FAST CHIRPLET TRANSFORM ENHANCES CNN-BASED AUDIO CLASSIFIER ON SMALL DATA

Hervé Glotin
DYNI, LSIS, Machine Learning & Bioacoustics team
AMU, University of Toulon, ENSAM, CNRS, IUF
La Garde, France
glotin@univ-tln.fr

Julien Ricard
DYNI, LSIS, Machine Learning & Bioacoustics team
AMU, University of Toulon, ENSAM, CNRS
La Garde, France
julien.ricard@gmail.com

Randall Balestriero
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA
randallbalestriero@gmail.com

ABSTRACT

Advanced soundscape analysis or machine listening are requiring efficient time frequency decompositions. The recent scattering theory is offering a robust hierarchical convolutional decomposition, nevertheless its kernels need to be fixed. Convolutional Neural Net (CNN) can be seen as the optimal kernel decomposition, nevertheless it requires large amount of training data. This paper shows that Chirplet kernels are providing good constant Q time-frequency representation which yields to better CNN classification than usual acoustic representation. The contributions are (1) to motivate Chirplet bioinspired auditory representation. (2) We define the first algorithm of Fast Chirplet Transform (FCT). (3) We validate FCT on environmental recordings of orca and birds. (4) We show that FCT improves CNN classification of complex overlapping bird calls on small data per species LifeClef Challenge. (5) We demonstrate on the same benchmark that pretraining on FCT representation of the low level layers of the CNN accelerates its convergence by 28%, while it yields to 8% relative gain of Mean Average Precision.

1 INTRODUCTION

Representation of bioacoustic sequences started with 'Human' speech in the 70'. Speech has been represented and processed during 50 years, yielding to standard and efficient Mel Filter Cepstral Coefficients (MFCC). Today new paradigms come from environmental monitoring and species classification at weak Signal to Noise Ratio (SNR) and small data per species. One difficulty in learning optimal filters by CNN is the lack of large number of high SNR samples. Moreover, theoretical acoustic invariants across environmental or species sounds are yet set.

Several neurobiological evidences suggest that auditory cortex is tuned to complex time varying acoustic features and consists of several fields that decompose sounds in parallel (Kowalski et al., 1996; Mercado et al., 2000). Therefore it is more than reasonable to investigate the Chirplet time-frequency representation from acoustic and neurophysiological point of views that we resume in the first section, and we argue that low level CNN layers shall be pretrained by a parametric chirplet like kernel.

Thus, we define a Fast Chirplet Transform (FCT) that we validate on real recordings of whale and birds. We then demonstrate that CNN classification benefits from low level layers FCT pretraining to balance small data. We conclude on the perspectives opened by possible integration of FCT representation for machine listening.

2 CHIRPLET

Chirps, or transient AM-FM waveforms, are ubiquitous in nature systems as presented in Flandrin (2001). Audio signals Chirps are naturally encountered in many audio signals, ranging from bird songs and music, to animal vocalization (frogs, whales) and speech. Actually the sinusoidal models are a typical attempt to representing audio signals as a superposition of chirp-like components.

Chirp signals are also commonly observed in natural sonar systems. Most species of bats make use of an ultrasound system based on chirps whose parameters can be shown to directly control echolocation.

2.1 NEUROPHYSIOLOGICAL MOTIVATIONS FOR CHIRPLET REPRESENTATION

Cortical neurons represent sounds efficiently (Mercado et al., 2000) and electrophysiological studies have demonstrated that ring patterns in specific neural regions can often be predictably correlated with particular sound features (Kowalski et al., 1996), even if the underlying neural codes that give rise to such correlations remain unclear. However the main properties of auditory cortex are as recalled in (Mercado et al., 2000): complex patterns of sound feature selectivity, species-specific signal decomposition, Dynamic modulation of response characteristics.

The Chirplet transform appears to be well suited for our purposes: it subsumes both Fourier analysis and wavelet analysis, providing a broad framework for mapping one-dimensional sound waveforms into an n-dimensional auditory parameter space. It offers the processing that have been described in previous section for different auditory fields, i.e. cortical regions with systematically related response sensitivities. Moreover, Chirplet spaces are highly over-complete because there is an infinite number of ways to segment a time-frequency plane, the dictionary is redundant: this corresponds well with the overlapping, parallel signal processing pathways of auditory cortex.

2.2 FORMAL DEFINITION OF CHIRPLET

A chirplet can be seen as a complex sinus with increasing or decreasing frequency over time modulated by a Gaussian window to have a localized support in the time and Fourier domain. It is a broad class of filters which includes wavelets and Fourier basis as special cases. As a result, and as presented in (Mann & Haykin, 1991; 1992), the chirplet transform is a generalization of many known time-frequency representations. We first present briefly the wavelet transform framework to extend it to chirplets. Given an input signal x one can compute a wavelet transform (Mallat, 1999) through the application of multiple wavelets ψ_λ. A wavelet is an atom with compact support in time and frequency domain which integrates to 0. The whole filter bank is derived from a mother wavelet ψ_0 and a set of dilation coefficients following a geometric progression defined as $\Lambda = \{2^{1+j/Q}, j = 0, ..., JQ-1\}$ with J being the number of octave to decompose and Q the number of wavelets per octave. As a result, one can create the filter-bank as the collection $\{\psi_0(\frac{t}{\lambda}) := \psi_\lambda, \lambda \in \Lambda\}$. After application of the filter-bank, one ends up with a time-scale representation, or scalogram, $Ux(\lambda, t) := |(x \star \psi_\lambda)(t)|$ where the complex modulus was applied in order to remove the phase information and contract the space. It is clear that a wavelet filter-bank is completely characterized by its mother wavelet and the set of scale parameters. Generalizing this framework for chirplets will be straightforward by now allowing a nonconstant frequency for each filter. As for wavelets, filters are generated from a Gaussian window determining the time support however the complex sinus has nonconstant frequency over time with center-frequency f_c. Since the scope of the parameters leads infinitely many different possible filters, we have to restrain ourselves and thus create only a fixed chirplet filter-bank allowing fast computations.The parameters defining these filters include the time position t_c, the frequency center f_c, the duration Δ_t and the chirp rate c:

