EXPLORING COMPACT ALTERNATIVES TO DEEP LEARNING IN TASK 1B

Technical Report

Prachi Patki

prachie.26@gmail.com

ABSTRACT

Task 1b appeared primarily geared toward finding compact deep learning models; however, our experience is that other methodologies may sometimes achieve similar accuracies with substantially smaller parameter counts. We focused on finding alternative classifier formulations that significantly reduce complexity while still achieving superior results. Our primary submission, based on a multi-channel SVM formulation, performs better than the reference design on test data, but requires only ~17.5 KB in parameter complexity.

1. APPROACH

The dcase Task 1b challenge focus was primarily geared toward finding compact deep learning models; however, our experience is that other methodologies may sometimes achieve similar accuracies with substantially smaller parameter counts. Parameter counts are referred to as "complexity" in this challenge. We focused on finding classifier formulations that significantly reduce complexity while still achieving superior results.

To do so, we explored standard low computational costs machine learning methods in combination with an extremely large space of candidate feature extraction methods.

Our first submission has only a 17.5 KB complexity score but achieves better results on the provided test data set than does the dcase reference solution with complexity close to 500 KB. We are interested to see if our compact, computationally simple classifier compares with other more advanced deep models.

2. COMPLEXITY AND MODEL STRUCTURE

In the spirit of the challenge, we count all learned parameters in the complexity total. The primary decision engine in our core model is a three class, one-vs-one, Support Vector Machine (SVM). In parallel with a neural network layer, each binary linear in an SVM classifier includes a weight vector and a bias. The weight vector, commonly referred to as w or β, is of vector length matching the input feature count, while the bias b is a single parameter value. [1] If input to an SVM is standardized, then it also includes μ and σ parameters of the same length as the weight vector for z score mapping of each feature column. β, μ, σ for each binary classifier form the bulk of the complexity in terms of stored learned parameters. Single value parameters, such as bias, are comparatively insignificant, but were also counted in the numbers.

The one-vs-one formulation requires three binary classifiers, and adds an ECOC (error correcting output coding) matrix for collapsing to a final classification decision.

Parameters for the all models created were encoded to 16 bit (2 byte fixed point) with no significant loss of accuracy. This version was used for assessment.

A linear SVM classifier formulation is both compact and extremely fast to compute, making it a very practical choice for embedded, TinyML solutions.

3. FEATURE AND DATA CHANNEL SELECTION

In order to leverage a comparatively simple decision engine, optimization of input features is key. We employed a proprietary search and optimization method to explore and refine over possible choices of both input channels and feature types. Again, in keeping with the spirit of the challenge, only features computable with common, standardized code were considered for submission. Specifically, frequency, cepstral, and wide variety of other feature spaces appropriate to acoustic and time-frequency signals, but which are computable with algorithms that are not dependent on learned filters or a large number of custom parameters. Basically, these are items that would be available to most embedded engineers based on common DSP functions.

Our results showed that Mel spectral features [2] were well suited. These were optimally computed on the loudest and quietest regions of each sound file, with an STFT mapped to 500 spectral bands using standard formulas. Our results also showed an optimal data input comprising the sum and difference channels (L+R and L-R) rather than using the raw audio channels individually or just using data in mono.

Our analysis also detected that a subset of the sound files actually appeared to be strict mono, and therefore had a hard zero difference channel. Since lack of true stereo generally resulted in errors, we created three variations of our base classifier for comparison -- one as described, another with only mono (L+R) input, and a third that included both options, but only applied the mono classifier in the case that the difference channel was zero.

On their face, these feature optimization results make sense. For example, the loudest and quietest regions of a sound file likely

highlight the most significant transient sounds and the best estimate of background noise, respectively. Likewise, inclusion of the difference channel allows the classifier to compare directional and non-directional sound sources.

We observe that the reference DL model expends a large number of complexity parameters in the two CNN layers on learning the "shape" of Mel spectral patterns in a spectrogram. In contrast, our method results in analysis of a sparse selection of prominent transient sounds, in a fixed feature space and is significantly more compact.

4. FURTHER OPTIMIZATION

Because these models are already much smaller than the reference DL design, we did not spend a great deal of effort further shrinking the parameter count. However, standard methods of feature pruning could be used to shrink them further if necessary. For example, we identified that at least 8% of the complexity parameters were associated with feature weights (beta) vanishingly close to zero and could probably be dropped in a live deployment with little or no effect on output.

