Neural Models for Target-Based Computer-Assisted Musical Orchestration: A Preliminary Study

Carmine Emanuele Cella[1], Luke Dzwonczyk[1], Alejandro Saldarriaga-Fuertes[1], Hongfu Liu[1], and Hélène-Camille Crayencour[2]

[1] CNMAT - University of California, Berkeley
[2] Centrale Supélec, L2S, Univ. Paris-Saclay, CNRS
carmine.cella@berkeley.edu, dz.luke@berkeley.edu

Abstract. In this paper we will perform 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. This paper is an extended version of the paper "A Study on Neural Models for Target-Based Computer-Assisted Musical Orchestration" published in the proceedings of *The 2020 Joint Conference on AI Music Creativity.*

Keywords: computer-assisted orchestration, CNN, LSTM, ResNet, Orchidea

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)* (Fernandez & Vico, 2013; Ariza, 2005). Within CAC, target-based computer-assisted orchestration is a compelling case of how machine learning can be used for enhancing and assisting music creativity (Maresz, 2013).

Target-based computer-assisted orchestration takes a target sound as an input and attempts to find instrumental samples that best match the target given a specific similarity metric and a set of constraints. The target can be any arbitrary sound file, it does not have to be pitched, rhythmic, or "musical" in any way. 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 (Carpentier, Tardieu, Harvey, Assayag, & Saint-James, 2010) consists in finding an orchestration for any given sound by searching combinations of sounds from a database with a multi-objective optimization heuristic 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 (Cella & Esling, 2018; Cella, 2020) 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.

In this paper, we try a different approach to this problem by framing computer-assisted orchestration as a classification task. The main idea is to train a model to classify combinations of real instruments and then use this model for orchestration. To solve this classification problem, we turn to deep neural networks, which have shown success among a variety of classification tasks. Where Orchidea has multiple pre- and post-processing steps, we create an end-to-end network that takes a target sound as input and outputs a list of samples that can be combined to create the orchestrated solution.

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 (Benetos, Kotti, & Kotropoulos, 2007; Kitahara, Goto, & Okuno, 2005) and the more recent use of deep neural networks for musical classification (Lostanlen & Cella, 2016; Bian et al., 2019).

We perform preliminary experiments with two 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 has yielded positive results for image classification (He, Zhang, Ren, & Sun, 2015). We choose to use a CNN because of its success in audio classification (Hershey et al., 2016) and include an LSTM unit in it because of its ability to learn long term dependencies in data (Hochreiter & Schmidhuber, 1997), which is important given the temporal nature of audio.

After training these models on varying combinations of 10 instruments, we perform orchestration by feeding target sounds to the networks and synthesizing a solution from the samples output from the system. We then compare our orchestrations to Orchidea's solutions for the same targets, through both qualitative and quantitative means. For quantitative comparison, we compute a specific distance metric on the amplitude spectrums of the target and solution, comparing the distance between Orchidea's solution to the target and our model's solution to the target.

The 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 can be found at `https://github.com/dzluke/DeepOrchestration` and you can listen to audio examples at `https://dzluke.github.io/DeepOrchestration/`.

2 Neural models

2.1 From Classification to Orchestration

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-note pairs 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 order to have a baseline to compare our classification results and to get a sense of the complexity of the problem, we used various parametric classifiers along with the neural models. As described in section 2.4, three baseline parametric classifiers have been trained on simplified versions of our problem. They have been trained to either identify only the instruments present, or the instruments and pitch classes present, two simplified versions of our problem.

Two neural classifiers have been trained on the problem of recognizing instrument-pitch pairs, which has a total of 424 classes (i.e. 424 unique instrument-pitch pairs). We want to point out, however, that this is still a simplified problem compared to assisted orchestration. A complete orchestration solution 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 section 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 employed a different strategy to determine dynamics. 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 but 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. Fig. 1 illustrates the described ideas.

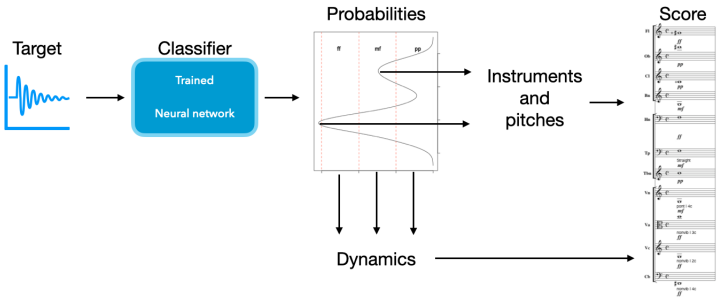


Fig. 1: An overview of the proposed method for assisted orchestration with neural models. Instruments and pitches are determined as peaks of the output probability distribution, while the dynamics are computed by quantizing the probabilities.

