DEEPFOURIER: CLASSIFYING SOUND WITH CONVOLUTIONAL NEURAL NETWORKS

Benjamin Jüttner, Lorenzo Belgrano, Péter Semság, Sébastien Demortain

DTU Compute, Technical University of Denmark

ABSTRACT

In this project we looked into sound classification using convolutional neural networks based on previous work done by Karol J. Piczak [1], who trained his network from Mel-spectrograms. Our aim was to improve upon the original work by making the classifier end-to-end through integration of the raw audio transformation in the deep learning network itself (which we call *DeepFourier*). We created 2 new networks, both performing as well as Piczak's original network. After reading this report, please open DEMO.ipynb (uploaded to DTUInside) to see a quick introduction to our code.
Our entire code can be found on
www.github.com/pr3d1c70r/DybLSPB

Index Terms— Convolutional Neural Network, sound classification, Mel-spectrogram, Short-Term Fourier Transformation

1. INTRODUCTION TO PREVIOUS WORK

1.1. Data

The dataset used in this project was the same as one of those used by Piczak, called UrbanSound8K [2]. This dataset consists of 8732 audio clips of length 4 seconds or less each. These clips all belong to 10 classes of "urban" sounds (see appendix A). The sampling rate of the dataset was (after resampling) 22.05 kHz and it comes pre-arranged into 10 folds for cross-validation. In the original work[1] (and the replication of it done by us) the transformation of the raw audio was done by first cutting the original audio clips into 50% overlapping windows (from now on called *segments*), each 20992 samples (≈ 1 sec.) long. On each segment a Mel-spectrogram was calculated , with $H = 60$ pixels in frequency direction (= number of Mel bins) and $U = 41$ pixels in temporal direction (= number of time bins). Every segment is normalized by its maximum amplitude before taking the spectrogram. The spectrograms are commonly used features as the mel scale approximates the human auditory system's response more closely[3]. In addition to the Mel-spectrogram, Piczak used delta's (which are the local estimates of the derivatives of the raw audio) as inputs.

1.2. Architecture: PiczakNet

Piczak considered 2 channels (Mel-spectrograms +delta's) as input images and applied for the classification: 2 convolutional layers with max-pooling, 2 fully connected layers and one softmax layer. This network (fig. 1) is from now on called PiczakNet.

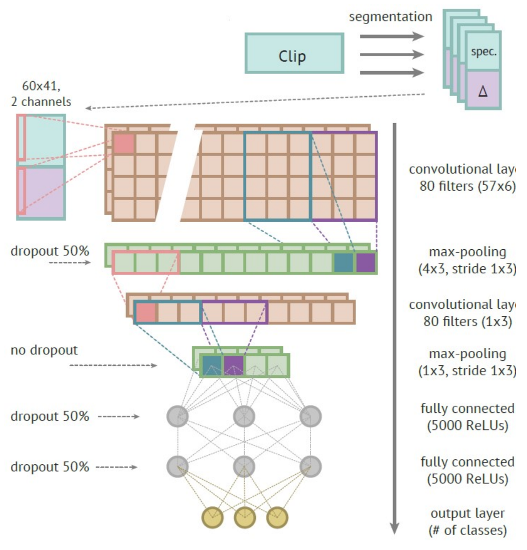

Fig. 1: The original architecture done by Piczak [1]. We reproduced it, except for the "Δ" in the picture.

1.3. Results

The original model by Piczak achieved an accuracy of about 70% on this dataset. Our replication of his work on the other hand achieved about 60% accuracy, but this was done without using the previously mentioned delta's since we wanted to focus our project on the spectrogram representation. Also, we sought to classify an individual *segment* correctly whereas Piczak used majority- and probability-voting schemes (classifying an entire audio clip based on all its segments) in his calculations. See fig. 2 for the confusion matrices of the original and the replicate network, showing a similar pattern.

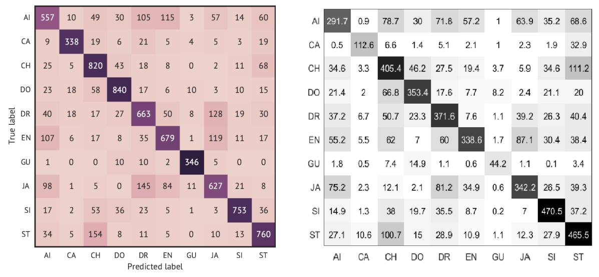

Fig. 2: Confusion matrices for PiczakNet. *Left:* Piczak's own results after cross-validation. *Right:* Our reproduction, averaged on the 10 folds. Scaling for each matrix is different but the pattern is similar.

