Chord detection with Deep Learning

Fatma Moalla

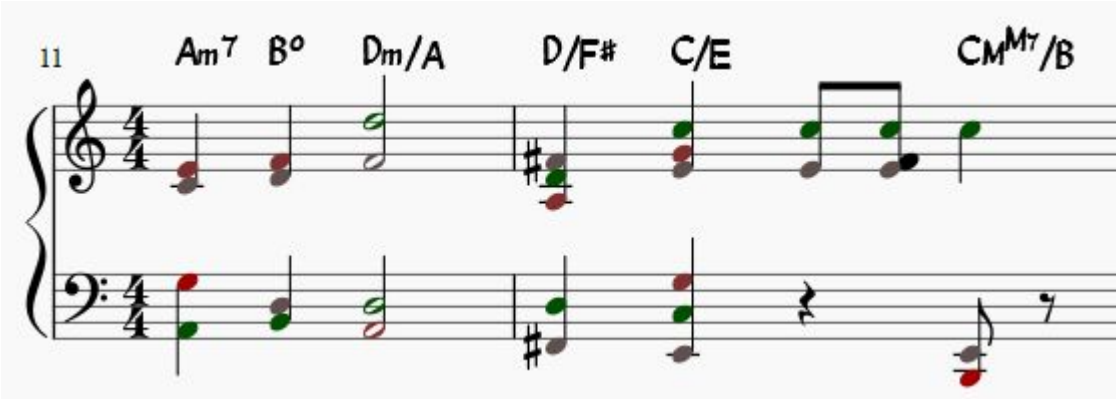
MVA - 2019/2020

26 - 03 - 2020

Motivation

- Music Information Retrieval : music transcription, musical analysis,structural segmentation.

