Speech Commands Recognition
Machine Learning final project

Caria Natascia 1225874
Cozzolino Claudia 1227998
Petrella Alfredo 1206627

1. Introduction

The development of accurate and efficient "speaker independent" methodologies that enable the recognition and translation of spoken language is a crucial task in Artificial Intelligence. Speech recognition is the interdisciplinary subfield of computer science and computational linguistics that study this problem and proposes techniques for several applications such as virtual assistant, smart domotic control or dictation software. In this project *TensorFlow Google Speech Commands* Dataset [5] has been used to test the performances of different classifiers, providing a comparison between generalised linear methods, Random Forest and Deep Learning methods for recognising audio sequences speech commands. In addition several data augmentation strategies have been proposed to improve and stabilise generalisation results. As expect Convolutional and Recurrent Neural Networks will outperform less complex methods, achieving a test accuracy greater than 95%, but the beneficial effect of data augmentation also makes SVM competitive with more than 80% test score.

2. Dataset

The employed audio Dataset has been designed by TensorFlow to help train and evaluate keyword spotting systems. It has more than 65,000 one-second long utterances of 30 short words, by thousands of different people. In particular, in this study just a small portion of 1845 instances has been exploited, considering only the following 8 basic terms: *down*, *go*, *left*, *off*, *on*, *right*, *stop*, *up*. The dataset is already split into train (1600), validation (109) and test (136) sets. Strictly speaking each audio recording is a matrix of shape 128×32 corresponding to the frequency VS time log-Mel-spectrum representation [1]. The Mel-frequency cepstrum is a well known strategy for extracting robust independent features from raw speech audio mimicking the human auditory system processing, which takes into account sound frequency and the amplitude. Moreover a logarithmic transformation is usually applied to improve computational stability. Finally here, the "image" representation is compressed into a 32×32 matrix and then arbitrarily flatten as a 1024 elements vector according to the employed method.

3. Methods

To solve the multiclass problem of speech commands classification, several models have been built and tested. In this section they are all presented in detail.

3.1. One VS Rest Binary Classifiers

The first class of investigated techniques is the one of binary classifiers. Since the problem is about multilabels predicting, they have been modified in order to perform One VS Rest strategy. In this way a binary classifier per class is trained using its samples as positive and all the other as negative class. Practically speaking, `sklearn` functions are called for the following methods:

- Logistic Regression (LR): the supervised learning classification algorithm that predicts $\forall$ sample x the corresponding label according to the value of the probability of the target computed as $h_\theta(x) = g(\theta^T x)$, where g is the Sigmoid function and θ is the vector of parameters to be estimated. Note that the Python class automatically performs the regularized version.
- Support Vector Machine (SVM): the supervised learning algorithm that performs classification generating the hyperplane in multidimensional space that best separates the two classes, i.e. that finds the parameters θ such that the hyperplane $\theta^T x = 0$ maximizes the minimum margin between all the point instances. In particular the ν-Support Vector Classifier, which has an additional parameter to control the number of support vectors, has been used on the normalized dataset exploiting a Radial Basis Function (RBF) Kernel.

3.2. Random Forest

The second type of models that is proposed is trees ensemble methods. Since single decision tree is too weak for this complex application, ensemble strategies like Random Forest (RF) are preferred. In detail, RF is a meta estimator that fits a number of decision tree classifiers on various sub-samples or on the full dataset. For each sample the prediction is cooperatively performed by all the trees selecting the best solution by means of voting or averaging, thus improving the predictive accuracy and controlling over-fitting. Also in this case the model has been implemented using the built in class of `sklearn`, in particular the entropy criterion to measure the quality of a split and an integer value to limit the maximum depth have been adopted.

3.3. Neural Networks

The third section is dedicated to the neural networks, for which the main tools have been `tensorflow` and `keras` to build-up the models.

It has been chosen to start with a generic deep neural network, moving then towards a convolutional framework and then concluding integrating recurrent layers on the basis of theoretical knowledge. Below, better performing nets only are described in depth.