$$g_{t_c,f_c,\log(\Delta t),c}(t) = \frac{1}{\sqrt{\sqrt{\pi}\Delta t}} e^{-\frac{1}{2}\frac{(t-t_c)^2}{\Delta_t^2}} e^{j2\pi(c(t-t_c)^2+f_c(t-t_c))}. \tag{1}$$

We now present the strategy we adopted to select a priori the parameters used for our chirplet filters.

3 Proposition of an algorithm for Fast Chirplet Transform (FCT)

The parameter space is basically of infinite dimension. Similarly to continuous wavelet transform however, it is possible to use some a priori knowledge in order to create a finite bank-filter. For example, wavelets are generated by knowing the number of wavelets per octave and the number of octave to decompose. As a result, we used the same motivation in order to reduce the number of possible chirplets required. The goal here is not to compute an invertible transform but rather provide a redundant transformation highlighting transient structures which are not the same tasks as discussed in (Coifman et al., 1992; Meyer, 1993; Coifman et al., 1994). As a result, we keep the same overall framework as for wavelets with the Q and J parameters. Standard parameters to analyze bird songs are $J = 6$ and $Q = 16$ with a sampling rate of 44100Hz. In addition, the chirp rate was proportional and the duration was included. Finally, since we are interested in frequency modulations, we compute the ascendant and descendant chirp filters as one being the symmetrized version of the other. As a result, we use a more straightforward analytical formula defined with a starting frequency F_0, an ending frequency F_1, and the usual wavelet like parameters σ being the bandwidth. Finally the hyperparameter p defining the polynomial order of the chirp is constant for the whole bank-filter generation. The case $p = 1$ leads to a linear chirp, $p = 2$ to a quadratic chirp. The starting and ending frequencies are chosen to approximately cover one octave and are directly computed from the λ parameters which define the scales. Finally, following the scattering network inspiration from (Bruna & Mallat, 2013), in order to remove unstable noisy pattern, we apply a low-pass filter namely a Gaussian blurring and thus increase the SNR.

$$\Lambda = \{2.0^{1+i/Q}, i = 0, ..., J \times Q - 1\} \tag{2}$$

$$F_0 = \frac{Fs}{2\lambda}, \lambda \in \Lambda \tag{3}$$

$$F_1 = \frac{Fs}{\lambda}, \lambda \in \Lambda \tag{4}$$

$$\sigma = 2\frac{d}{\lambda}, \lambda \in \Lambda \tag{5}$$

3.1 A Nearly Real-Time FCT Implementation

We propose in this paper a Fast Chirplet Transform (FCT), taking advantage of the a priori knowledge for the filter-bank creation and the fast convolution algorithm discussed in 3.2. Therefore, we first create the chirplet with the ascendant and descendant versions in once with:

```
Algo 1: Chirplet Generation
INPUT: F0,F1,Fs,sigma,p
OUTPUT: coefficients_upward,coefficients_downward
if(p):
    w=cos(2*pi*((F1-F0)/((p+1)*sigma**p)*t**p+F0)*t)
else:
    w=cos(2*pi*((F0*(F1/F0)**(t/sigma)-F0)*sigma/log(F1/F0)))
coefficients_upward=w*exp(-((t-\sigma/2.0)**2)/(2*sigma**2))
coefficients_downward=flipud(coefficients_upward).
```

Then we generate the whole filter-bank using Algo 1 with the defined λ and hyper-parameters:

```
Algo 2: Chirplet Filter-Bank Generation
INPUT: J, Q, Fs, sigma, p
lambdas            = 2.0**(1+arrange(J*Q)/float(Q))
start_frequencies  = (Fs /lambdas)/2.0
end_frequencies    = Fs /lambdas
distances          = 2.0*d/flipud(lambdas)
filters=list()
for f0,f1,d in zip(start_frequencies,end_frequencies,distances):
    filters.append(chirplet(Fs,f0,f1,d,p))
return filters.
```