2. EXTENSION OF THE NETWORK

2.1. Motivation

In our attempt to surpass the performance of the PiczakNet, we remind ourselves that the Mel-spectrogram of a waveform segment necessarily contains less information than the segment itself. Therefore, by training end-to-end (i.e. designing a network that takes raw waveform as an input) we will access the entire information from the signal and hopefully draw a better classification accuracy from it. The next intuition is to keep using PiczakNet since it served well so far. So we just place additional neural network layers between the raw waveform and PiczakNet – exactly where the Mel-transformation used to be – and aim to learn a spectral representation of the raw audio better than the Mel-spectrogram. We call these additional network layers *DeepFourier*. That leads us to the graph of our new, extended network:

$$\text{waveform} \rightarrow \text{DeepFourier} \rightarrow \text{PiczakNet} \rightarrow \text{label}$$

To orient our choice of DeepFourier architecture, our heuristic argument is that this "more effective representation" lies **in the neighbourhood of a Mel-transformation**. But what kind of neural network architecture can "hit" such a neighbourhood? To answer that, we need to mathematically explain what the Mel-transformation does. (We tried to keep our symbols intuitive, but when in doubt, please refer to our "notation appendix" B.) The Mel-transformation maps a segment $\mathbf{x}$ to a *Mel*-spectrogram M. The first step is an STFT:

$$S_{t,k} = |\sum_{l=1}^{L} x_{t,l} \cdot v(k,l)|^2, \quad t = 1..U, \quad k = 1..F \qquad (1)$$

$$x_{t,l} := \mathbf{x}_{(t-1)Z+l}, \qquad v(k,l) := \exp(-2\pi i(l-1)(k-1)/L)$$

$x_{t,l}$ is an element of $\mathbf{x}$ and therefore a scalar, Z is the hop length and F is the number of sampling points in the frequency domain. The result of the STFT, S, is a spectrogram. In the next step, the spectrum at each time bin, S_t, is mapped onto Mel-frequency-bins, by calculating

$$M_{m,t} = 10 \log_{10} < S_t, \Delta_m >, \quad m = 1..H, \qquad (2)$$

where Δ_m is the m-th triangular window of the mel filter bank[4]. In [1] the following values are used:

$$F = 1024, \quad Z = 512, \quad L = 1024, \quad H = 60, \quad U = 41$$

We now consider that eq. 1 is a discrete 1D-convolution with stride Z and F kernels. The k-th of these kernels is $\begin{bmatrix} v(k,1) & \dots & v(k,L) \end{bmatrix}$. Eq. 2 is equivalent to a 2D-convolution: the $U \times F$ "image" S is convolved with the $1 \times F$ filter Δ_m with stride 1 in t-direction. In k-direction this convolution is trivially just a scalar product. Now that the Mel-transformation can be described with convolutions[1], we know which main brick we should build our DeepFourier with: the convolutional layer. To approximate the *non*-convolutional parts of the above equations (e.g. the logarithm), we bet on the inherent flexibility of a neural network.

In the sections 2.2.1 and 2.2.2 we introduce two candidate DeepFourier nets, coming from papers that in fact used convolutional layers for the same purpose as we did.

2.2. New architectures

2.2.1. HeuriNet

The first DeepFourier has been inspired by [5, 6] which aimed to do speech recognition, putting convolutional layers right after raw audio inputs. Our *heuristic* argument is that since their networks perform well on speech recognition with raw audio as input, a simple upscaling of their kernel dimensions to our problem would lead to good performance also for urban sound. From this intuition follows the nomenclature *HeuriNet* := (DeepFourier, PiczakNet), the DeepFourier part being:

	layer	#in	#out	kernel width	stride	activ. func.
$\mathbf{z}^{(1)}$	CONV1D	1	80	155	10	ReLU
$\hat{\mathbf{z}}^{(1)}$	MAXPOOL	80	80	9	3	–
$\mathbf{z}^{(2)}$	CONV1D	80	60	15	4	ReLU
$\hat{\mathbf{z}}^{(2)}$	MAXPOOL	60	60	9	4	–

Table 1: The DeepFourier part of HeuriNet. "#in" and "#out" correspond to the in- and outgoing number of channels.

The network equations are

$$\mathbf{z}_p^{(1)} = \text{relu}(x * \mathbf{w}_p^{(1)}) \qquad (3)$$

$$\hat{\mathbf{z}}_p^{(1)} = \text{maxpool}_{9,3}(\mathbf{z}_p^{(1)}), \quad p = 1..80 \qquad (4)$$

$$\mathbf{z}_q^{(2)} = \text{relu}(\hat{\mathbf{z}}_1^{(1)} * \mathbf{w}_{q,1}^{(2)} + \ldots + \hat{\mathbf{z}}_{80}^{(1)} * \mathbf{w}_{q,80}^{(2)}) \qquad (5)$$

$$\hat{\mathbf{z}}_q^{(2)} = \text{maxpool}_{9,4}(\mathbf{z}_q^{(2)}), \quad q = 1..60 \qquad (6)$$

[1]that means: depending on the kernels pre-trained/learned and the activation functions used, CNNs can explore an arbitrarily small neighbourhood of the Mel-transformation.

