FEW-SHOT BIOACOUSTIC EVENT DETECTION: DON'T WASTE INFORMATION

Technical Report

Michael Hertkorn

ZF Friedrichshafen AG

ABSTRACT

In the past a lot of attention has been dedicated into finding a good neural network architecture, mainly adopting large NN architectures found in image processing.[1] The parameters in the fixed preprocess, which usually consists of a short-time Fourier transform (STFT) and optionally adding a Mel or Mel frequency cepstral coefficient (MFCC) transformation, can be made trainable[2], however some major parameters stay fixed, like the window size and the fact that the absolute of the complex output of the Fourier transformation is calculated. Also, a learnable frontend is not desirable for a few-shot training setting.

This investigation shall demonstrate the importance of choosing suitable parameters for the acoustic preprocess. In order to do this, a standard CNN with a minor tweak is used and the pretraining with training data has been skipped which means that the model is only trained on the 5 shots provided in the validation and evaluation datasets, similar to the pattern matching baseline.

1. MOTIVATION

In last years DCASE 2021 task 5, we saw that a prototypical network does not always outperform a model that is not pretrained on a separate training dataset, but only uses the 5 shots from the validation and evaluation dataset, as the template matching baseline had achieved a much better f-measure on the evaluation dataset. However, template matching is not state-of-the-art anymore. To make this investigation simple, I decided to use a standard CNN with only minor mandatory modifications.

After checking the baseline repository for this challenge[3] and I noticed that the window length is set to a rather high value. This can lead to blurry STFT spectrograms, depending on the target sound. After trying several parameters for the FFT window length, I found a good compromise for all sounds in the training set, which led to much sharper spectrograms. A classifier will have an easier job of differentiating those clear lines from background noise.

A Fourier transformation will keep all information, as it can be transformed back to the original signal, however for most acoustic preprocesses the complex values in the STFT are made real by calculating the absolute value, which reduces the available dimensions by half, dropping a lot of important information. This means that if we do not choose the parameters for the Fourier transformation well, we will remove important information, making the task more difficult and reducing the overall system performance.

Most CNN architectures reduce the spacial dimensions to a very low number while increasing the number of kernels used. For image classification this approach makes sense as the absolute position of the object in the image doesn't matter, so the reduction of the spacial dimensions will lead to a so called translation invariance, which is beneficial for image classification. For audio classification the frequency axis contains more information than just the location of the object. It is therefore beneficial to not reduce the dimensions on the frequency axis too much.

2. IMPLEMENTATION DETAILS

2.1. Data Preparation

A fixed FFT window length is used: 256 (submissions 1 and 3) or 512 (submissions 2 and 4). Those values showed the best sharpness in the spectrogram representation. Those values would have been adjusted manually in a real world sound event detection development for the evaluation data, but as subjective judgments are forbidden, I decided to use those two values for the FFT window length.

I used the same noise reduction techniques as being used in the template matching baseline.[3]

One training sample shall consist of a 2D spectrogram. The frequency dimension is dictated by the FFT window length and is not cropped. The time dimension is calculated to be within a certain constraint and ideally has the same value as the shortest sequence of all negative and positive samples. After that, up to 1000 overlapping training samples are drawn from each of the positive and negative samples that are available in the first 5 shots of the validation data.

2.2. Model Architecture

As a model, I used the most simple baseline for a CNN model: the MNIST tensorflow tutorial.[4] As we will have too many dimensions left after the convolutional layers, becaue the input is larger for this challenge compared to MNIST, we need to extent the model with additional pairs of Conv2D- and MaxPooling2D-layers. In order to not lose the frequency resolution, the pooling layers will not pool along the frequency axis at this point. Additional Conv2D- and MaxPooling2D-layers are added until not enough dimensions are available.

Table 2 gives a summary of the used model for the audio file "ME1.wav". For this file, two additional pairs of Conv2D and MaxPooling2D are added to reduce the dimensions before passing over to the fully connected layers.

model name	parameters			validation set scores			subfolder f-measure		
	n_{fft}	#epochs	min time threshold	f-measure	precision	recall	HB	ME	PB
Hertkorn_ZF_task5_1	256	10	25%	60.58%	49.71%	77.54%	**87%**	53%	52%
Hertkorn_ZF_task5_2	512	10	25%	61.76%	56.19%	68.55%	84%	62%	49%
Hertkorn_ZF_task5_3	256	10	50%	**67.87%**	**68.42%**	67.33%	77%	**69%**	**60%**
Hertkorn_ZF_task5_4	512	5	50%	60.53%	58.90%	62.25%	78%	54%	55%
submission 1 with sr//10	sr//10	10	25%	24.13%	37.54%	17.78%	76%	25%	14%
submission 1 with 2x2 pooling	256	10	25%	51.45%	38.11%	**79.18%**	86%	39%	48%
baseline template matching	sr//10	-	-	3.36%	1.82%	21.60%	21%	44%	1%
baseline prototypical network	1024	10	-	29.59%	36.34%	24.96%	?	?	?