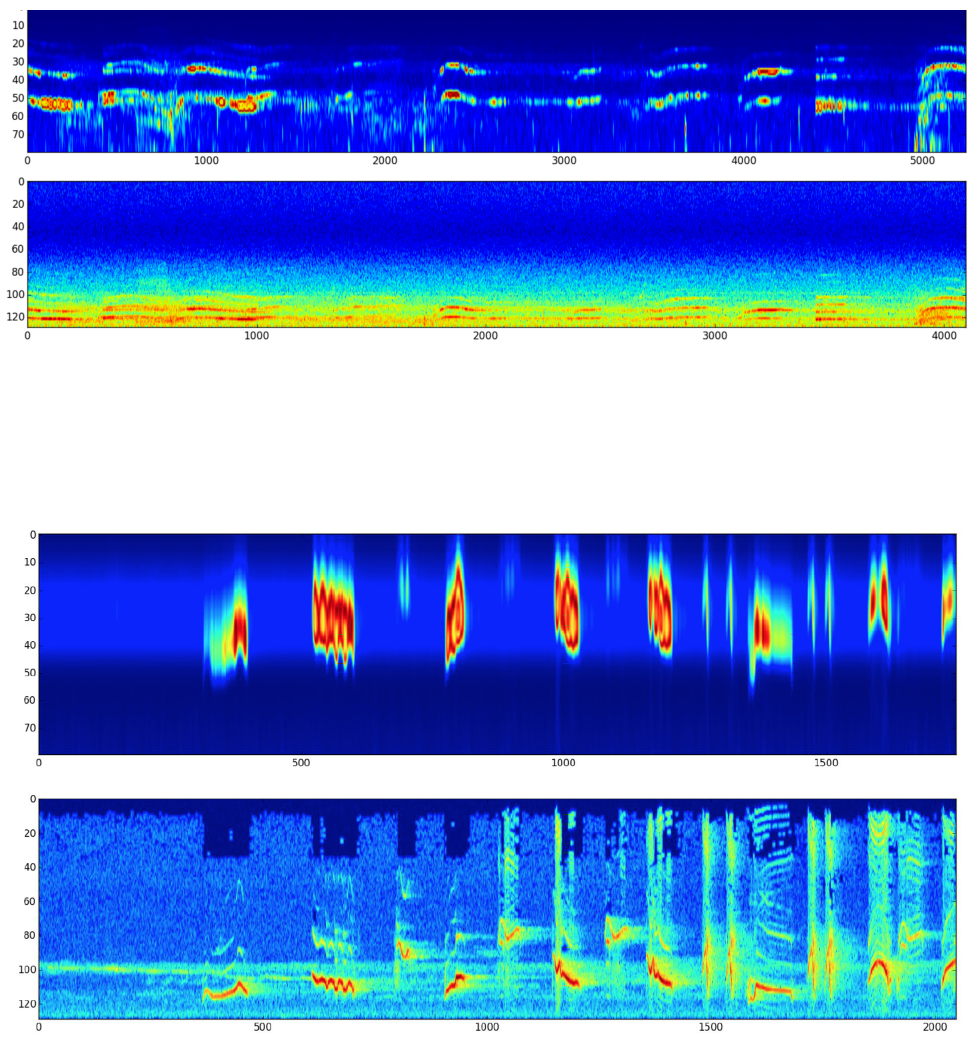

Figure 1: Top : Chirplet of orca call with p=3, j=4, q=16, t=0.001, s=0.01, with usual FFT spectrogram, showing the calls (in red) enhanced in the Chirplet compared to the FFT. Bottom the same for bird calls from Amazonia [8] (Sampling Rate = 16 kHz, 16 bits.). Wave and Chirplets of orca are online : `http://sabiod.univ-tln.fr/orcalab`

Finally, we use the scattering framework (Mallat et al. (Bruna & Mallat, 2013; Andén & Mallat, 2014)). The scattering coefficients Sx are resulting from a time-averaging on the time-frequency representation Ux bringing local time-invariance. This time-averaging is computed through the application of a scaling function ϕ, usually a Gabor function with specified standard deviation. As a result, one computes these coefficients as: $Sx(\lambda, t) = (|x \star \psi_\lambda| \star \phi)(t)$ where ψ_λ is a chirplet with λ parameters and ϕ. Similarly, we perform local time-averaging on the chirplet representation in the same manner.

We present some possible filters in Fig. 2, and some the bird features Fig. 3. The main setting comes from the SNR compared to other types of transforms commonly used such as the FFT spectrogram.

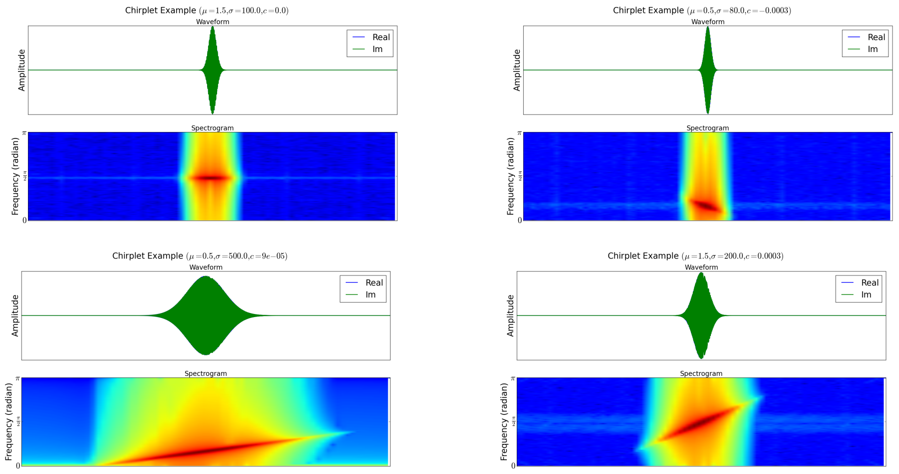

Figure 2: Some FCT displayed in the physical domain and in the time-frequency domain through a spectrogram. The first one reduces to a wavelet since the chirp rate is 0. One can see the importance of the time duration and the chirp rate and well as the center frequency depending on what one wishes to capture.

.