3.3.1 CNN

Taking into account a trade-off between performance and training time, the best convolutional neural network (CNN) that has been found via a trial and error approach has the following architecture. After the input layer, the net is composed of three convolutional layers with increasing number of filters, same kernel size, padding to keep the original dimension. Each of them is additonally followed by a max-pooling layer to scale the problem. The weights are randomly drown by the Xavier uniform initializer [2]. Regarding the activation functions, it has been chosen the Rectified Linear Unit (ReLU) for the hidden layers, trying to encourage the sparsity of the network and to avoid a gradient vanishing problem, which would have been a risk with other activations given the depth of the model. For the output layer it was quite natural to choose the Softmax, ideal for multiclass classification problems. The network had been fed first with flattened input features, but, as expected, the performances were not impressive, due to the lack of structure of the data. For this reason, the net has been adapted to accept 32$\times$32 input feature matrices, giving better results.

3.3.2 CRNN

The next remarkable model is a combination of convolutional and recurrent neural nets, which should adapt the best to the sequential nature of the features.

In this case, as for the previous methods, both the one-dimension and the bidimensional version of the model have been evaluated. Moreover, both Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) layers have been considered, obtaining a better resulting performance with the latter.

In detail, the network is composed by three convolutional layers, in the same fashion as the previous models, and an additional GRU layer before the dense output one. As usual, ReLU turned out to be the best option for the hidden layers, while Softmax is used for the output. Consistently with the previous attempts, the two-dimentions network better describes the data structure.

4. Experiments and Results

With regard to LR, SVM and RF, the hyper-parameters discussed above are chosen with an heuristic approach, while for neural networks it has been decided to proceed more rigorously by selecting the hyper-parameters with a grid search procedure with 5-fold cross validation (CV) on the provided train set, keeping the classes balanced in the validation fold. In particular, the grid search has been performed for all NNs on the following `keras` parameters:

- *filters*: $[32, 64]$
- *kernel_size*: $[3, 5]$
- *pool_size*: $[2, 3]$
- GRU *n_units*: $[32, 64]$ [1]
- *batch_size*: $[128, 256]$
- *epochs*: $[30]$

Given the execution time required for the grid search, to speed up the results the code has been executed on *CloudVeneto*, a cloud infrastructure at UNIPD. Table 1 shows the average and standard deviation results of the 5-fold CV for each method used, while Table 2 shows the values obtained from a single prediction on the originally provided sets, further improved using early stopping to avoid overfitting. The results are reported in terms of accuracy since data are well balanced in each set. A large gap can be noticed between the CV results and the results on the original validation and test sets. This suggests that the sets provided are probably "simpler" than the given train set. However, one should take into account that cross validating reduces the train set size by 20%. For this reason, and also as an attempt to improve the initial models' generalisation ability, data augmentation is performed.

4.1. Data Augmentation

Data augmentation has been proposed in several works as a method to generate additional training data for Automatic Speech Recognition [4] inspired by the success in the vision domains. The method is simple and computationally cheap to apply, as it directly acts on the log-Mel-spectrogram as if it were an image. In this work, basic transformations of the audio recordings have been experimented, and then new instances have been built randomly combining the different techniques on each instance of the

[1]Note that this parameter is to be considered only in the context of CRNN.

original data, doubling the initial sample dimension. In detail the implemented transformations are the following:

- Time shift: it just shifts of a random number of milliseconds the audio to left or right accordingly adding silence phase in front or at the end.
- Time masking: it completely mutes the audio signal at some randomly chosen instants of time.
- Frequency masking: it completely mutes the audio signal at some randomly chosen frequencies.
- White noise: it adds to the original audio data matrix random noise sampled from the normal standard distribution and scaled by an amplitude factor.

An example of the original speech command and its transformation is shown in Figure 1, 2. A final remark to conclude the presentation of the exploited data augmentation strategy is that all the parameters involved in the functions or in their random sampling processes have been manually tested and chosen inverting the obtained Mel representation back to the reproducible audio version (via the `librosa` Python library [3]). This step has been performed in order to guarantee that the produced artificial speech commands instances remain entirely understandable to a human listener.