Table 1: Model performance summary.
"sr//10" means 10% of sampling rate. "2x2 pooling" means always using 2x2 pooling (not 1x2 pooling in order to preseve frequency axis)
For some reason the reproduction of the baseline template matching model didn't produce the same results as published on the challenge webpage. The results for the prototypical network were just copied from the webpage and not recalculated.

Model: CNN

Layer	Output Shape	Param #
conv2d	(None, 127, 48, 32)	320
max_pooling2d	(None, 63, 24, 32)	0
conv2d_1	(None, 61, 22, 64)	18496
max_pooling2d_1	(None, 30, 11, 64)	0
conv2d_2	(None, 28, 9, 32)	18464
max_pooling2d_2	(None, 28, 4, 32)	0
conv2d_3	(None, 26, 2, 32)	9248
max_pooling2d_3	(None, 26, 1, 32)	0
flatten_2	(None, 832)	0
dropout_2	(None, 832)	0
dense_6	(None, 10)	8330
dense_7	(None, 10)	110
dense_8	(None, 1)	11
Total params: 54,979		
Trainable params: 54,979		
Non-trainable params: 0		

Table 2: Model summary.

2.3. Training

I used binary crossentropy as loss and nadam as optimizer. I trained the model for 10 epochs (submissions 1-3) or 5 epochs (submission 4). The reason for training for 10 epochs is that I want the model to slightly "overfit" to the data, as we see very similar sounds for the positive samples in the training data. So we want to only trigger on sounds that are very similar to the first 5 shots. Increasing the number of epochs usually reduces false positives while increasing false negatives. If the loss does not reach a certain limit, the training is continued for another set of epochs until that limit is reached.

2.4. Postprocessing

One important aspect due to the way the scores are produced is to avoid large numbers of false positive events, as those play a very big role even though the events might be very short.

To solve this issue, a smooth function is used which calculates the average positive event duration for the first 5 shots, then drops all predicted events that are shorter than 25% (for submissions 1 and 2) or 50% (for submissions 3 and 4) of the average positive event duration. I also stitch all events together that are only 15% of the average positive event duration apart.

3. RESULTS

Besides the four submission systems, two more tests were executed to show the influence of an improper sampling rate or by dropping information in the CNN due to pooling too much along the frequency axis. In table 1 the performances of the systems are shown.

We can clearly see comparing the models "submission 1" and "1 with sr//10" that the influence on the f-measure is very large depending on the chosen FFT window size. Comparing models "submission 1" with "1 with 2x2 pooling", we can see that the pooling strategy that perserves frequency information also increases overall system performance. We can also see that the influence can be very small (on subfolder HB) or large (on subfolder ME).

4. DISCUSSION

The task of creating an acoustic event classifier given only a few shots of labeled data is difficult enough, but it's even more difficult that the background sound scapes of the evaluation data are unknown, as well as the typical event duration. In a real world setting, these informations are usually available.

For some audio files, a special call of an animal has to be detected, and other calls need to be ignored. On other audio files, any sound of that animal has to be detected. A different training setup could be chosen to improve performance if it was known what kind of sound has to be detected.

The label procedure seems to be different between the datasets: For the WMW dataset, the long labels seem to also catch breaks between the actual sound. For example, in "XC26959.wav" during the event which lasts from 41.19s to 54.54s, the audible signal pauses at 48s. If a classifier would consider this pause as a negative and will start a new event, that would be counted as a false positive, thus reducing the score of the classifier. Another approach would be to create a grid on the time axis and only allow labels and predictions on that grid. The spacing could be 0.1s. Then for each of the 0.1s intervals, it is checked if the classifier produces the same output as the ground truth label, and calculate false positives and false nega-

tives and f1 scores. Longer events will have a larger impact on the overall score, but it solves the issue of guessing the label procedure that has been used.

Another issue rises due to the way the score is calculated. If for a single wav file, the model was not able to learn the correct audio signature and is therefore detecting too many events, the FP score for the complete subfolder will be high, reducing the f-measure for this subfolder. As the harmonic mean is used to combine all subfolders, one bad f-measure will drastically reduce the overall system performance.

5. REFERENCES

[1] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. [Online]. Available: https://arxiv.org/abs/1609.09430

[2] N. Zeghidour, O. Teboul, F. de Chaumont Quitry, and M. Tagliasacchi, "Leaf: A learnable frontend for audio classification," *ICLR*, 2021.

[3] https://github.com/c4dm/dcase-few-shot-bioacoustic.

[4] https://keras.io/examples/vision/mnist_convnet/.