3.2 Optimal FFT windowing for FCT

The third step in our FCT consists in the reduction of the convolution task. The asymptotic complexity of the Chirplet transform is $O(N.\log(N))$ with N being the size of the input signal. This is the same asymptotic complexity as for the continuous wavelet transform and the scattering network. However, it is possible to reach lower asymptotic complexity simply by a division of the convolution task. usually the convolutions are carried through application of an element-wise multiplication of the signal and the filter in the frequency domain and then compute the inverse Fourier transform to end up with $x \star \psi_\lambda$. However, if we denote by M the length of the filter ψ_λ it is possible to instead perform multiple times this operation on different overlapping chunks of the signal to then concatenate the results to obtain at the end the same convolution result but now in $O(N.\log(M))$. Finally a last improvement induced by this approach is to allow easy tackling of signals with a length just above a power of 2 which otherwise would require to be padded in order to obtain a FFT with real $O(N.\log(N))$ complexity through the Danielson-Lanczos lemma (Press, 2007). Applying this scheme allowed to compute the convolutions between 3 to 4 times faster. The variations came from the distance between N and the closest next power of 2 depending on the desired chunk size.

3.3 FCT vs FFT on real bioacoustic scenes of orca and birds

We defined a simple procedure to optimize the appropriate basis functions of the chirplets according to the maximisation of the SNR gain of the time frequency targets in the Fourier domain vs FCT. We validated also the efficiency of FCT on real bioacoustic recordings. First we processed on 10 medium speed CPUs (4 year old) in 2 days the whole 999 bird species data set from LifeClef bird challenge (16 kHz Sampling Rate, 16 bits, nearly 100 hours of recordings). Second, we processed in 7 days the equivalent of 1 month of continuous recordings of orca whale from Orcalab.org ONG project (22 kHz SR, 16 bits). Chirplet samples are depicted Fig. 1,2,3 and online at `sabiod.univ-tln.fr/orcalab/`.

4 Enhancing CNN by FCT

A strategy for CNN fine-tuning can be to retrain a classifier on top of a CNN on a new dataset, but to also fine-tune the weights of a pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the CNN or to freeze some of the earlier, later or central layers fixed (due to overfitting concerns), and to only fine-tune some portion of the network. As the DNN

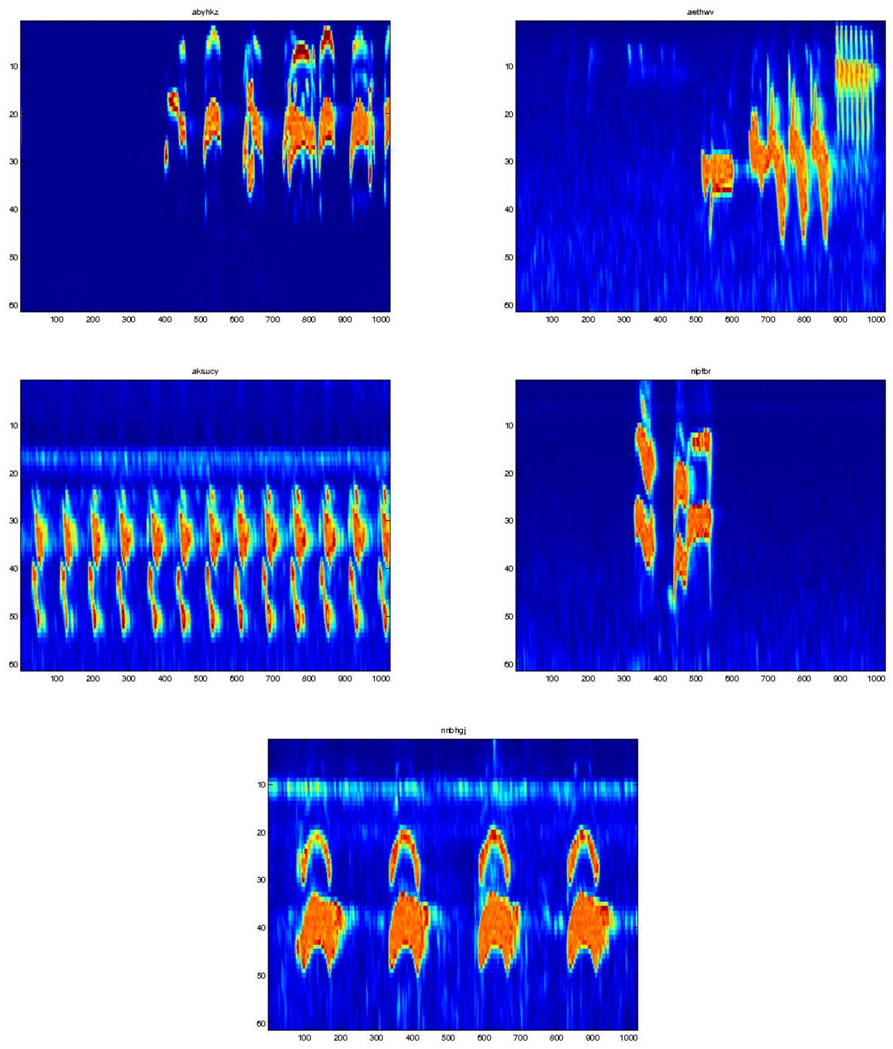

Figure 3: FCT of 5 species of amazonian birds LifeClef 2015 challenge including BIRD10 dataset). The calls are the high SNR red regions. The species are international codes, from top to bottom, right to left, species code name are : abyhkz, aethwv, aksucy, nipfbr, nnbhgj.

.

features propagate deeper and deeper in the network layers, they become increasingly invariant and discriminative (Seltzer et al 2013). Thus usually only the higher level are fine-tuned : the earlier features of a CNN contain more generic features that should be useful to many tasks. As denoted in later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset in ImageNet for example.

Here we adapt a parametric chirplet decomposition to a specific acoustic domain with a specific CNN. We first recall the chirplet definition and properties. Then we compare a CNN trained on raw audio to one trained on mel and chirplet. The best model is the one trained on parametric chirplet. Second we show that one can boost the CNN by pretraining Chirp in low level layer.

