Appendix 3: Experimental framework

In this appendix we describe the most important details of the experimental framework used along the *Xception1d* development and the design process followed.

1. Bibliography review: we found several works proposing different solutions to the *Google Speech-Commands* problem. *Andrade et al.* [1] proposed using *convolutional recurrent neural networks* with *self-attention*, over the *Mel-spectrograms* generated from the raw audio signals. *Zhang et al.* [2] used a bunch of different architectures (*MLPs*, *CNNs*, various *RNNs*, *Convolutional RNNs* and *depthwise separable CNNs*) focusing on building a lightweight model for implementing it into a microcontroller. They trained their models over a set of features extracted step using *Log-Mel Filter Bank Energies* and *Mel-frequency Cepstral Coefficients*. *McMahan and Rao* [3] explore *dilated convolutions* and *transfer learning*, applied to the same problem; we reported their best results (with or without *transfer learning*). Finally, *Warden* [4] uses *convolutional neural networks* over a set of *log-Mel filterbank* features to solve the same problem. All the works reported (to the best of our knowledge) propose shifting the data from the temporal to the frequency domain before running the model.
2. Hypotheses: after the literature review, the following hypotheses were defined
   - A neural network can achieve a competitive performance while operating in the temporal domain.
   - High performing architectures in computer vision should also perform well with audio signals.
   - A high performing model should mean being competitive with the human performance (in this case).
3. Benchmark consolidation: there are two versions of the *Google Speech-Commands* data set. Different studies [1, 4] provide results on the two versions while others [3, 2] only focus on V1 and V2 respectively. Besides, each of the previous references define different subtasks (consisting of simplifying the problem by reducing the subset of classes to predict; more details in the manuscript). We have run our model against all the combinations of data set version and task, in order to enable the comparison between all the existing benchmarks.
4. Cross-validation: with the aim of facilitating results reproduction and fair benchmarking, we used the train-dev-test splits suggested by the authors of the data set [5, 4] (containing about 80%-10%-10% of the data respectively).
5. Architecture design iterations: to get our final architecture, we have iterated across a set of experiments. Below, a brief summary of these steps is included[1] (in chronological order).
   - Following the self-imposed constraint of operating at temporal domain, we quickly discarded the possibility of implementing an RNN based architecture over the raw signal, due to their known issues on finding long-term dependencies, and their inability to parallelize across samples. Attention models were also discarded due to its quadratic computational complexity with respect to the number of time steps in the input signal.
   - The first architecture was designed mimicking the original *Xception architecture* [6], finding that the network didn't generalize well.
   - We tweaked the momentum hyperparameters of the *batch-normalization* operations until we achieved good performance on the validation set.
   - We realized that although the network performed well out of sample, the training accuracy quickly reached 100% in few epochs, despite the use of regularization techniques like *dropout*. This motivated us to implement a data-augmentation pipeline which solved the problem

[1] We don't have record of the accuracy contribution of the different components changed at each iteration

- We decided to switch to a normalization method that didn't depend on the instances dimension, to remove the dependence on the *momentum* hyperparameters. We tried *layer normalization* and *instance normalization* in different combinations, concluding that the best setting was to use *instance normalization* for the *convolutional layers* and *layer normalization* for the *fully-connected* ones (in the head).
- To improve further the results, we decided to implemented a method to decrease the *learning rate* every time a *plateau* was reached.

6. Repeated measurements: we run every experiment 5 times with different random seeds (40 runs; 4 tasks $\times$ 2 data versions $\times$ 5 random seeds) in order to evaluate the impact of different *random initializations*.
7. We measured the human accuracy level by using 4 subjects to manually annotate 1000 random recordings per subject. We run a *Student's T* test to see in which cases the accuracy of the model was significantly different than the human accuracy.
8. As a test experiment, we implemented a *Telegram* bot in order to check the ability of the model to generalize on fresh data. We concluded that the model was able to perform very well in these circumstances, although we didn't properly measure and report the results.

We run each of these trials in a single *GPU* environment (*Nvidia Titan Pascal*). We trained the models using *PyTorch* and we used *Tensorboard* (with *TensorboardX*) to analyze and track the experiments. Each experiment lasted around 1-3 days, depending on the task being solved.

References

[1] D. Coimbra de Andrade, S. Leo, M. Loesener Da Silva Viana, C. Bernkopf, A neural attention model for speech command recognition, Computing Research Repository CoRR (August 2018). arXiv:1808.08929.

[2] Y. Zhang, N. Suda, L. Lai, V. Chandra, Hello edge: Keyword spotting on microcontrollers, Computing Research Repository CoRR abs/1711.07128 (November 2017). arXiv:1711.07128.

[3] B. McMahan, D. Rao, Listening to the world improves speech command recognition, Computing Research Repository CoRR abs/1710.08377 (October 2017). arXiv:1710.08377.

[4] P. Warden, Speech commands: A dataset for limited-vocabulary speech recognition, Computing Research Repository CoRR abs/1804.03209 (April 2018). arXiv:1804.03209.

[5] P. Warden, Speech commands: A public dataset for single-word speech recognition., Datasets available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz and http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz (August 2017).

[6] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807. doi:10.1109/CVPR.2017.195.