratio between the distance calculated and Orchidea's distance for the same target. A value less than 1 means that the model performed worse (i.e. had a larger distance), and a value greater than 1 means the model performed better than Orchidea. The last column shows the ratio of the average distances for the model across all targets.

whose harmonics are much closer to the target, which is an important part of aural similarity. Orchidea's solutions, moreover, can use any number of instruments included in the orchestra specified by the user, thus having variable-sized orchestrations from a single instrument to the whole set (this property is called *sparsity* of the solution).

In order to have a more fair comparison, we did not allow Orchidea to use any of the advanced features: we did not apply any symbolic constraints or harmonic analysis and we forced it forced to use all 10 instruments in each solution. This creates a more fair comparison, since our models are unable to create constrained or sparse solutions.

Qualitative evaluation 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 loosely represented a C minor chord, so we checked if the orchestration contained the notes of the chord. If a target had specific notes, we identified if the note or its partials were present. For example there is target of an Oboe playing an A4 and a Bassoon playing a C#3, the solution from ResNet contained the partials of the Bassoon's note: C#3, G#4, C#5, and E6 were all included on various instruments in the solution.

For quantitative evaluation, we used the distance metric defined in Eqn. (3) to calculate differences in timbre between targets and solutions. This metric is proposed in [8] as part of the cost function used in Orchidea during the optimization. The equation takes in the full FFT of the target x and full FFT of the solution $\tilde{x}$. Then for each bin k of the FFT, it calculates the absolute difference between the values. The differing values of λ_1 and λ_2 allow the metric to penalize the solution in different ways.

In the first summation, λ_1 is multiplied by all the distances calculated when there was more energy in the target than the solution, since $\delta_{k1} = 1$ only when $x_k \geq \tilde{x}_k$. Similarly in the second summation, λ_2 is multiplied by all the distances when the solution provided more energy to a frequency than the target. Therefore, the relation between λ_1 and λ_2 determines whether a solution is penalized more for undershooting or overshooting the target.

We calculated 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 Table 3.

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

where $\delta_{k1} = \begin{cases} 1, \text{if } x_k \geq \tilde{x}_k \\ 0, \text{otherwise} \end{cases}$, $\delta_{k2} = \begin{cases} 1, \text{if } x_k < \tilde{x}_k \\ 0, \text{otherwise} \end{cases}$.

While our model is not able to outperform Orchidea, it shows consistent results. 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 harmonic content of the target.

4. CONCLUSIONS

Target-based computer-assisted orchestration through deep learning models seems a promising path, thanks to the ability of deep networks to classify individual instruments and pitches out of dense combinations of samples. This work, however, represents only a preliminary study of the potential of these methods for the task of assisted orchestration.

The first natural extension would be to support sparsity in our models. Our current models, indeed, orchestrate all targets using a constant number of instruments and it is not able to drop specific instruments from the solution. This does not take into account the density of different targets. Sparse solutions, in which the model decides how many samples should be used to best represent the target, would allow a small number of samples to be used for sonically sparse sounds and many to be used for sonically dense sounds. This would generate more meaningful orchestrations that would compare more favourably to the state of the art.

Another important extension would be to create a more powerful embedding spaces for the target and combinations. In [21] the authors propose to use LSTM-based models to predict the embedding features for the combinations used during the optimisation process in assisted orchestration. We believe that by combining their prediction model with our classification models could generate more faithful representations and improve the overall quality of generated orchestrations.

5. REFERENCES

[1] Jose David Fernández and Francisco Vico. AI Methods in Algorithmic Composition: A Comprehensive Survey. Journal of Artificial Intelligence Research, vol.48, pp. 513–582, 2013.

[2] Christopher Ariza, Navigating the landscape of computer aided algorithmic composition systems: A definition, seven descriptors, and a lexicon of systems and research. In Proceedings of the International Computer Music Conference, 2005.

[3] Yan Maresz, On Computer-Assisted Orchestration. Contemporary Music Review, vol. 32, n. 1, pg. 99-109, Routledge, 2013.

[4] Grégoire Carpentier, Damien Tardieu, Jonathan Harvey, Gérard Assayag and Emmanuel Saint-James. Predicting Timbre Features of Instrument Sound Combinations: Application to Automatic Orchestration, Journal of New Music Research, vol. 39, n. 1, pg. 47-61, Routlegde, 2010.

[5] Emmanouil Benetos, Margarita Kotti, and Constantine Kotropoulos. Large scale musical instrument identification. Sound and Music Computing Conference, Greece, 2007.

[6] Siddharth Bhardwaj. Audio data augmentation with respect to musical instrument recognition. Master's thesis, 2017.

[7] Wenhao Bian, Jie Wang, Bojin Zhuang, Shaojun Wang Jiankui Yang, and Jing Xiao. Audio-based music classification with densenet and data augmentation. arXiv preprint, https://arxiv.org/abs/1906.11620v1, 2019.

[8] Carmine-Emanuele Cella. Orchidea: a comprehensive framework for target-based assisted orchestration. In preparation; available on request from the author, 2020.

[9] Carmine-Emanuele Cella, Daniele Ghisi, Vincent Lostanlen, Fabién Levy, Yan Maresz and Joshua Fineberg. OrchideaSOL: a dataset of extended instrumental techniques for computer-aided orchestration. ICMC (under review), 2020.

[10] Nicolas Chetry, Mike Davies, and Mark Sandler. Musical instrument identification using LSF and K-means. Audio Engineering Society Convention 118, 2005.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv preprint, https://arxiv.org/abs/1512.03385, 2015.

[12] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[14] Tetsuro Kitahara, Masataka Goto, and Hiroshi G. Okuno. Pitch-dependent identification of musical instrument sounds. Applied Intelligence 23, 267–275, Springer, 2005.

[15] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for musical instrument recognition. ISMIR, 2016.

[16] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. Proceedings of the 14th Python in Science Conference, 2015.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, Mar 2017.

[19] Martin F. McKinney and Jeroen Breebaart. Features for Audio and Music Classification. ISMIR, 2003.

[20] Carmine-Emanuele Cella and Philippe Esling, Open-source modular toolbox for computer-aided orchestration, Proc. of the Timbre conference, Montreal, Canada, 2018.

[21] Jon Gillick, Carmine-Emanuele Cella and David Bamman. Estimating Unobserved Audio Features for Target-Based Orchestration, ISMIR, 2019.