4.1 Comparison of FCT to raw audio and Mel at similar CNN

We train CNNs (LeCun & Bengio, 1995) on the Lasagne Theano platform, on a subset of LifeClef 2016 bird classification challenge that was extracted for ENS Ulm data challenge 2016, on the 3 species numbered [26,45,9], for a total of 15 minutes of recordings including testing and training

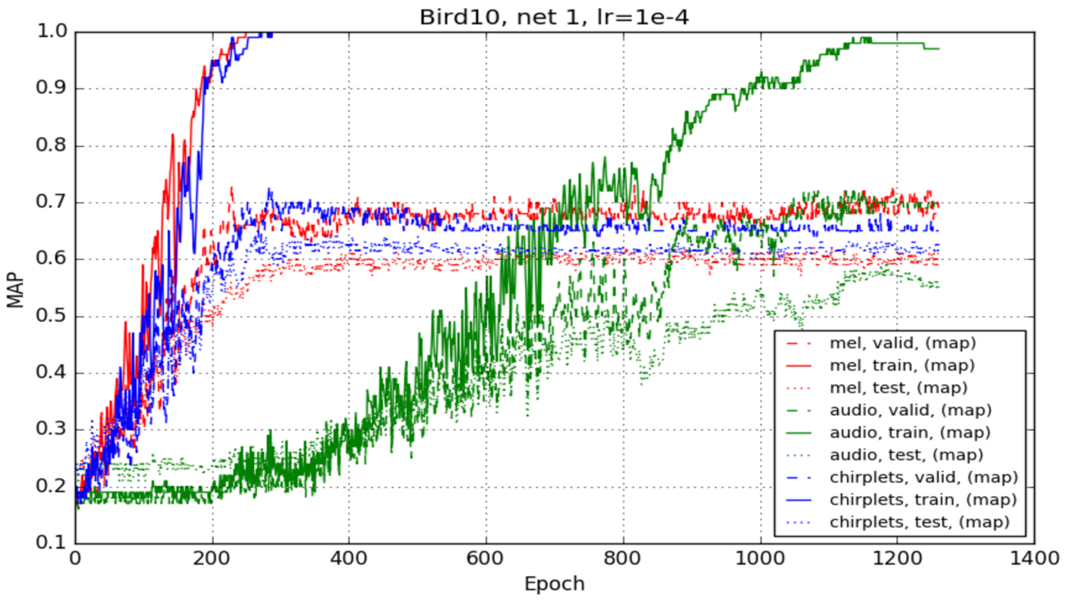

Figure 4: The accuracy of the CNN trained on the Mel versus Audio raw versus the Chirplets representations. The training is line, validation is dashed, the dot line is the test. The training conditions are the same on the three CNN. They all have similar size and topology. The CNN trained on chirplet is slightly better than trained on Mel or raw audio, and one interesting point is that it is is converging faster.

data sets. This data set is available with Chirplet features at `http://sabiod.univ-tln.fr/workspace/BIRD10/`.

We then train three different CNNs. A baseline CNN is trained from the raw audio. A second CNN, with similar topology (see annexe) is trained on a simple log of the simple 64 channel mel scale of FFT spectrum (from usual Columbia univ. tool http://pydoc.net/Python/librosa/0.2.0/librosa.feature/). We overlap by 90% the time windows. A third CNN is trained on our Chirplet representation as depicted above. The parameters of both CNN are similar, with 64 frequency bands each (we remove top and bottom band from the Chirplet to set to 64 bands only). Then the input layer is 64 x 86, the Conv layer of 20 filters of size 8 x 10. All activation functions are relu. We maxpool 2 x 2, follow the 20 filters of size 8 x 10, maxpooling, dense layer (200), dropout at 10%, with a final softmax dense layer with 3 classes and same dropout. Each CNN is trained by cross-entropy, L2 reg., with a learning rate set of 0.001.

The Fig. 4 gives the MAP of these two CNNs having similar number and values of hyperparameters. The CNN on Chirplet gives the best MAP with 61.5% at epoch 280 compared to later epoch (820) for Mel with a similar MAP of 61%. Audio is slower and weaker (58% MAP at epoch 1140).

4.2 Stacking pretrained Chirplet representation CNNs

In order to show the efficiency of the chirplets, we pretrain a CNN to encode audio to chirplets (a.k.a. Audio2Chirp CNN) and a CNN to convert parametric Chirplet to classes (a.k.a. Chirp2Class CNN). The topology of these CNNs are set not to reach optimal MAP, but in order to have reasonable time of training (Tab. 2 and 3). We also speed up the training with shorter time overlap of the time windows (only 30% instead of 90% in the previous experimentation). We then decrease the average MAP, however the objective here is to compare the gain in MAP and time of convergence by stacking and freezing Chirplet representations.

We simply stack the low level layer audio2chirp with the chirp2class CNN to build a complete audio2class CNN. We train this new CNN from random, or from pretrained CNN. Note that the seed for the random initialisation of the CNN is similar in all CNN to allow fair comparisons. We report

(Tab. 1) the results for each of the stacked CNN: the number of epochs giving the best MAP on dev set, and the corresponding MAP on the test set.

Model	**Epoch (dev)**	**MAP (%) (test)**
Baseline (Fig. 5): training Audio2Class CNN from random initialisation	530	51
Simple forward of stacked Audio2Chirp.Chirp2Class separately trained	0	31
Training Audio2Class CNN from stacked trained Audio2Chirp and trained Chirp2Class CNNs (Fig. 6)	390	53
Idem with frozen Chirp2Class : training Audio2class CNN from stacked trained Audio2Chirp and trained Chirp2Class CNNs without update of Chirp2Class CNN (Fig. 7)	380	55

Table 1: Summary of the CNN enhanced by the FCT representation, duration of convergence on dev set, and Mean Average Precision on test set.