, where all symbols (except p and q) are vector-valued and every vector can be interpreted as a time series. From [7], we have reason to believe that (after training) each of the $\mathbf{w}_p^{(1)}$'s will pick up a frequency or a combination of a few frequencies[2]. Then the $\hat{\mathbf{z}}_p^{(1)}$'s are 80 different frequency components as they evolve over time. The outputs of the second convolutional layer, the $\mathbf{z}_q^{(2)}$'s, are the temporal evolution of 60 different combinations of the $\hat{\mathbf{z}}_p^{(1)}$'s. As a simple example, $\mathbf{z}_5^{(2)}$ could contain $\hat{\mathbf{z}}_{17}^{(1)}$ (with $\mathbf{w}_{5,17}^{(2)}$ being a rectangular window and all of the other $\mathbf{w}_{5,\bullet}^{(2)}$'s being zero vectors). With such operations, we assume that HeuriNet can explore a neighbourhood of Mel-transformation (because it performs convolutions on the raw audio), but maybe not the closest neighbourhood of *Piczak*'s Mel-transformation, since its first layer kernel is not of size 1024 but 155.

2.2.2. MSTNet

Eager to try a network that will explore a neighbourhood of transformations closer to Piczak's Mel-transformation, we re-built (without changes) the architecture introduced in [7], because its inventors are able to train this network to learn Piczak's Mel-transformation with raw audio as an input. We define MSTNet:= (DeepFourier, PiczakNet), with the DeepFourier part being:[3]

	layer	#in	#out	kernel width	stride	activ. func.
$\mathbf{z}^{(1)}$	CONV1D	1	512	1024	512	ReLU
$\mathbf{z}^{(2)}$	CONV1D	512	256	3	1	ReLU
$\mathbf{z}^{(3)}$	CONV1D	256	60	3	1	tanh

Table 2: DeepFourier part of MSTNet. "#in" and "#out" correspond to the in- and outgoing number of channels.

The network equations are

$$\mathbf{z}_p^{(1)} = \text{relu}(x * \mathbf{w}_p^{(1)}) \tag{7}$$

$$\mathbf{z}_q^{(2)} = \text{relu}(\mathbf{z}_1^{(1)} * \mathbf{w}_{q,1}^{(2)} + \ldots + \mathbf{z}_{512}^{(1)} * \mathbf{w}_{q,512}^{(2)}) \tag{8}$$

$$\mathbf{z}_r^{(3)} = \tanh(\mathbf{z}_1^{(2)} * \mathbf{w}_{r,1}^{(3)} + \ldots + \mathbf{z}_{256}^{(2)} * \mathbf{w}_{r,256}^{(3)}) \tag{9}$$

$$p = 1..512,\ \ q = 1..256,\ \ r = 1..60$$

One reason why this is so close to Piczak's Mel-transformation is that the kernel length is equal to $L = 1024$ and the stride equal to $Z = 512$, the values used in [1]. Moreover, in eqns. 8 and 9 the convolutions have a very narrow kernel (only 3), and they extend over a direction with temporal meaning, implying that they will not look much into the past and future. This is similar to the aforementioned convolution that is "hidden in" eq. 2, which does not look at all into past and future (only the present). In comparison, HeuriNet's $\mathbf{w}_{q,\bullet}^{(2)}$'s in eq. 5 capture a wide time horizon, with their kernel width of 15, thus showing less similarity to Piczak's Mel-transformation. To sum it up, we expect that the MSTNet will operate in a neighbourhood rather close to Piczak's Mel-transformation.

[2]often referred to as "bandpass" in the papers

[3]Our nomenclature deviates from [7]: In the paper, only the layers *before* PiczakNet(that we callDeepFourier part) are named MSTmodel.

2.3. Methodology

2.3.1. Training with the "IceBreaker"

Training the full network starting from random initialization may lead to local minima and quickly "send" us quite far away from a configuration of PiczakNet that happened to be effective with a preceding Mel-transformation, of which we would like to explore the neighbourhood. Hence, we decided to start the training from the pre-trained PiczakNet and to smoothly increase the flexibility of the full network along epochs. To do so, we applied a 4-phase-scheme that we call the *IceBreaker*, owing its name to the fact that frozen parts of the network are "un-frozen" phase by phase. We call a weight *frozen* if we deliberately do not update it in the network training. The partial derivative of the cost function w.r.t. a frozen weight is however backpropagated in the training as if the weight was not frozen.

Phase 0: Train PiczakNet with spectrograms as input and store the optimal weights according to early stopping (i.e. taking the weights leading to the highest balanced validation accuracy).

Phase 1: Train the network with PiczakNet's part frozen. The PiczakNet weights are identical to the optimal weights from Phase 0. The DeepFourier weights are initialized with Xavier (HeuriNet) or the optimal weights from spectrogram forcing (MSTNet) (see Section 2.3.2).

