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

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, a.saldarriagafuertes@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.

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. 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.

The approach studied in (Carpentier, Tardieu, Harvey, Assayag, & Saint-James, 2010) 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 (Cella & Esling, 2018; Cella, 2020) and is implemented in the *Orchidea*

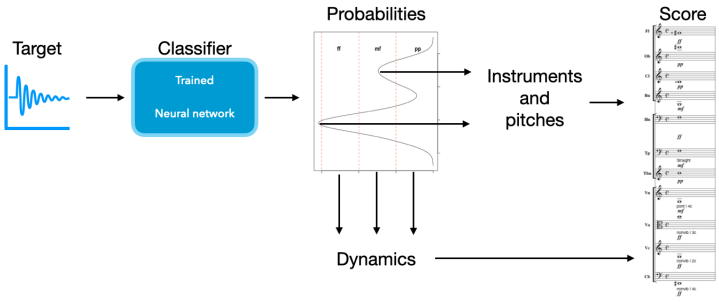


Fig. 1: An overview of the proposed method for assisted orchestration with neural models. Instruments and pitches are determined as peaks of the output probability distribution, while the dynamics are computed by quantizing the probabilities.

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 experimenting with deep neural architectures. The main idea is to train a model to classify combinations of real instruments and then use it 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 (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).

In this paper we perform preliminary experiments with two deep architectures: a convolutional neural network (CNN) with a long short-term memory (LSTM) unit and *ResNet*, a well known residual architecture that already yielded good results for image classification (He, Zhang, Ren, & Sun, 2015). We chose to use a CNN because of its success in audio classification (Hershey et al., 2016) and we decided to 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. The codebase for this paper 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 this reasons, our models were not trained to determine *instrument-pitch-dynamics* triples but instead *instrument-pitch* pairs.

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`).

For a given number of instruments N, each input to our model is a combination of N TinySOL samples chosen among an orchestra of 10 instruments. The data is generated by selecting N random TinySOL samples, 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. Then the chosen samples are combined to be played simultaneously and normalized by the number of instruments. The resulting combination has a sample rate of 44100Hz and is padded or trimmed to be exactly 4 seconds long. The Mel spectrogram of the mixture is computed using an FFT hop length of 2048 samples

(the window of each FFT is 46ms wide) and 128 Mel bins. Therefore, the Mel features fed to the model are matrices of size 128×345.

The choice of using the Mel spectrogram as input features for classification models is common in music information retrieval (Mckinney & Breebaart, 2003) and can be considered to be an appropriate representation of sound and musical signals. Such setup proved to be successful in (Salamon & Bello, 2017).

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. 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 (Pedregosa et al., 2011). In this case, differently from the neural models, the features used are the MFCCs of the resulting combination; as the number of features is more manageable for parametric classifiers. We found SVM to have the highest accuracy of the three classifiers across all experiments.

We started with a simplification of our problem. We performed experiments in which two instruments were selected, and the classifier attempted to identify instrument and pitch *class* of the samples. For two instruments, SVM had an accuracy of 39.9% and RF had an accuracy of 17.5%. As the number of instruments used in combination increased, the accuracy dropped sharply. With three instruments, SVM accuracy was 11.1%, with four instruments it was 2.7%. Clearly, these classifiers were not going to be able to achieve meaningful results with the full setting of our problem.

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). The LSTM unit was added in order to 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 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 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 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.

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, 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 maximum accuracy over the epochs. 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. 3a and Fig. 3b show the maximum test accuracy for each model. 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

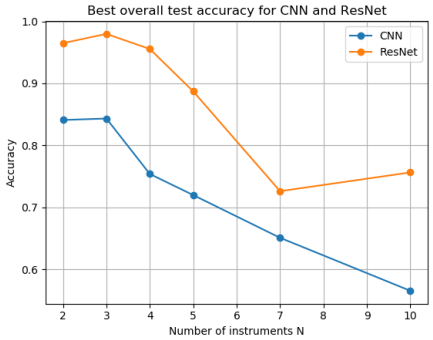


Fig. 2: Best overall accuracy for CNN with LSTM (50 epochs, 200k samples per epoch) and ResNet (20 epochs, 400k samples per epoch) depending on the number of instruments in the combinations used for training.

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.

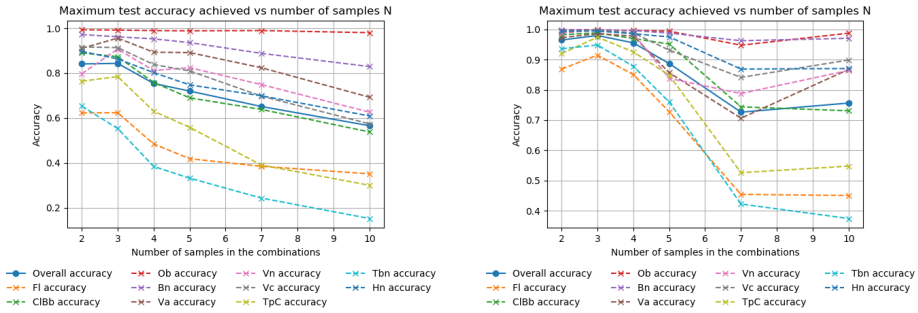
Strings give similar results across both models. An interesting point to notice is the very high accuracy of Oboe on both models. This could indicate that there is an optimal spectral shape that maximizes the probability of being detected in such classification framework.

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 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, 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. A probability between 0.33 and 0.66, is mezzoforte and less than 0.33 is pianissimo. The idea behind this quantization



(a) CNN with LSTM (b) ResNet

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

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 and chords, others are bells or gongs, and some do not feature any musical instruments

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 1: Quantitative comparison of orchestrations as ratios to Orchidea. Eqn. 1 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. In order to have a fairer comparison, we did not allow Orchidea to use any of its advanced features: we did not apply any symbolic constraints or harmonic analysis and we forced it to use all 10 instruments in each solution.

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 quantitative evaluation, we used the distance metric defined in Eqn. 1 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 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.

$$d(x, \tilde{x}) = \lambda_1 \sum_k \delta_{k1}(x_k - \tilde{x}_k) + \lambda_2 \sum_k \delta_{k2}|x_k - \tilde{x}_k| \,. \tag{1}$$

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.

A comparison of distances between our solutions and targets and Orchidea's solutions and targets is in Table 1. 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. You can listen to the targets and orchestrated solutions from Orchidea, ResNet, and the CNN with LSTM at `https://dzluke.github.io/DeepOrchestration/`.

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 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.

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.

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. *In preparation; available on request from the author*.

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

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.*

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.

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.

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