These results demonstrate that the pretraining of low level layers of the CNN by Chirplet representation boosts its convergence from 530 epochs (baseline) to 380 epochs, while it increases its MAP by 4 points (nearly 9% of relative gain). More details are given in Annexe with the training curves of each of the CNN (respectively Fig. 5, 6, 7).

5 Discussion and Conclusion

We presented a novel implementation for a Fast Chirplet Transform (FCT). FCT can then be computed nearly online on large data acoustic set. In the first experimentation part of this paper, we showed that FCT offers much faster convergence at similar MAP than Mel or raw audio (280 epochs versus 820 or 1140). This can be interpreted by the sparsity of the chirplets as well as the direct filtering of noisy structures versus bird related features that should be coded by the CNN.

Third, we demonstrate that it is easy to train a Chirpnet, and that we can improve the convergence of a stacked CNN by pretraining of its low level layer by a audio2chirp CNN. We demonstrated then an acceleration of the convergence of the CNN from 530 epochs to 380 epochs only, while an increase of 4 points of MAP (Table 1).

A perspective of this seminal work would be to integrate Chirplet parametrization into the CNN training itself, as a constrained embedded layer. Then the CNN would benefit of the optimal SNR due to adapted Chirplet representation. This might be done by developing a framework similar to a Wavelet Neural Network (Adeli & Jiang, 2006) but with Chirplet activation functions.

Finally, these experiences bring to light the problem of deep learning for small, noisy and biased dataset for which a full learning strategy is sub-optimal due to local optimum convergence. As a result, prior knowledge can be used to mitigate this drawback by reducing the modelisation work of the deep-net architecture.

Future work will consist in sparse Chirpnet inspired from tonotopic net (Strom 1997). It would consist to force a tonotopic organization of the chirpnet inspired from the auditory nerve and auditory cortex Pironkov et al. (2015).This approach consists of limiting the neurons connections between the hidden layers. This preserves frequency proximity : the vibrations in the air are transmitted through the tympanic membrane to the base of the cochlea thus each region of the basilar membrane will be excited by different frequencies. The lower frequencies excite areas closer to the cochlea base, whereas higher frequencies are closer to the apex.This implies that neurons connected to a specific zone of the basilar membrane will be simultaneously stimulated (a.k.a. tonotopic connections).

6 ACKNOWLEDGEMENTS

We thank great colleagues for fruitful discussions these last years on CNN, Scattering and Time-frequency representations. Experimentations were written on Theano, Lasagne [1], Librosa [2] and Pysoundfile [3].

REFERENCES

Hojjat Adeli and Xiaomo Jiang. Dynamic fuzzy wavelet neural network model for structural system identification. *Journal of Structural Engineering*, 132(1):102–111, 2006.

Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Ronald R Coifman, Yves Meyer, and Victor Wickerhauser. Wavelet analysis and signal processing. In *In Wavelets and their Applications*. Citeseer, 1992.

Ronald R Coifman, Yves Meyer, Steven Quake, and M Victor Wickerhauser. Signal processing and compression with wavelet packets. In *Wavelets and their applications*, pp. 363–379. Springer, 1994.

Patrick Flandrin. Time frequency and chirps. *Proc. SPIE*, 4391:161–175, 2001. doi: 10.1117/12.421196. URL `http://dx.doi.org/10.1117/12.421196`.

Nina Kowalski, Didier A Depireux, and Shihab A Shamma. Analysis of dynamic spectra in ferret primary auditory cortex. ii. prediction of unit responses to arbitrary dynamic spectra. *Journal of Neurophysiology*, 76(5):3524–3534, 1996.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

Steve Mann and Simon Haykin. The chirplet transform: A generalization of gabor's logon transform. In *Vision Interface*, volume 91, pp. 205–212, 1991.

Steve Mann and Simon Haykin. Adaptive chirplet transform: an adaptive generalization of the wavelet transform. *Optical Engineering*, 31(6):1243–1256, 1992.

Eduardo Mercado, Catherine E Myers, and Mark A Gluck. Modeling auditory cortical processing as an adaptive chirplet transform. *Neurocomputing*, 32:913–919, 2000.

Yves Meyer. Wavelets-algorithms and applications. *Wavelets-Algorithms and applications Society for Industrial and Applied Mathematics Translation., 142 p.*, 1, 1993.

Gueorgui Pironkov, Stéphane Dupont, and Thierry Dutoit. Investigating sparse deep neural networks for speech recognition. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pp. 124–129, 2015. doi: 10.1109/ASRU.2015.7404784. URL `http://dx.doi.org/10.1109/ASRU.2015.7404784`.

William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[1]https://github.com/Lasagne/Lasagne
[2]https://github.com/librosa/librosa
[3]https://github.com/bastibe/PySoundFile

A BIRD10 Dataset preparation

BIRD10 is a online `http://sabiod.univ-tln.fr/workspace/BIRD10/` subset of the LIFEClef 2016 challenge on bird classification. This subset contains 454 audio files (22050 Hz, 16 bits) from 10 bird classes, split in 0.5s overlapping segments. Only segments with detected bird activity were kept, assuming a bird sound to have prominent energy and to be mostly harmonic. The algorithm for our bird detection is, for a given segment:

```
if (energy_ratio > energy_threshold and
    spectral_flatness_weighted_mean < spectral_flatness_threshold):
    has_activity = True
else:
    has_activity = False
```

where the energy and the spectral flatness are computed on 50% overlapping frames of 256 samples, and

$$er = energy\_ratio = \frac{mean(seg\_energy)}{95thpercentile(file\_energy)}$$

$$sfw = spectral\_flatness\_weighted\_mean = \frac{sum(seg\_spectral\_flatness \times seg\_energy)}{sum(seg\_energy)}$$

This naive algorithm proved to perform quite well on a manually labelled dataset of bird vocalizations, with a detection precision of 0.89 and a recall of 0.57 for er =0.2 and sfw = 0.3[4].