Phase 2: Train the network with PiczakNet frozen from its 2nd convolutional layer to the output. The DeepFourier part is initialized with its optimal weights from Phase 1. The PiczakNet part is initialized from Phase 0.

Phase 3: Train the network with all weights trainable. The DeepFourier part and the first convolutional layer of the PiczakNet part are initialized with their respective optimal weights from Phase 2. The rest of the PiczakNet part is initialized from Phase 0.

The reason for unfreezing the first convolutional layer of PiczakNet in Phase 2 is that we want this layer to be able to adapt to possible new representations learned in the DeepFourier part. (As opposed to Phase 1, where the DeepFourier is hypothesized to stay relatively faithful to the Mel-transformation.)

2.3.2. Initialization

For the stand-alone PiczakNet, we used Xavier random initialization for weights and zero initialization for biases.

For MSTNet's DeepFourier layers, we apply a pretraining that we call *spectrogram forcing:* the DeepFourier layers of MSTNet are trained stand-alone with the normalized Mel-spectrogram as a target, like in [7]. The cost function is the mean square error between the network prediction to an observation and the normalized Mel-spectrogram of the observation.[4] We used the same optimizer as in [7] (ADAM with a learning rate of $3 \cdot 10^{-4}$). The normalization is performed by centering around the mean and dividing by the range (maximum minus minimum) of the Mel-spectrograms over the entire data set. So the target Mel-spectrograms consist of values between -1 and 1 (as in [7]), which fits to the tanh in eq. 9. As a consequence, PiczakNet's pretrained weights are different for MSTNet (where training is based on normalized spectrograms) and HeuriNet (where spectrograms are not normalized, like in Piczak's original work).

As HeuriNet DeepFourier's layers do not aim to catch Piczak's Mel-transformation (and anyways the final activation function is ReLU, not well suited for normalized spectrograms), HeuriNet's weights have been randomly initialized (Xavier), without spectrogram forcing.

2.3.3. *Training operation*

Like in [1], we used stochastic gradient descent with Nesterov momentum of 0.9 as a training procedure, with mini-batches of size 1000, 300 epochs for Phase 0 and 100 epochs for each next phase. The learning rate was fine tuned ($5 \cdot 10^{-3}$ for MSTNet and $2 \cdot 10^{-3}$ for HeuriNet) and gradient clipping was applied for MSTNet thresholding each layer's gradient's L2 norm to 3. Indeed, these L2-norms reached some high peak values that could not be countered using learning rate decay.
Let us note that classes are unequally represented in the dataset and some models led to misclassification bias towards low-represented classes. We tried to balance each mini-batch doing oversampling, but it did not improve the class balanced accuracy, hence in the end we did not include any balancing procedure in the training operation.

2.3.4. *Performance evaluation*

In order to evaluate each network's performance, one should ideally use cross-validation. Due to time constraints and for simplicity, we split the data into three folds and only performed one run for this configuration:
80% of the data for training;
10% of the data for validation (for learning rate tuning and early stopping);
10% of the data for testing and assessing the model performance.
Because of the class imbalance, we used the class balanced accuracy (i.e. the recall averaged over all classes) as a performance measure in order not to discriminate against any class. In order to assess the "guiding effect" of the IceBreaker, we compared HeuriNet and MSTNet performances using the IceBreaker with their equivalent networks' where all weights are trainable from the beginning and randomly initialized (Xavier). In fig. 4, we will call this training process "random init".

[4] since in [7] a majority voting among all the observations of the same audio clip was used, the authors did not calculate the Mel-spectrogram of each segment individually as we did.

3. RESULTS

The training and balanced validation accuracies for each network are presented in fig. 3. One can see that MSTNet tends

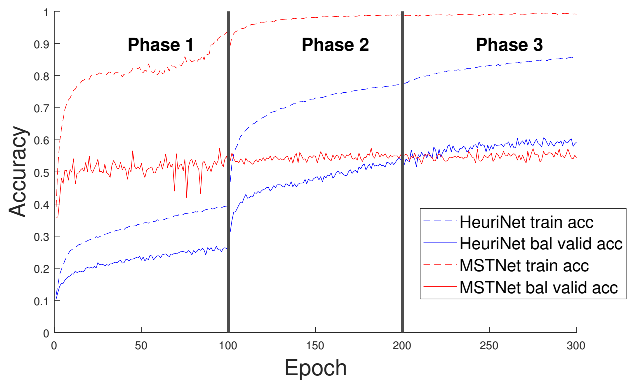

Fig. 3: Training and balanced validation accuracies of HeuriNet (blue) and MSTNet (red) during Phase 1, 2 and 3 of the IceBreaker.