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 (Cella et al., 2020). 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`). The instruments and ranges over which they were recorded are summarized in Tab. 1.

For a given number of instruments N, each input to our model is a combination of N randomly selected TinySOL samples chosen from an orchestra of ten instruments. The only constraint imposed on the random selection is that there cannot be more than three samples from the same instrument in any one combination. We do this to ensure instrumental variety in the combinations. By randomly selecting the samples, we achieve a variety of instruments, pitches, and dynamics while avoiding introducing any bias in the combinations that could result from hand-picking the samples. The selected samples are combined to be played simultaneously and the magnitude is normalized by the number of instruments. The resulting combination has a sample rate of 44100Hz and is padded or trimmed to be exactly four seconds long.

We then computed the Mel spectrogram of the combination to be the features input to our models. We generated the Mel spectrogram using an FFT hop length of 2048 samples (the window of each FFT was 46ms long), 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 (McFee et al., 2015). The choice of the hop length is a compromise between the amount of information extracted by each FFT window, and the ability to capture temporal changes in the sound.

The Mel spectrogram is a representation of sound that mirrors the non-linear way in which humans perceive frequency, which makes it a perceptual representation of sound. Since we are attempting to recreate the timbre of a sound, and timbre is a perceptual descriptor, the Mel spectrogram is a fitting representation. It is common in music information retrieval (Mckinney & Breebaart, 2003) and has been successfully used in other tasks that employ neural networks for sound classification (Salamon & Bello, 2017).

Table 1: Table showing the pitch ranges present in the TinySOL database.

Instrument	**Abbreviation**	**Range**
Violin	Vn	G3-E7
Cello	Vc	C2-C6
Viola	Va	C3-C7
Trumpet in 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
Clarinet in B♭	ClBb	D3-G6
Contrabass	Cb	E1-C5
Bass Tuba	BTb	F#1-F4
Bassoon	Bn	A#1-D#5

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 (Salamon & Bello, 2017; Bhardwaj, 2017); more specifically, we used pitch shifting and partial feature dropout.

Pitch shifting was applied on the TinySOL samples each time they were 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.

Partial feature dropout 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 zero, which yielded an average of 1.28 columns and 3.45 rows being zero-ed out.

We chose to perform data augmentation solely for increasing the robustness of the models, not for any musical reasons. Data augmentation is only applied during training, so no pitch shifting or feature dropout occurs when orchestrating a target.

2.4 Baselines

In order to get a better sense of the complexity of the problem, we tested three baseline classifiers: support vector machine (SVM), random forest (RF), and K-nearest neighbors (KNN). We used the implementations provided in the *scikit-learn* library for Python (Pedregosa et al., 2011). 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. At the beginning of each experiment, a new set of combinations was created.

In this case, differently from the neural models, the features used are the Mel-frequency cepstral coefficients (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 which is more manageable for parametric classifiers. We found SVM to have the highest accuracy of the three classifiers across all experiments.

The complete problem of classification of the instrument-pitch pairs has 424 total classes. This problem is very difficult 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, 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. A selection of the results from this experiment are outlined in Tab. 2. 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. 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, and instead turned to neural models.

Table 2: Comparison of accuracies between SVM and RF. Each input to the models 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.

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%**

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, whose structure is inspired by the success in (Salamon & Bello, 2017). CNNs show good performance on classification problems for their ability to extract spatial features, and have shown success in audio classification (Hershey et al., 2016). LSTM units provide a way to learn long term dependencies in the data (Hochreiter & Schmidhuber, 1997), which is relevant given the sequential nature of audio.

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 two. The kernel size is 3×3 with a stride of 1 and a padding of 1. The number of filters are eight, 16, 32, and 32.

Following the first three convolutional layers, there is an LSTM layer which outputs 32 matrices. 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 a fully connected layer with Dropout, then 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 Fig. 2.

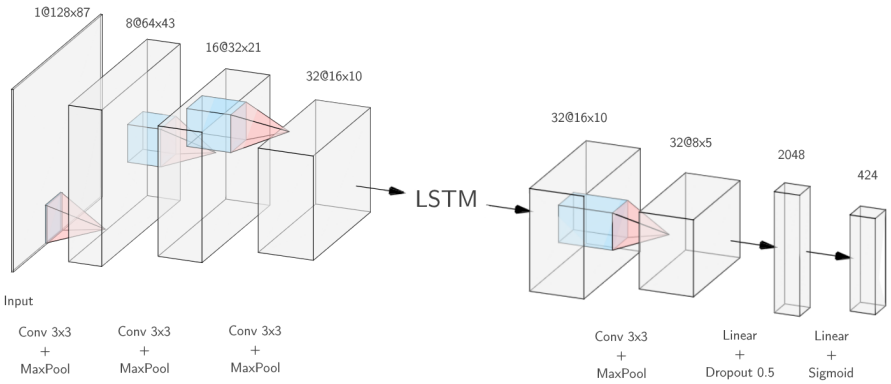


Fig. 2: Diagram of the CNN with LSTM architecture.