The training and test sets were defined in the LifeClef challenge. 20% of the training set was used as the validation set.

A.1 Comparing audio, log-amplitude mel spectrum and chirplets as inputs for CNN classification

The first experiment consisted in running similar CNNs to compare the performance of using raw audio and two time-frequency representations as the input: a standard log-amplitude mel spectrum and the chirplet representation described in the frist part of this paper. In this experiments the segments were overlapping by 90%. The topologies of the networks are given in Tab. 2. The cost function used is the cross-entropy, we used a learning rate of 1e-4 and applied a L2 regularisation coefficient of 1e-4. The results are shown in Fig. 4.

The mel spectrum is computed from 64 bands between 0 and 11025 Hz (SR / 2).

Both the log-amplitude mel spectrum and the chirplets were normalized to get mean=0 and std=1 on the training set.

B Stacking CNNs : audio2chirp with chirp2class

Here networks were trained with 30% overlap (as opposed to 90% in experiment 7.2) to get faster training. Since it generates less examples, the performance is lower but it converges faster.

In all experiments, a given topology is always initialized using the same set of random parameters, unless specified otherwise. The Symbol “*” refers to the optimal trained parameters of a net.

[4]A quick grid search on the two parameters was conducted, aiming at getting good precision and not too bad recall (we are interested in detecting segments that really contain bird activity, even though we miss some).

Input	Topology
Audio, shape (1, 11025)	conv_1: 20 filters of shape (1, 400) (nonlinearity: relu) pool_1: (1, 4) max pooling conv_2: 20 filters of shape (1, 100) (nonlinearity: relu) pool_2: (1, 4) max pooling dense_1: 400 units (nonlinearity: relu, 10% dropout) dense_2: 10 units (nonlinearity: softmax, 10% dropout)
Log-amplitude mel spectrum, shape (64, 80)	conv_1: 20 filters of shape (8, 20) (nonlinearity: relu) pool_1: (2, 2) max pooling conv_2: 20 filters of shape (8, 20) (nonlinearity: relu) pool_2: (2, 2) max pooling dense_1: 200 units (nonlinearity: relu, 10% dropout) dense_2: 10 units (nonlinearity: softmax, 10% dropout)
Chirplets (chirp2class), shape (80, 110)	conv_1: 20 filters of shape (8, 20) (nonlinearity: relu) pool_1: (2, 2) max pooling conv_2: 20 filters of shape (8, 20) (nonlinearity: relu) pool_2: (2, 2) max pooling dense_1: 200 units (nonlinearity: relu, 10% dropout) dense_2: 10 units (nonlinearity: softmax, 10% dropout)

Table 2: CNN topologies for the 3 different inputs

Input	Topology
Audio, shape (1, 11025)	conv_1: 40 filters of shape (1, 1001) (nonlinearity: relu) pool_1: (1, 4) max pooling conv_2: 40 filters of shape (1, 501) (nonlinearity: relu) pool_2: (1, 4) max pooling conv_3: 40 filters of shape (1, 101) (nonlinearity: relu) pool_3: (1, 4) max pooling dense_1: 8800 units (nonlinearity: relu, 10% dropout) reshape_1: 8800 -> (80, 110)

Table 3: CNN topology of the chirp encoder (*audio2chirp*)

B.1 AUDIO2CHIRP - CHIRPLET ENCODER

The chirp encoder, aka *audio2chirp*, aims at training a net to get a chirplet-like representation. It is a simple CNN taking audio as input, chirplets as output and minimizing the square error. It converges easely in 180 epochs. The topology of the audio2chirp net is given Tab. 2.