to overfit a lot, especially from Phase 2 where it reaches almost 100% training accuracy, while the validation accuracy is quickly stable around 50-55%. Indeed, MSTNet seems to benefit a lot from the spectrogram forcing but not so much from the unfreezing of the network. Meanwhile, both training and balanced validation accuracies of HeuriNet keep increasing as the network unfreezes, especially during Phase 1 and Phase 2. This suggests that the step by step introduced flexibility helps learning a better classifier. Results on the test set are presented in fig. 4. Both HeuriNet and MSTNet following the IceBreaker reach a performance comparable to PiczakNet. This means that the representations learnt in HeuriNet and MSTNet are as good as the Mel-spectrogram in terms of final performances. Again, one can see that MSTNet benefits a lot from the spectrogram forcing compared to its randomly initialized version that reaches a lower final test accuracy, falling into a local minimum. HeuriNet does not seem to benefit so much from the IceBreaker since its randomly initialized version performs as good: local minima do not seem to be a problem here, and it needs only its flexibility to reach its highest performance.

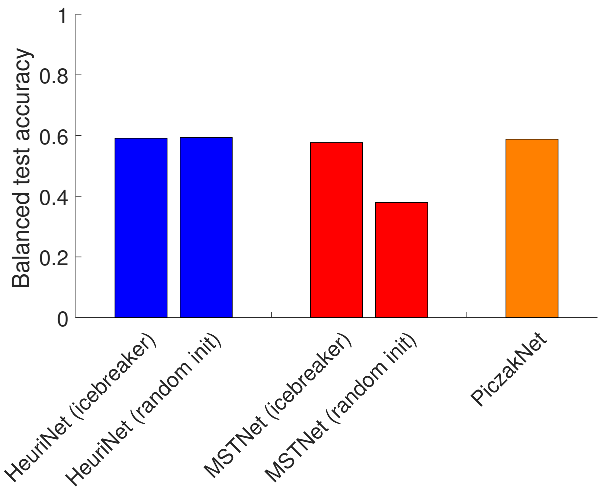

Fig. 4: Performances of HeuriNet and MSTNet compared with PiczakNet, from both the IceBreaker and full random initialization training procedures.

4. INTERPRETATION

4.1. Activations and resulting data transformation

We looked at the activation[5] of 10 example observations right after the DeepFourier layers (i.e. right before the PiczakNet part), at Phases 1-3 of the IceBreaker. This we did in order to find out whether the changes in accuracy in fig. 3 correlate with changes in the representation of the data. The example observations were picked randomly from the test set. We show one of these observations (the last class) in this section and all 10 in appendix C.

4.1.1. Activation analysis: HeuriNet

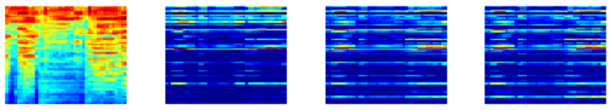

Fig. 5: Activation of an observation of class "ST", with HeuriNet. From left to right: Mel-spectrogram, IceBreaker Phase 1, Phase 2, Phase 3.

From fig. 6 it appears that the transformation learned by HeuriNet is not even close to the Mel-transformation. It is important to point out that we never forced the kernels to be *ordered* in terms of their dominant frequencies (i.e. we never forced $\mathbf{w}_1^{(1)}$ to detect a lower frequency than $\mathbf{w}_2^{(1)}$ and so on). So the transformation obtained by HeuriNet *could* be related to a spectrogram, but could be "shuffled" because of the lack of ordering. Also, looking at the activations, we see that HeuriNet obtains a much more sparse response than the corresponding Mel-spectrogram. This seems to point out that HeuriNet focuses on only the more salient features of each observation, disregarding (or not being able to capture) some of the finer "details" of the data. This could be justified by considering the relatively small dimension and amount of parameters of the network. Whether this behaviour is desirable or detrimental could be further investigated, for instance by making the network face a less "clean" dataset, with noisy or unclear samples, and see if this "focus on the most prominent information" works in favour or against the classification task. We see that the obtained representation varies mostly between Phase 1 and Phase 2, similarly to the validation curves in fig. 3. This suggests that the first convolutional layer of PiczakNet is quite critical in terms of learning: although it does not contain most of PiczakNet's parameters[6], making it trainable is a key ingredient for increasing the performance.

4.1.2. Activation analysis: MSTNet

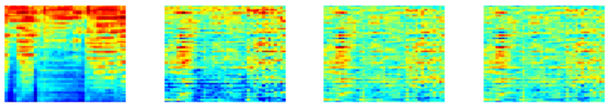

Fig. 6: Activation of an observation of class "ST", with MSTNet. From left to right: Mel-spectrogram, IceBreaker Phase 1, Phase 2, Phase 3.

MSTNet achieves a transformation that is much closer to Piczak's Mel-transformation, which is expected due to the spectrogram forcing (section 2.3.2) and MSTNet's architecture. It is also interesting to notice how little the transformation changes across the IceBreaker phases. Again, this is consistent with the validation accuracy curve in fig. 3, which doesn't change a lot across the IceBreaker phases either.