2.6 ResNet

The second and deeper model that we trained as classifier was the well known deep residual network *ResNet* (He et al., 2015). Specifically, we used 18-layer ResNet, which allows information to pass directly from the input to the final layer of each block. To make the model more suitable to our problem, we decided to use an architecture with 4 blocks whose outputs are of size 32, 64, 32 and 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 multi-label 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 one. 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

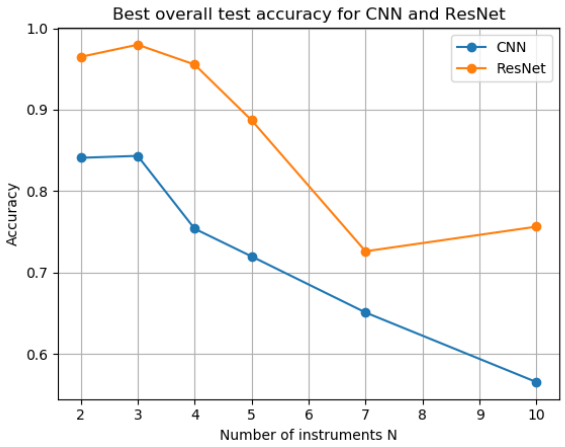


Fig. 3: Best overall accuracy for CNN with LSTM (50 epochs, 200k samples per epoch) and ResNet (20 epochs, 400k samples per epoch) depending on the number of instruments in the combinations used for training.

orchestration that matches the expected one. For this reason, the evaluation was done by comparing the proportion of estimated orchestration samples, chosen among the samples that output the highest probability, that matched the expected orchestration.

Different experiments were made by varying the number N of samples in each mixture. We used an orchestra of 10 instruments: French 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 accuracy on the test sets across epochs, and kept the best one as an indicator of the performance of the models.

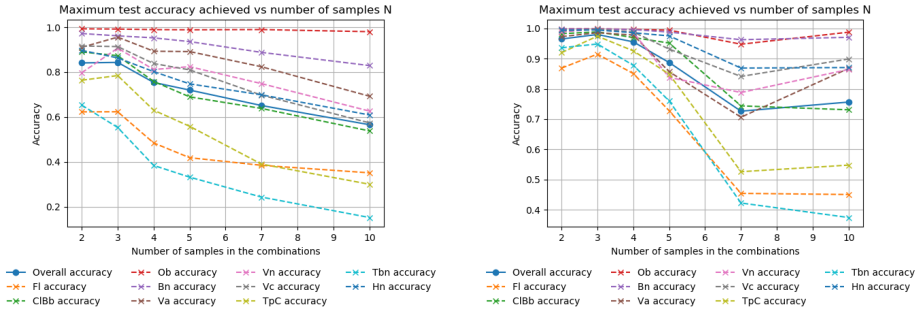
The CNN was trained on 200,000 generated samples over 50 epochs, and ResNet was trained on 400,000 generated samples over 20 epochs. ResNet, being a deeper model than the CNN, requires more data per epoch during training. Fig. 3 shows the best overall test accuracy achieved by both models across the 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 (He et al., 2015), as residual networks usually perform well in classification problems.

Fig. 4 shows the maximum test accuracy for each model. For ResNet, the variance in accuracy is much smaller until N reaches five, 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 notes 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.



(a) CNN with LSTM **(b)** ResNet

Fig. 4: Plot showing per instrument accuracies for CNN with LSTM and ResNet depending on the number of samples present in the combinations

3 Orchestration Experiments

After training the neural models for classification, we finally tested them for the task of target-based computer-assisted orchestration.

To orchestrate, a target sound is input to the model, and the ten classes with the highest probability are extracted. These ten classes are the instrument-pitch pairs that are most represented in the target, and can be from any combination of the ten 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 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, meaning they do not have significant harmonic change 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. Other targets were recordings that were not instrumental: bell sounds, car horn, gong, screaming voice, wind harp, the attack of a brass instrument, and the recording of a boat docking.

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

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

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 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 the *sparsity* of the solution).