5. RESULTING MODELS

We submitted three variations of our classifiers. None of these used any outside or third party data, but were trained only on the provided examples.

An SVM typically produces a class score in the range [0, -∞). Since we understood that only final classification, not actual probabilities, were required for assessment, we renormalized our submitted class scores to relative values between 0 and 1, but took no other special steps to estimate actual likelihoods.

As discussed above, model 1 used L+R and L-R as input. Model 2 used either that classifier or a second mono (L+R) classifier, depending on whether a mono signal was detected. Model 3 is simply the mono classifier; this is included only for comparison, as it is not ideal in most cases.

The accuracy numbers below are for each model, trained on the provided training set and tested against the provided test set.

5.1. Model 1: (L+R) & (L-R) Input channel

Complexity: 17.5 KB

Macro Average Accuracy: 88.4%

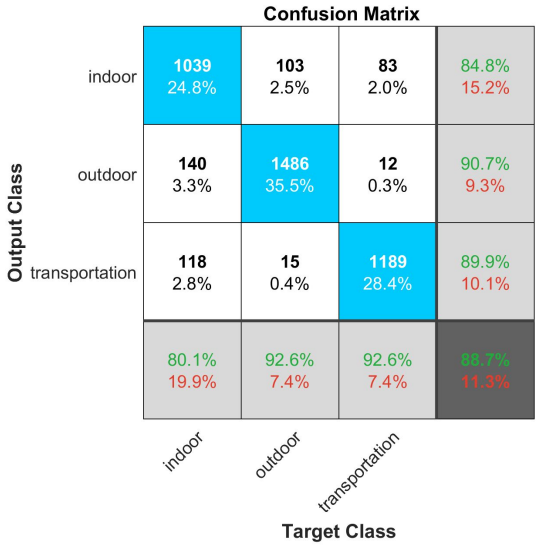

Figure 1: Confusion matrix for Model 1

5.2. Model 2: Combined Model 1 and Model 3

Complexity: 26.3 KB

Macro Average Accuracy: 88.5%

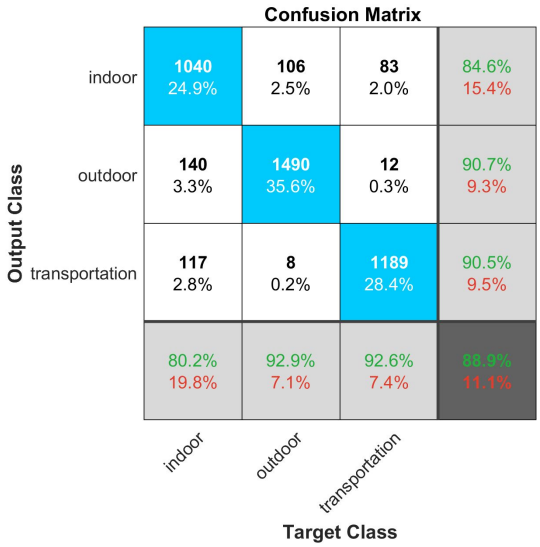

Figure 2: Confusion matrix for Model 2

5.3. Model 3: (L+R) mono only input channel

Complexity: 8.8 KB

Macro Average Accuracy: 86.5%

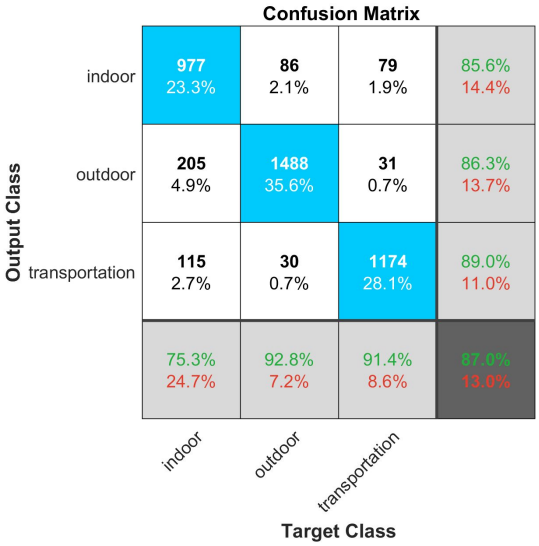

Figure 3: Confusion matrix for Model 3

6. REFERENCES

[1] https://en.wikipedia.org/wiki/Support_vector_machine
[2]https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html