4.2. Kernel weights

Since our MSTNet and HeuriNet have very different behaviours and activations, we want to analyze in which sense they differ or are alike. Hence, we decided to analyze the best weights of the first layer of the DeepFourier for each network (from Phase 3). Indeed, this layer is in direct "contact" with raw audio which can give us some clues as to what information gets picked up from the waveform. In fig. 7 we have plotted a few examples of these first-layer-weights.[7] Upon inspection, it is clear that these weights exhibit periodical behaviour. Furthermore, since they act on the raw audio directly, and their shape seems to resemble

[5]We call the *output* of the activation functions "activations". This is in contrast to [8], who calls the input of the activation functions "activations".

[6]most of the parameters are in the dense layers, frozen until Phase 3.

[7]Only from HeuriNet, since many kernels from MSTNet are not so readable since they are dominated by higher frequencies.

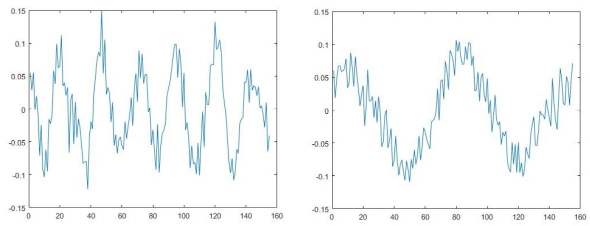

Fig. 7: Some of the 80 kernels from the first 1D convolutional layer (= the $\mathbf{w}_p^{(1)}$'s) of HeuriNet.

sinusoidal curves superimposed over each other, we can assume that they select certain combinations of frequencies. To make a comparison between the HeuriNet and the MSTNet architecture, we can take the FFT of each kernel, namely "$p-$ th kernel spectrum := FFT($\mathbf{w}_p^{(1)}$)" for both models. fig. 8 shows these kernel spectra side by side. Beware that we

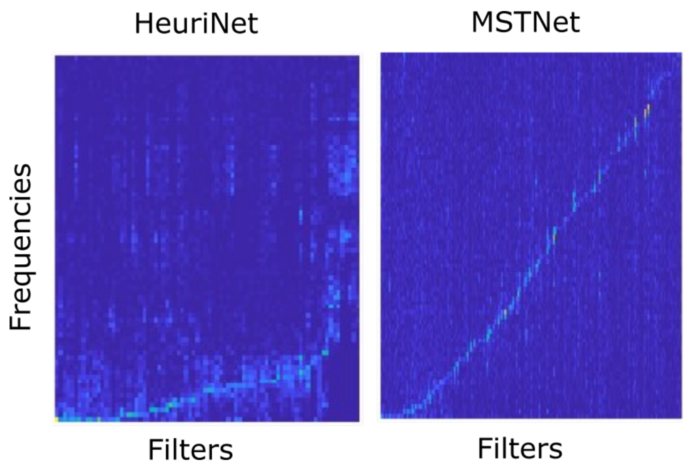

Fig. 8: The FFTs of the $\mathbf{w}_p^{(1)}$'s for the two different DeepFourier variants. Beware that we sorted these kernel spectra w.r.t. their dominant frequencies post hoc.

sorted the kernel spectra post hoc, so that the kernel spectra where the lower frequencies dominate are found on the left, and the kernel spectra where the higher frequencies dominate are found on the right. This "nice order" is *not* delivered by the training. We see that (at least in the first layer!) the HeuriNet puts more emphasis on the lower frequencies, while the MSTNet seems to put equal weight over the whole range of frequencies that are represented in the filter-bank. Another difference is the sparsity: the MSTNet figure shows much more contrast, by specifically amplifying certain frequency bands, while in the HeuriNet we can see some more dispersed values. All in all, we see that there are indeed differences between HeuriNet and MSTNet in the way that their first layer picks up frequencies. However, this alone is not enough to draw a conclusion, since the second (and third) DeepFourier layer also contribute.

5. CONCLUSION AND FUTURE WORK

According to the random initialization results, the strength of HeuriNet lies more in its architecture rather than on the guiding effect of the IceBreaker, while MSTNet owes its performance probably only to the spectrogram forcing. So we have no proof that the IceBreaker's unfreezing process helps. Testing on a noisier dataset would help arguing for or against the benefit of the IceBreaker procedure.

Our results suggest that the representations learnt in our DeepFourier architectures offer comparable performances as the Mel-spectrogram for one-second-segments. Two hypotheses might explain why we do not achieve better:

H1: PiczakNet is a bottleneck that limits the performance, meaning that even if DeepFourier is "better" than Piczak's Mel-transformation, no better performance is reached

H2: There exist DeepFourier hyperparameters that result in better performance, but we haven't found them

H1 could be tested by taking other architectures from the literature with the same purpose as PiczakNet.[8] For H2 one interesting experiment would be to design an architecture where the DeepFourier part starts off by being exactly equivalent to a Mel-transformation. That means a two-layer CNN where the first layer's kernels would be initialized with complex exponentials, the second layer's kernels would be initialized with triangular windows, the biases would be zeros, and a candidate for the first layer's activation function would[9] be $|\bullet|$ (eqns. 1 and 2). With this initialization, the DeepFourier part would start in the *closest possible* neighbourhood of a Mel-transformation, and by training, i.e. learning new filters, explore that neighbourhood in search of better performance.