In order to have a better comparison, we did not allow Orchidea to use any of the advanced features: we did not apply any symbolic constraints or harmonic analysis and forced it to use all ten 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 whether the orchestration contained the notes of this chord. If a target included specific notes, we identified whether the note or its partials were present. For example there is a target of an Oboe playing an A4 and a Bassoon playing a C#3. ResNet's solution for this target 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 (Cella, 2020) as part of the cost function used in Orchidea during the optimization. The equation takes in the full amplitude spectrum of the target x and full amplitude spectrum of the solution $\tilde{x}$. Then for each bin k of the amplitude spectrum, 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.

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

where $\delta_{k1} = 1$ if $x_k \geq \tilde{x}_k, 0$ otherwise; and $\delta_{k2} = 1$ if $x_k < \tilde{x}_k, 0$ otherwise.

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.

4 Conclusions

While our models are not able to outperform Orchidea, they show consistent results. During training, CNN and ResNet both perform well, with ResNet achieving higher training accuracies than CNN. For orchestration, CNN performs quantitatively better with an average ratio of 0.71 compared to ResNet with an average ratio of 0.66 (see Table 3). Both models outperform Orchidea on three out of fifteen of the targets. Through our qualitative inspection, we find that CNN seems to better emulate the timbre in its orchestrations, where ResNet is better for recreating the harmonic content of the target.

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 orchestrate all targets using a constant number of instruments and are 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 (Gillick, Cella, & Bamman, 2019) 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 we could generate more faithful representations and improve the overall quality of generated orchestrations.

Finally, the OrchideaSOL dataset could replace TinySOL as the dataset used. OrchideaSOL could improve the system by adding extended playing techniques to the data, which can allow noisier targets to be better orchestrated. OrchideaSOL contains 13,265 samples from 14 instruments, almost ten times as many samples as TinySOL (Cella et al., 2020). This would greatly increase the number of classes, and therefore the models would need to be trained on more samples and the number of layers in the models may need to increase.

References

Ariza, C. (2005). Navigating the landscape of computer aided algorithmic composition systems: a definition, seven descriptors, and a lexicon of systems and research. In *ICMC.*

Benetos, E., Kotti, M., & Kotropoulos, C. (2007). Large scale musical instrument identification.

Bhardwaj, S. (2017). *Audio data augmentation with respect to musical instrument recognition* (Master's thesis). doi: https://doi.org/10.5281/zenodo.1066137

Bian, W., Wang, J., Zhuang, B., Yang, J., Wang, S., & Xiao, J. (2019). *Audio-based music classification with densenet and data augmentation.*

Carpentier, G., Tardieu, D., Harvey, J., Assayag, G., & Saint-James, E. (2010, 03). Predicting timbre features of instrument sound combinations: Application to automatic orchestration. *Journal of New Music Research*, *39*. doi: 10.1080/09298210903581566

Cella, C.-E. (2020). Orchidea: a comprehensive framework for target-based assisted orchestration. *submitted to Journal of New Music Research, under review*.

Cella, C.-E., & Esling, P. (2018). Open-source modular toolbox for computer-aided orchestration. In *Timbre conference.*

Cella, C. E., Ghisi, D., Lostanlen, V., Lévy, F., Fineberg, J., & Maresz, Y. (2020). Orchideasol: a dataset of extended instrumental techniques for computer-aided orchestration. In *ICMC*.

Fernandez, J., & Vico, F. (2013, Nov). Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, *48*, 513–582. Retrieved from `http://dx.doi.org/10.1613/jair.3908` doi: 10.1613/jair.3908

Gillick, J., Cella, C.-E., & Bamman, D. (2019). Estimating unobserved audio features for target-based orchestration. In *Ismir*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. In *Ieee conference on computer vision and pattern recognition (CVPR)*.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., ... Wilson, K. (2016). *CNN architectures for large-scale audio classification.*

Hochreiter, S., & Schmidhuber, J. (1997, November). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. Retrieved from `https://doi.org/10.1162/neco.1997.9.8.1735` doi: 10.1162/neco.1997.9.8.1735

Kitahara, T., Goto, M., & Okuno, H. G. (2005). Pitch-dependent identification of musical instrument sounds.

Lostanlen, V., & Cella, C.-E. (2016). Deep convolutional networks on the pitch spiral for musical instrument recognition.

Maresz, Y. (2013, 02). On computer-assisted orchestration. *Contemporary Music Review*, *32*. doi: 10.1080/07494467.2013.774515

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Scipy*.

Mckinney, M., & Breebaart, J. (2003). Features for audio and music classification. In *Proceedings of the international symposium on music information retrieval* (pp. 151–158).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Salamon, J., & Bello, J. P. (2017, Mar). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, *24*(3), 279–283. Retrieved from `http://dx.doi.org/10.1109/LSP.2017.2657381` doi: 10.1109/lsp.2017.2657381