DEEP LEARNING 2022

SPEECH COMMANDS CLASSIFICATION WITH RECURRENT NEURAL NETWORKS PROJECT NO. 2 PLAN

April 25, 2022

Elżbieta Jowik (298821), Agata Makarewicz (298827)

1 Datasets

The dataset used in this project is a *Speech Commands* [1] dataset - a set of 64,727 one-second `.wav` audio files, each containing a single spoken English word. These words are from a small set of commands and are spoken by a variety of different speakers. The audio files are organized into folders based on the word they contain. Alongside the data itself, there are two text files provided, containing the lists of filenames according to which the split into train, validation and test datasets should be conducted.

2 Data preparation

As specified in the title, the aim of this project is to perform a classification task based on the audio data. Considered task belongs to the class of multi-classification problems. The labels we will need to predict are `yes`, `no`, `up`, `down`, `left`, `right`, `on`, `off`, `stop`, `go`. There are other types of commands available in the datasets, however everything else should be considered either `unknown` or `silence`. Therefore, we need to first rename the classes according to the mentioned guidelines and then encode the target variable using the `One-Hot Encoding` method.
Mainly, we need to convert provided audio files into processable data, which we will do using the `scipy.signal` and `librosa` Python packages for audio analysis and information retrieval. Next, we need to ensure that all the recordings are exactly one second long - the identified shorter or longer recordings will be cropped or artificially prolonged.
Additionally, to help train the networks to cope with noisy environments, it can be helpful to mix in realistic or artificially generated sounds to simulate background noise. There is a collection of such sounds provided within the dataset - the `_background_noise_` folder contains a set of longer audio clips that are either recordings or mathematical simulations of noise. First of all, these records will need to be divided into sets of one-second-long recordings to match the length of the rest of the data. Then, the sounds will be randomly added to the training dataset in order to improve the learning process. Another approach that will be tested regarding the mentioned subset of data will be treating it as an individual class - `silence`. As the `_background_noise_` recordings are not present on the aforementioned lists specifying the division into the train, validation, and test dataset, their split will be performed manually, preserving the proportion of data.

3 Network architectures

We are going to test and compare 3 different network architectures (depicted in the simplified diagrams below) implementing the following concepts:

- `Conv1D`: a convolution layer used to extract features from the data.
- `Dropout`: applies Dropout to the input.
- `Dense`: a fully connected layer i.e. all the neurons in the current layer are connected to the next layer.
- `Activation`: a layer responsible for the application of an activation function.
- `BatchNormalization`: a layer that normalizes its inputs. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

[1] Warden P. Speech Commands: A public dataset for single-word speech recognition, 2017. Available from `http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz` .

- `SimpleRNN`: the recurrent layer object in Keras; fully-connected RNN where the output is to be fed back to the input.
- `LSTM`: Long Short-Term Memory layer, which is a type of recurrent neural network capable of learning order dependence in sequence prediction problems.
- `GRU`: Applies a multi-layer gated recurrent unit (GRU) RNN to an input sequence. A GRU layer learns dependencies between time steps in time series and sequence data.

Model concept no. 1	Model concept no. 2	Model concept no. 3
Layer type	Layer type	Layer type
Conv1D	Conv1D	Conv1D
BatchNormalization	BatchNormalization	BatchNormalization
Activation	Activation	Activation
Dropout	Dropout	Dropout
SimpleRNN	LSTM	GRU
SimpleRNN	LSTM	GRU
Dense	Dense	Dense
Dropout	Dropout	Dropout
Dense	Dense	Dense

4 Learning process parametrization & evaluation

As an optimizer, we plan to use the ADAM method which is currently perceived as the most powerful one. We estimate that the learning process will require up to 50 epochs. However, to reduce the number of unnecessary steps (if needed) early stopping will be applied. We are to investigate the influence of parameter change on the obtained results. The grids of hyperparameters considered like `batch size` or `learning rate` will be determined empirically through conducted experiments and based on available existing sources. Models' performance will be evaluated on metrics resistant to the problem of class imbalance, such as F1-score, confusion matrices heatmaps and visualizations illustrating the course of the loss function in the learning process.