We underline that we oriented our architecture choice towards understanding rather than brute forcing. But optimization of hyperparameters (kernel width, number of filters, etc.) and regularization techniques *shall be performed*, and ideally with more rigorous performance evaluation methods such as cross-validation. HeuriNet, for one, reached similar performance to PiczakNet although its hyperparameters come "just" from a heuristic and are not so fine-tuned, so it might be possible to achieve even better performance with fine-tuning. And MSTNet gets equivalent performance although it overfits.

Finally, results for MSTNet with spectrogram forcing suggest that being in a close neighbourhood of a Mel-transformation can help networks with a high number of parameters, since they would otherwise fall into local minima or saddle points.

[8]Trying to fine-tune PiczakNet's parameters might also be an option, even though we think that Piczak has already spent some time on that.

[9]$|\bullet|^2$ is not an option, even though it would be more faithful to the Mel-transformation. The reason is that $|\bullet|^2$ might provoke exploding gradients.

6. REFERENCES

[1] Piczak K. J., “Environmental sound classification with convolutional neural networks,” in *Machine Learning for Signal Processing (MLSP): 25th International Workshop on Machine Learning for Signal Processing.*, Boston, USA, IEEE. Sep. 2015.

[2] Salamon J., Jacoby C., and Bello J. P., “A dataset and taxonomy for urban sound research,” in *22st ACM International Conference on Multimedia (ACM-MM’14)*, Orlando, FL, USA, Nov. 2014.

[3] Stevens S. S., “A Scale for the Measurement of the Psychological Magnitude Pitch,” *Acoustical Society of America Journal*, vol. 8, pp. 185, 1937.

[4] Muda L., Begam M., and Elamvazuthi I., “Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques,” in *JOURNAL OF COMPUTING*, 2010, vol. 2 issue 3, p. 139 fig. 3.

[5] Collobert R., Puhrsch C., and Synnaeve G., “Wav2letter: an end-to-end convnet-based speech recognition system,” in *arXiv:1609.03193v2 [cs.LG]*, 2016.

[6] Palaz D., Collobert R., et al., “Analysis of cnn-based speech recognition system using raw speech as input,” Tech. Rep., Idiap, 2015.

[7] Maaløe L., Purwins H., Antich, J. L. D., and Tax T. M. S., “Utilizing domain knowledge in end-to-end audio processing,” in *31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[8] Bishop C. M., “Pattern recognition and machine learning,” Springer, 2006, p. 227.

A. CLASSES OF THE URBANSOUND8K DATA

AI	air conditioner
CA	car horn
CH	children playing
DO	dog bark
DR	drilling
EN	engine idling
GU	gun shot
JA	jackhammer
SI	siren
ST	street music

B. ABBREVIATIONS AND NOTATION

CNN	Convolutional Neural Network
FFT	Fast Fourier Transformation
STFT	Short-Term Fourier Transformation
$<,>$	denotes the scalar / dot product of vectors
$*$	denotes the discrete convolution
i	is the imaginary unit
$\mathbf{x}$	is a 1×20992 vector and is one of the segments
	$\mathbf{x}_{(t-1)Z+l}$ is the "$(t-1)Z + l$"-th element of $\mathbf{x}$
$\mathbf{w}^{(2)}$	is a 3-dimensional tensor
$\mathbf{w}^{(2)}_{1,1}$	is an element of $\mathbf{w}^{(2)}$, is a vector, and a convolutional kernel
k	is the frequency sample index
m	is the Mel-frequency-bin index
t	is the time bin index
Z	is the hop length
L	is the kernel length
F	is the number of sampling points in the frequency domain
H	is the number of Mel-frequency-bins
U	is the number of time bins
S	is a $U \times F$ matrix and is a spectrogram
S_t	is a $1 \times F$ vector and is the t-th row of S
$S_{t,k}$	is a scalar and is the element at the t-th row and k-th column of S
M	is a $H \times U$ matrix and is a Mel-spectrogram
$M_{m,t}$	is a scalar and is the element at the m-th row and t-th column of M

