1. Project results and conclusions

1.1. Results

A summary of the first CNN implemented for root note estimation is shown below. The number of parameters per layer depends on the hyperparameters configuration. The final values are an agreement between computation time and network performance. The number of epochs and batch size allow the network to learn and improve the performance until it becomes steady around a fixed value of accuracy and loss parameter.

baseline_model architecture:

An overview of the final network proposed is shown below, where **X** must be replaced by 18 (note network) or 15 (chord network). The number of parameters in each layer are not written in order to generalize the model to any configuration of hyperparameters. The MaxPooling layer is deleted in some of the experiments.

Layer (type)	Output Shape
input_part (InputLayer)	(None, 1, 4656, 24)
convolution2d_1 (Convolution2D)	(None, 16, 4656, 24)
convolution2d_2 (Convolution2D)	(None, 32, 4656, 24)
permute_1 (Permute)	(None, 4656, 32, 24)
maxpooling2d_1 (MaxPooling2D)	(None, 2328, 16, 24)
reshape_1 (Reshape)	(None, 2328, 384)
timedistributed_1 (TimeDistribute	(None, 2328, **X**)

With the aim of getting a better idea of what our network is getting right and what types of errors it is making, a confusion matrix is calculated in the testing phase. The confusion matrix summarizes the performance of our classifier (our prediction task can also be seen as a classification system). We can see below an example of this matrix for the root notes of a test file. The numbers in the diagonal indicate right predictions. Calculating the trace of the matrix and dividing by the total number of notes, the accuracy for this file is obtained.

Experiments run:

					notes			chords		
Max pooling	# epochs	batch size	kernel shape	Total training time (s)	Train acc	Valid acc	Test acc	Train acc	Valid acc	Test acc
Yes	20	16	2x2	1626.2	0.609	0.596	0.617	0.420	0.411	0.421
Yes	20	16	3x3	1239.7	0.620	0.604	0.624	0.430	0.415	0.432
No	20	16	2x2	1563.4	0.602	0.586	0.608	0.421	0.411	0.421
No	20	16	3x2	1358.7	0.616	0.601	0.616	0.419	0.410	0.418
No	20	64	3x2	1153.3	0.605	0.591	0.610	0.425	0.412	0.420
No	20	64	3x3	1398.7	0.606	0.591	0.610	0.419	0.411	0.420
Yes	20	64	3x3	1515.4	0.613	0.599	0.617	0.522	0.517	0.499

Note: the experiments have been run in a GPU that allows faster implementation. Compared to the execution in a CPU, the training time is five times faster.

complex_model architecture:

Trying deeper networks out, the configuration shown in the next captions lead to better results.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_part (InputLayer)      (None, 1, 4656, 24)       0
_________________________________________________________________
convolution2d_1 (Convolution2D) (None, 16, 4656, 24)   80
_________________________________________________________________
maxpooling2d_1 (MaxPooling2D) (None, 8, 2328, 24)      0
_________________________________________________________________
permute_1 (Permute)          (None, 2328, 8, 24)       0
_________________________________________________________________
reshape_1 (Reshape)          (None, 2328, 192)         0
_________________________________________________________________
convolution1d_1 (Convolution1D) (None, 2328, 96)       36960
_________________________________________________________________
convolution1d_2 (Convolution1D) (None, 2328, 48)       9264
_________________________________________________________________
convolution1d_3 (Convolution1D) (None, 2328, 32)       3104
_________________________________________________________________
timedistributed_1 (TimeDistribute(None, 2328, 18)      594
=================================================================
Total params: 50002
```

Figure 1. Network for notes information

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_part (InputLayer)      (None, 1, 4656, 24)       0
_________________________________________________________________
convolution2d_2 (Convolution2D) (None, 16, 4656, 24)   80
_________________________________________________________________
maxpooling2d_2 (MaxPooling2D) (None, 8, 2328, 24)      0
_________________________________________________________________
permute_2 (Permute)          (None, 2328, 8, 24)       0
_________________________________________________________________
reshape_2 (Reshape)          (None, 2328, 192)         0
_________________________________________________________________
convolution1d_4 (Convolution1D) (None, 2328, 96)       36960
_________________________________________________________________
convolution1d_5 (Convolution1D) (None, 2328, 48)       9264
_________________________________________________________________
convolution1d_6 (Convolution1D) (None, 2328, 32)       3104
_________________________________________________________________
timedistributed_2 (TimeDistribute(None, 2328, 15)      495
=================================================================
Total params: 49903
```

Figure 2. Network for chord information

					notes			chords		
Max pooling	# epochs	batch size	kernel shape	Total training time (s) *	Train acc	Valid acc	Test acc	Train acc	Valid acc	Test acc
Yes	20	64	2x2	5245.2	0.660	0.653	0.661	0.522	0.518	0.501

* Training time without GPU enabled.

Output format:

A plain text file is generated with the chord prediction. The first number is the start time of the chord, followed by the end time and the full chord label. Here is a brief example for one of the test files:

```
0.0         0.139319727     N
0.139319727     3.622312925     E:maj
3.622312925     3.761632653     A:maj
3.761632653     5.201269841     E:maj
```