B.2 TRAINING CURVES OF THE PARTS OF THE STACKED BOOSTED CNN

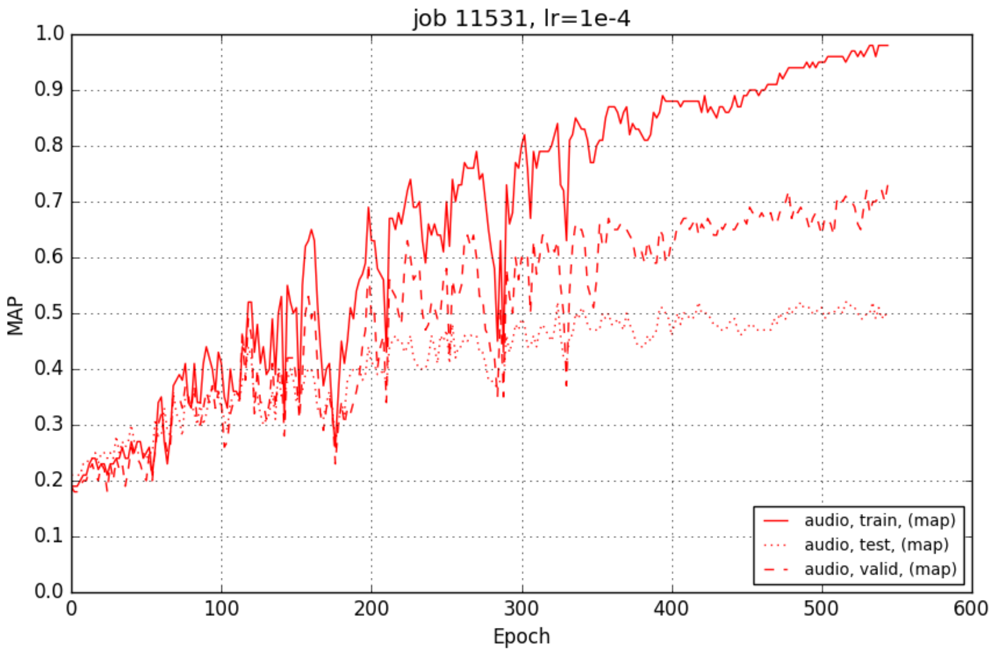

Figure 5: Trained stacked CNN from random : Initialized with the same random values as used to initialized audio2chirp* and chirp2class*.

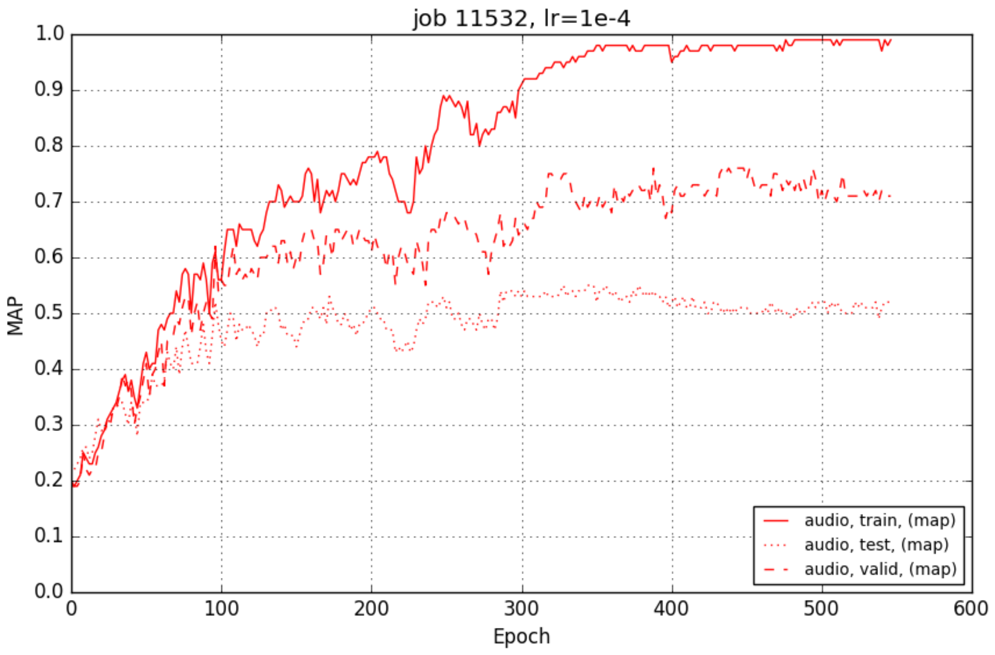

Figure 6: Trained stacked CNN from pretrained CNN : Initialized with optimal audio2chirp* and optimal chirp2class*.

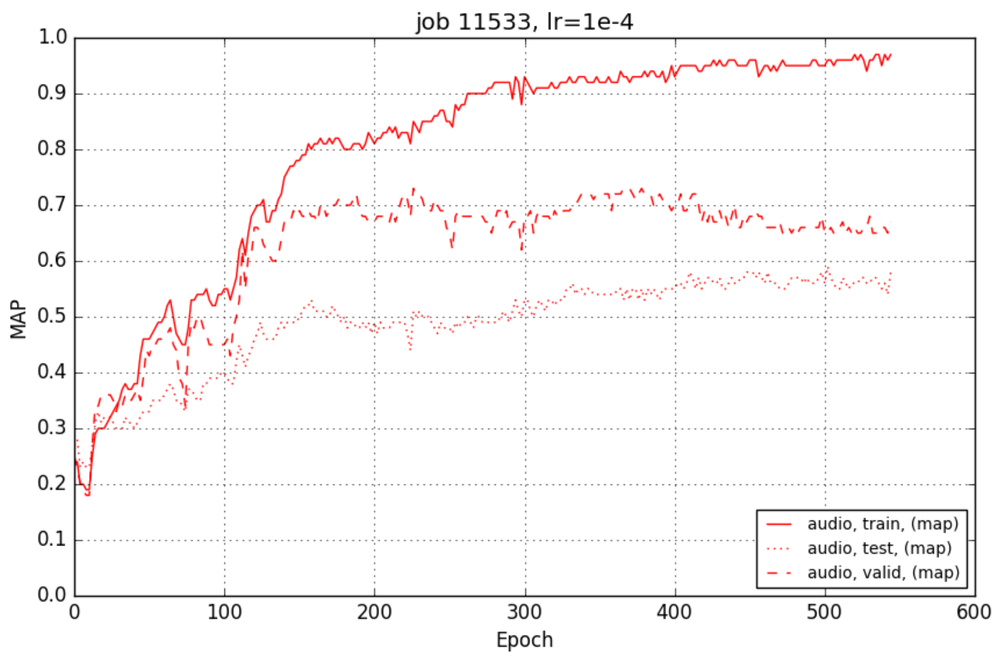

Figure 7: Trained stacked CNN from pretrained CNN but freezing Chirp2class : Initialized with optimal audio2chirp* and optimal chirp2class* and freezing chirp2class (no weight update).