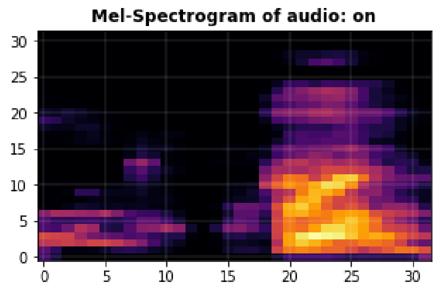

Figure 1. Example of Mel-frequency spectrogram of original compressed audio corresponding to the word ”on”. The colouring scale represents the sound intensity at each frequency and time instant, where black can be considered as silence.

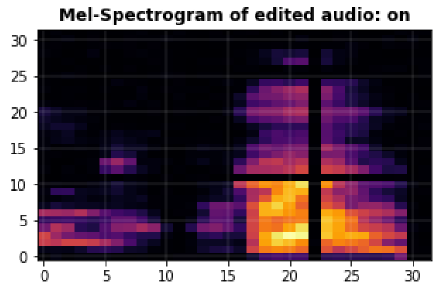

Figure 2. Mel-frequency spectrogram of the modified compressed audio previously presented. Different transformation can be observed: left horizontal shifting of two slots of time, horizontal and vertical black bands respectively corresponding to time instant or frequency muting, spot colour shadows due to noise injection.

4.2. Results

The overall results are reported in the tables below. Comparing the scores obtained on the original dataset and those obtained after data augmentation in Table1, it can be observed that an improvement in the average accuracy occurs for all the methods; the exception is LR which does not converge, probably for the large amount of data to fit. In general, the neural networks outperform the other machine learning algorithms, as one might expect, since the Mel-spectrogram representation of the speech commands is suitable for CNNs. Combining convolutional with recurrent layers seems not to be effective on the original dataset, however, with the increase in the dimensionality of the train set the CRNN model seems to be competitive with the CNN one. One would have expected CRNN to have better results than CNN, with the idea that the recurring layer might be able to take into account the temporal relationships of the data. This probably does not happen either because they are short commands or because the underling sequential structure is lost. Further research over the entire dataset could reveal undetected inter-model performance differences.

	Original		**Augmented**	
Model	Mean	SD	Mean	SD
LR	0.4664	0.0242	-	-
SVM	0.6922	0.0178	0.7253	0.0319
RF	0.5237	0.0163	0.6372	0.0142
CNN	0.8051	0.0286	0.8381	0.0149
CRNN	0.7753	0.0095	0.8284	0.0121

Table 1. CV best model accuracy statistics.

Model	Train	Valid.	Test
LR	1.0000	0.6239	0.4926
SVM	0.9094	0.8440	0.7941
RF	0.9988	0.7064	0.6471
CNN	0.9819	0.9617	0.9485
CRNN	0.9463	0.9266	0.9338

Table 2. Best models accuracy on the original data.

Model	Train	Valid.	Test
LR	-	-	-
SVM	0.8844	0.8716	0.8162
RF	0.9566	0.7172	0.7059
CNN	0.9803	0.9725	0.9632
CRNN	0.9812	0.9541	0.9559

Table 3. Best models accuracy on the augmented data.

References

[1] S.B. Davis and P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4), pp. 357–366*, 1980.

[2] Official Keras documentation. Xavier glorot's initializer. *https://keras.io/api/layers/initializers/*.

[3] McFee, Brian, C. Raffel, D. Liang, D.P.W. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. *Proceedings of the 14th python in science conference, pp. 18-25*, 2015.

[4] D.S. Park, Chan W., Zhang Y., Chiu C.C., Zoph B., Cubuk E.D., and Le Q.V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *ArXiv e-prints*, 2019.

[5] P. Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *ArXiv e-prints*, Apr. 2018.