  → Automatic Chord Recognition: challenging



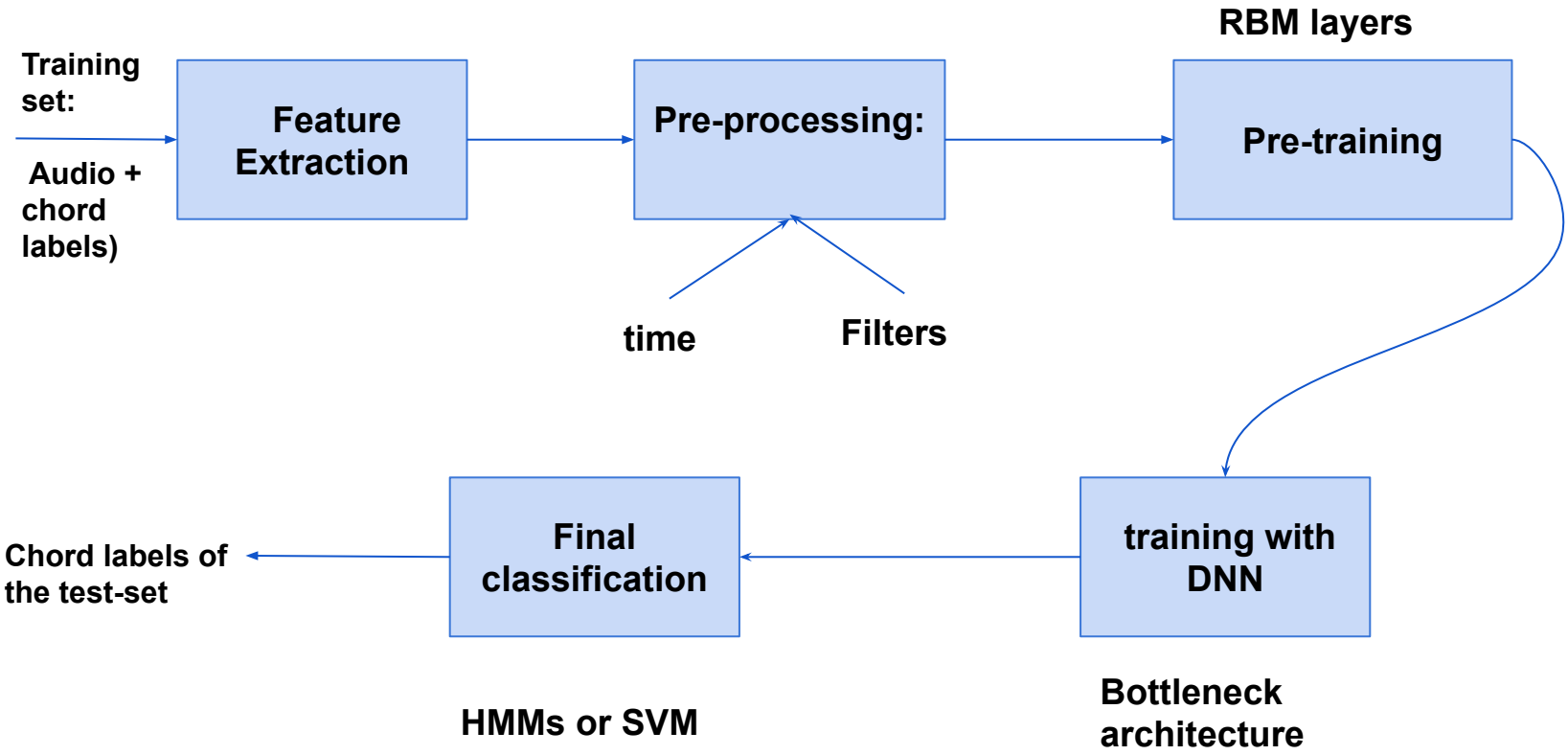
Paper Analysis

Paper Presentation

- Standard approach in previous papers : chroma features + HMMs

- New approach presented by Zhou and Lerch, 2017 :
- Use 24 + 1 pitch labels instead of 12 pitches
- Using Deep Belief Network to learn high-level features : bottleneck and common structures
- Use SVM or HMMs as final classifier

Overall structure of the solution



Feature extraction

- Use Constant Q Transform : inspired from Short-Time Fourier Transform (STFT)

→ Advantage : Center frequencies = musical notes and they are not linearly-spaced

- Center frequencies : $f_k = f_0 2^{\frac{k}{\alpha}}$

- Spectrogram : $X[n] = \frac{1}{N[n]} \sum_{i=0}^{N[n]-1} w[i-n]x[i]\exp(\frac{-j2\pi Qi}{N[n]})$

