Sparrow

Input

My input is a mel spectrogram with 80 bins from 50 Hz to 10 kHz, computed on GPU (you'll see why) from ordinary spectrograms (22050 Hz samplerate, frame length 1024, 70 frames per second).

Architecture

I used a CNN of the following architecture:

Conv 32 @ 3x3, batchnorm, leaky relu
Conv 32 @ 3x3, batchnorm, leaky relu
Max-Pool @ 3x3
Conv 32 @ 3x3, batchnorm, leaky relu
Conv 32 @ 3x3, batchnorm, leaky relu
Conv 64 @ 3x19, batchnorm, leaky relu # at this stage, the input has 21 frequency bins, so this convolution leaves 3 bins
Max-Pool @ 3x3 # this pools over the 3 bins left by the previous convolution
dropout 0.5
Conv 256 @ 13x1, batchnorm, leaky relu
dropout 0.5
Conv 64 @ 1x1, batchnorm, leaky relu
dropout 0.5
Conv 1 @ 1x1, linear

This gives a single output for 139 frames (~2 seconds). I.e., if the input to this network has 139 frames, the Conv 256, Conv 64, Conv 1 are equivalent to three dense layers. For longer inputs, it will give one output every 9 frames (because we have 3-fold time pooling twice), for overlapping 2-second excerpts.

My reasoning now was as follows: To detect a bird in an audio recording, 2 seconds should be enough (that's what I designed the architecture for). A longer audio excerpt should be labeled as positive if (and only if) it contains a bird somewhere. In fact, the label should not be any different for a file that has a single chirp and 9 seconds of silence, or a single chirp and 30 seconds of silence, or one hundred chirps. So the global label for a file should be the *maximum* over the local predictions. Thus, the network continues with:

GlobalMaxPool, sigmoid

During training, this means the gradient will only affect the network at the position it produced the maximum output. (This also makes great sense if the input file contains a single chirp somewhere -- we only want it to learn to label this chirp as positive, not anything else in the file. For negative examples, the network will gradually learn to suppress any large output.)
Unfortunately, it's easy for the network to cheat: To drive the loss to zero, it only has to output a single positive local prediction per file, and negative predictions everywhere else, and it can learn this by heart. Also it's a bit risky to have test predictions depend on a single local prediction. I found it improves things to put a sliding temporal average before that. So the new ending becomes:

Conv 1 @ 5x1 with all weights fixed to 0.2 (i.e., a sliding average)
GlobalMaxPool, sigmoid

Training

I trained on excerpts of 301 frames (which are allowed to wrap around the edge of a file), with binary cross-entropy, ADAM with initial learning rate 0.001 reduced by 0.1 every 16000 updates, for 40000 updates in total, using mini-batches of 32 excerpts.
For positive input files, it's possible that such short excerpts do not contain a bird at all, but this still worked better than using longer excerpts of 701 frames -- possibly because it allowed the network to learn about additional chirps in a file of multiple chirps where it got one of the local predictions correct already. For all submissions but the last, I trained on 70% of the data and validated on 30%.

Data Augmentation

I used two augmentation techniques:
1. Random pitch-shifting: For every batch, I chose a random factor between 0.9 and 1.1, and multiplied the minimum and maximum frequency of the mel filterbank with it (which is applied on-the-fly on the GPU). This stretches or shrinks the full filterbank and is a good and cheap approximation to pitch-shifting.
2. Random noise mixing: I choose the central frames of the last 8 excerpts of each mini-batch, and add them to the first 8 excerpts of each mini-batch, with random mixing coefficients between 0.0 and 0.4 for the noise (and the converse factor for the original excerpt). This adds a kind of static noise floor. I also tried mixing the full excerpts instead, adjusting the labels accordingly, but this worked a little worse on the validation set.

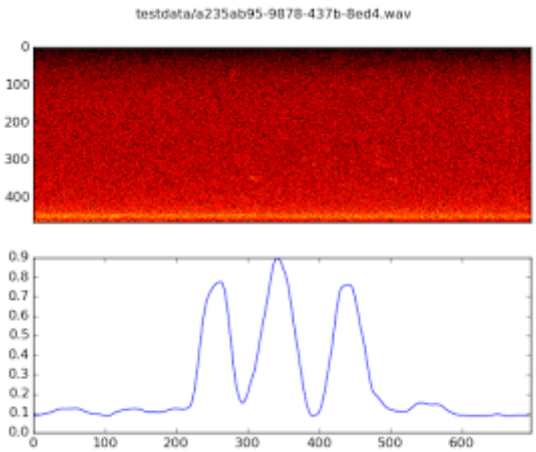
Prediction

At test time, I convert the network such that Max-Pooling is fully-overlapping (stride 1) over time, and the subsequent convolutions and poolings use temporal dilation (3 after the first pooling layer, 9 after the last). This allows me to get a prediction at every frame, not only at every 9th frame. I change the running average to use 45 frames instead of 5, and continue taking the global maximum in the end.

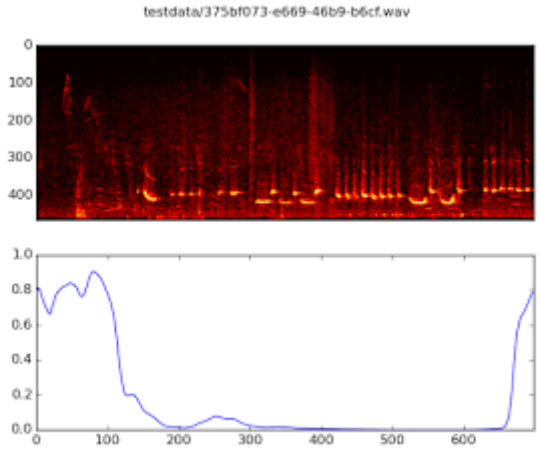
Results

On the validation set, my single best model gets 96.6% AUROC and 8.2% classification error, and a bag of five models get 96.9% and 7.8%. On the test set, this ensemble got 84.65% AUROC.

Due to the maximum over time, it's very sensitive and detects single chirps in a file full of noise:



It also detects real birds in the background of somebody imitating a bird (I'm looping the file to be able make predictions near the edge, that's why the curve increases at the end again):



But it also falls for some false positives, and it often has quite high activation for the noise floor (e.g., 0.1 in the first example above). A possible explanation is that while 2 seconds are enough to detect a bird, it's not enough context to understand and remove the noise floor of the current recording.

Other things tried

For the last submission, I tried to clean up the training data a bit, which has a lot of false annotations. I trained three models on 50% of the data and three more on the other half, and had them label the data they were not trained on. If all models individually confidently agreed that a training label was wrong (all < 0.1 for a file labeled positively, or all > 0.9 for a file labeled negatively), I removed the file from training (this affected 365 files). I also took all test files the models confidently agreed on and included them in training, and I included the validation set in training. Altogether, this reduced test set performance from 84.65% to 82.25%. Well...

Hope this is useful!

Best, Jan

PS: I can also make the code available upon request, it would just need to be cleaned up. It's built on open source tools (Python, numpy/scipy, Theano, Lasagne).

detection/48fc65d9-f94e-4619-872f-9ee213b55b77%40googlegroups.com.
For more options, visit https://groups.google.com/d/optout.