C. ACTIVATIONS RIGHT BEFORE PICZAKNET

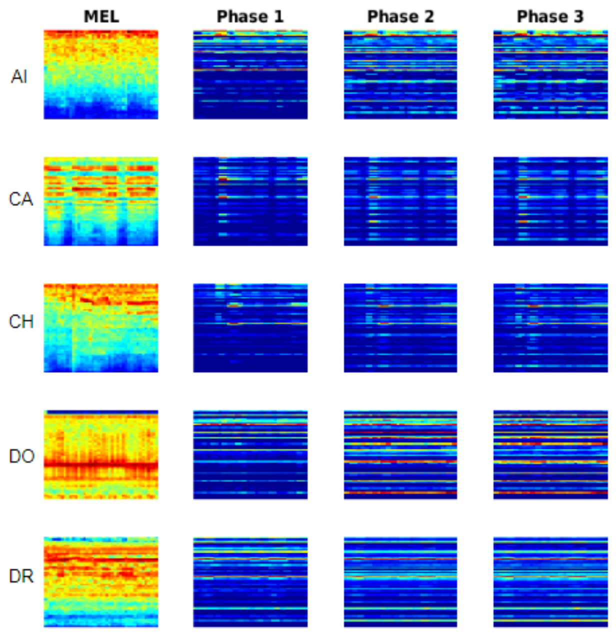

Fig. 9: HeuriNet classes 1-5

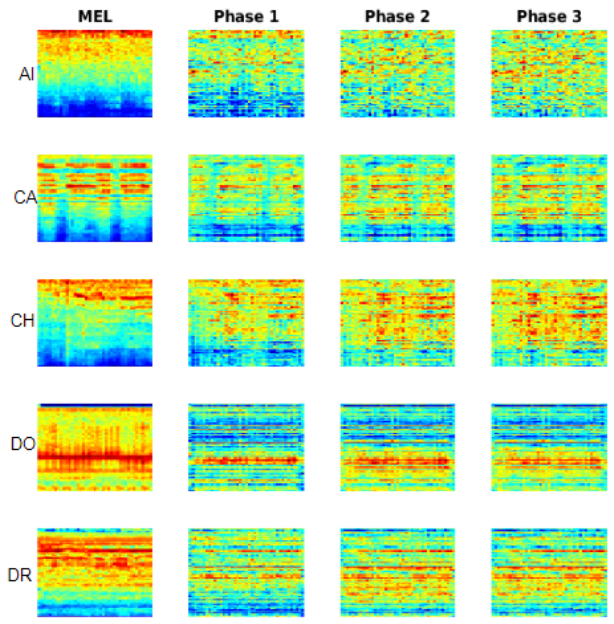

Fig. 11: MSTNet classes 1-5

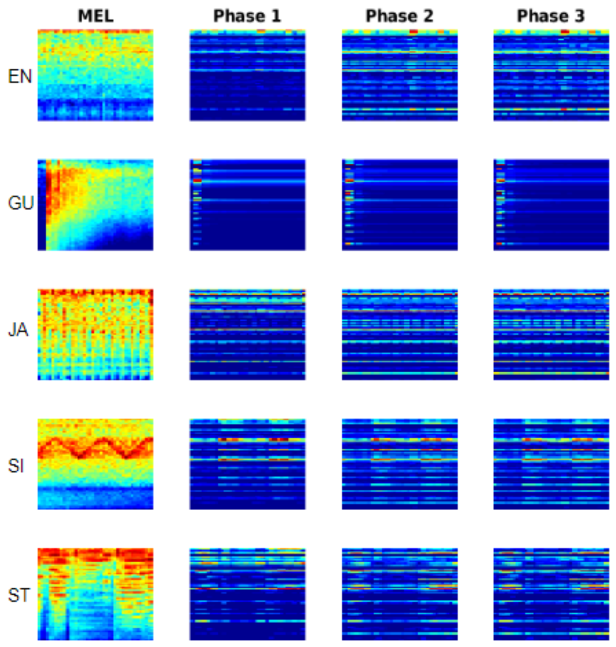

Fig. 10: HeuriNet classes 6-10

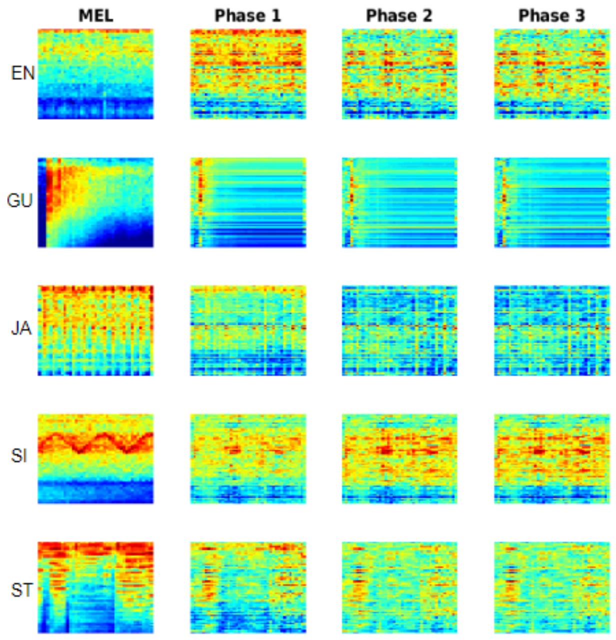

Fig. 12: MSTNet classes 6-10