- Quality factor : $Q = (2^{\frac{1}{\alpha}} - 1)^{-1}$

Pre-Processing

- **Time splicing :** “superframes”= previous+current +following frames

- **Filter splicing**

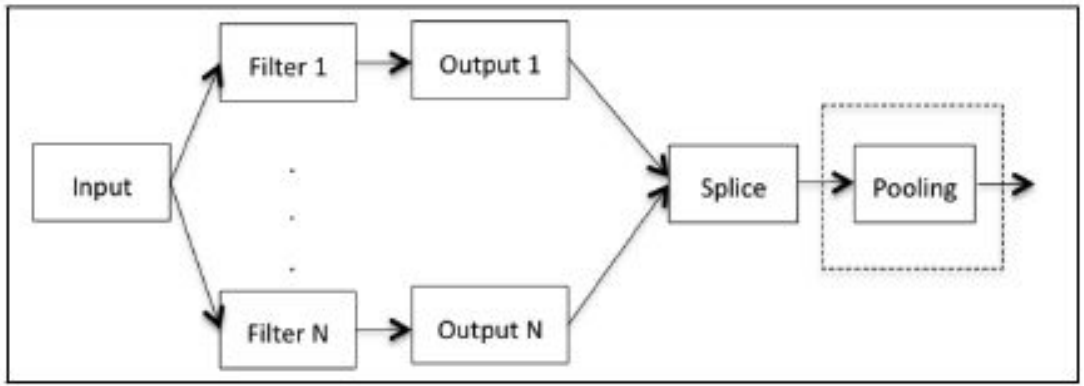


Fig. 2: Filters configurations to perform splicing

Filters in Pre-Processing step

1) **Single Low pass filter:**

$$y_n = (1-\alpha)y_{n-1} + \alpha x_n$$

2) **Two FIR filters + exp. decay impulse**

$$y_1(n) = \sum_{k=1}^{N} a^{-k+1} x(n-N+k)$$

$$y_2(n) = \sum_{k=1}^{N} a^{-k+1} x(n-N-k)$$

Deep Belief Network Learning

- **Gibbs sampling + Restricted Boltzman Machine layers** : pre-training phase

- **DNN** with two approaches : common and **bottleneck architecture**

→ **Objective:** learn more high level features

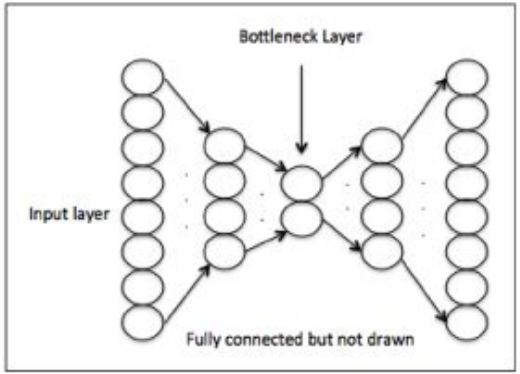
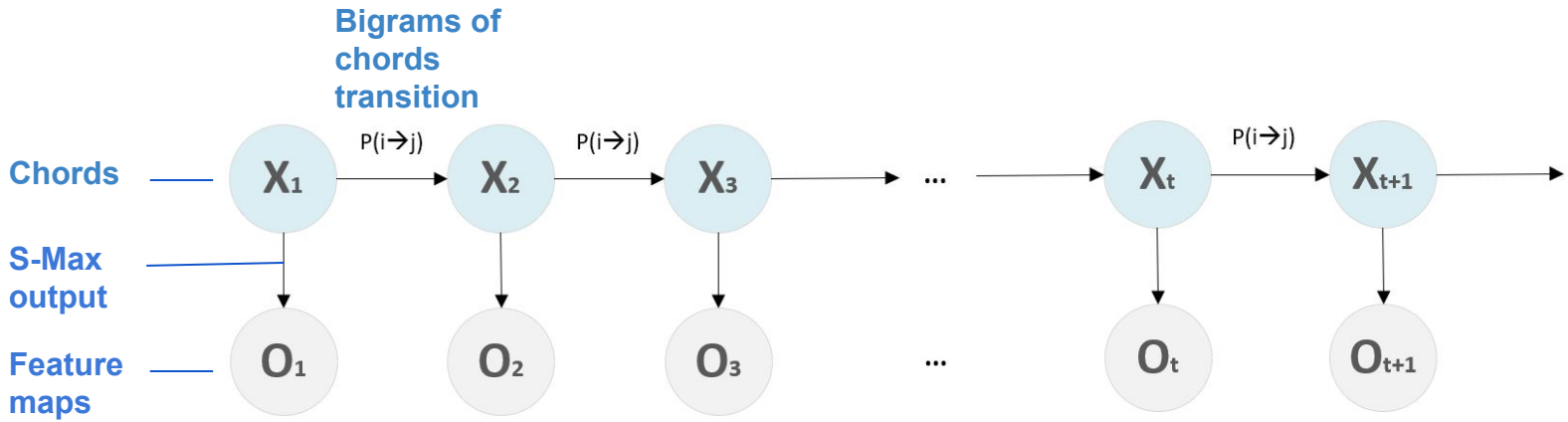


Fig. 3: Bottleneck architecture

Post-Processing

- **SVM :** use the output of the softmax layer + frame-by-frame labelling
- **HMMs :** not the backbone of the solution but a final classifier



Comparison with recent papers

Evaluation

- **Recognition rate:** $WCSR = \frac{1}{N}\sum_{k=1}^{n} C_k$

  → **achieved 0.919**

- **Other papers:**

- **RNN structure** (Korzeniowski and Widmer, 2018) : 0.821 with richer dataset (songs, instrumental records ..)
- **HMMs structure** (Lee and Slaney, 2006) : 0.8

Experiments

Dataset and Set-Up

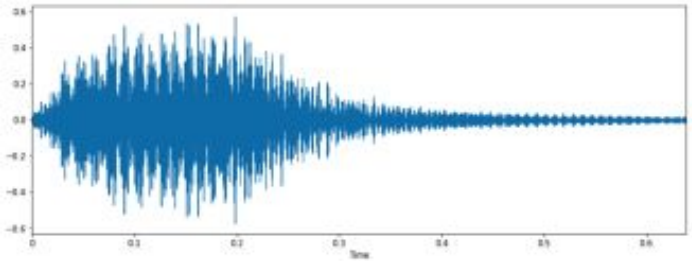
- Violin dataset with 100 recordings

- 10 chord type (A,Am,Bm,C,D,Dm,E,Em,F,G)

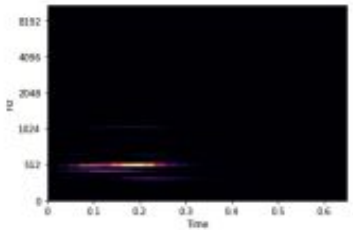
→ **Objective :** automatic chord detection with a customized network (keras)

Processing step

- **Data augmentation :** speed and pitch shifting (-1 semitone)
- **Pre-processing :** Mel-filter spectrogram



(a) Waveplot of Aminor



(b) Spectrogram of Aminor

training and results

- Convolutional and dense layers (+dropout).
- Evaluation metric : **f1-score** (instead of WCSR) → **0.986 !**
- Misclassification within A and C chord labels.

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 124, 68, 24)	624
max_pooling2d_3 (MaxPooling2	(None, 31, 34, 24)	0
activation_6 (Activation)	(None, 31, 34, 24)	0
conv2d_5 (Conv2D)	(None, 27, 30, 48)	28848
max_pooling2d_4 (MaxPooling2	(None, 6, 15, 48)	0
activation_7 (Activation)	(None, 6, 15, 48)	0
flatten_2 (Flatten)	(None, 4320)	0
dropout_3 (Dropout)	(None, 4320)	0
dense_3 (Dense)	(None, 64)	276544
activation_8 (Activation)	(None, 64)	0
dropout_4 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 10)	650
activation_9 (Ac	Chord_detection_violin_example - Jupyter Notebook	0

Thank you for your attention