Project 3
Methods of Deep Learning

Michał Bortkiewicz
Przemysław Kaleta
Jeremiasz Wołosiuk

May 2020

Contents

1 Description of the research problem

Third project of Methods of Deep Learning course at MINI PW aims to familiarize students with speech recognition problem and neural recurrent networks architectures including LSTM cells.
During research students have to develop deep recurrent neural network that will classify word on one-second long voice commands. It is permitted to use already trained weights and neural models and it is advisable to come up with new ideas and architectures using techniques as models concatenation, data augmentation, ensembles or voting models. One of the crucial aspects of research is to implement high performance model that will score high in kaggle competition.

2 Purpose of research

As a baseline, we will train and evaluate simple recurrent neural network and see how it will perform. It is possible, that such architecture will significantly underfit the data. Thus, we will look into architectures consisting LSTM cells and see if we can improve our classifier by using them. It will require adjusting hyperparameters of the network such as number of cells per layer, number of layers, batch size etc. We will try to understand which of them are crucial for the performance of the network to help us better apply them in other problems as well.

3 Planned techniques

Data we work with is speech. That means, that it is represented as a sequence of amplitude values sampled multiple times per second. Because of that, the input we have is often of different size (a sequence can have different lengths). This means, that the models we use have to take that into account. One common family of models that do that are Recurrent Neural Networks (RNNs) described in the simplest version back in 1986 in [1]. Several improvements were made on top of them, that dealt for example with technical problems of gradient vanishing and difficulty in capturing long term dependencies. Probably the most popular nowadays is Long Short Term Memory Network (LSTM, see [2]) or Gated Recurrent Units Network.

LSTMs work well with text modelled as sequence of words. In the case of audio however, it may turn out that the sequences are too long, and it is difficult to model such long dependencies. In this case, it is possible to use Convolutional Neural Networks first to reduce the size of an input before feeding it to recurrent layers. It may require padding the sequences first or other method to deal with variable sequence length.

This problem if a classification problem. It is contrary to general speech recognition systems (used for example in YouTube), which transcript long speeches into text ([3]). Such systems use sequence to sequence architectures (see [4]), often consisting of encoder and a decoder. Encoder and decoder are in this scenario recurrent neural networks (usually LSTMs or GRUs). Even though our problem is different, this techniques are still relevant: we may want to use encoder architecture and build a classifier on top of it. We believe, that if such architecture is useful for general speech recognition, it will be useful here as well.

One of the sequence to sequence problems is Machine Translation (statistical translation systems used for example in Google Translate). One of the most successful ideas there, which turned out to be fundamental building block in machine translation and Natural Language Processing in general was *attention* (see [5]). It can be applicable to our problem by using weighted combination of hidden states to produce classification output (in this way model can choose what part of an input to focus on). It was shown to greatly improve performance given enough data, but is computationally expensive. We believe it would be interesting to test this idea.

A strategy for this competition for us might be to:

1. Implement data preprocessing.
2. Quickly implement and train a simplest possible baseline model using LSTM network (or anything else).
3. Look for more sophisticated approach that was successful in this problem and is already implemented by someone else.
4. Test more complicated architectures and look for ways to improve on our own.
5. Create a model ensemble to boost performance.

3.1 Model ensemble

Machine learning technique intended to reduce learning bias, variance and overfitting problem by combining multiple models into one ensemble. There are multiple approaches to the problem, and few of them, which we are going to test, are:

1. majority voting - all models do their predictions and final prediction is the class with most voters,
2. average - models do their predictions, but now we average probabilities of individual classes for each model and then take class with the biggest average probability ,
3. weighted average/voting - same as above, but weighted by some value, for example accuracy of a model,
4. meta models,
5. bagging (**B**ootstrap **agg**regating).

4 Data description

Data for this project consists of audio files in .wav format with simple commands. They include:

- yes
- no
- up
- down
- left
- right
- on
- off
- stop
- go

Everything else is considered either *unknown* or *silence*. Last two labels are really important, because in speech recognition used in real world applications a lot of input is noise or just silence. For example, we wouldn't be happy if our Google Assistant turned on because a classifier someone have built kept classifying random noise as "Hey Google" command.

These samples are meant to be used for building a speech recognition classifier to recognise simple voice commands. Such system can be used in phone applications to recognise commands. They were recorded by people in various environments, possibly with some background noise.

Motivations behind creation of this dataset and all details are available in [6]. This dataset can serve as a benchmark for testing keyword spotting systems (systems that detect a selected set of phrases in speech).

5 Results verification methods

Since we will evaluate multiple neural network architectures, it is important to properly evaluate their performance. We will do it using standard procedure of doing cross-validation and splitting data into train, validation and test subsets. Train data will consist of 90% of observation, and will be used to train our models on it. Validation data (10 %) will be used for doing multiple evaluations of models and helping us choose best performing one. Because of the fact that we will perform multiple experiments, it is possible that we will get overly good performance on validation data just by chance. However, as we have learnt from experience on previous project, deep learning models we work with are computationally expensive and it is difficult to perform cross-validation which is a proper way to estimate performance of a model. Because of that it is likely that we will rely on single runs of experiments and use Kaggle for final evaluation of promising models.

The metric used for this Kaggle competition is accuracy and that's what will be used for final evaluation. The best one one achieved in that competition two years ago was 91.06%.

6 Experiments

As a preprocessing step, we merged all labels that we are not trying to predict and treated them as 'unknown'. For 'silence' label, we extracted samples from background noise clips. Additionally, in order to reduce the dimensionality of the input, we resampled all clips to 8000Hz. We also added 10% of background noise to each sample as a data augmentation strategy.

6.1 Baseline models

During research phase of the project we started with the convolutional model implemented in Keras based on notebook from one of the Kaggle competitors *CheonSeong Kang*. We used his methods for class labeling and signal prepossessing. For the purpose of quicker model convergence we removed some of the layers from the model proposed in CheonSeong Kang notebook. To see if LSTM layers will somehow improve accuracy of the model we modified baseline model and incorporated 2 LSTM layers after Conv1D layers.

	training Accuracy	*validation Accuracy*
convolution only	76.05%	79.24%
convolution + LSTM	91.13%	88.05%

Table 1: Basic models effectiveness with simple data augmentation.

Although presented metrics on train and validation sets seems really promising, it turned out that baseline models achieved only about 70% accuracy on test set (Kaggle).

6.2 Data representation - Spectrograms

For the purpose of more insightful data representation we transformed audio files, which were 1D vectors of audio amplitude signal to spectrograms. Spectograms is a 2D visual representation of the spectrum of frequencies of a signal as it varies with time. We derived basic knowledge about spectrograms from *DavidS notebook* and *Dalya Gartzman article* This form of input let us operate on audio data in similar way as with images and we are able perform convolutional operations to distill crucial features for classification based on this representation. There are two most popular kinds of spectrograms used in deep learning solutions: Spectrogram and Mel Spectrogram which is a better representation for DL algorithms because of nonlinear transformation of the frequency

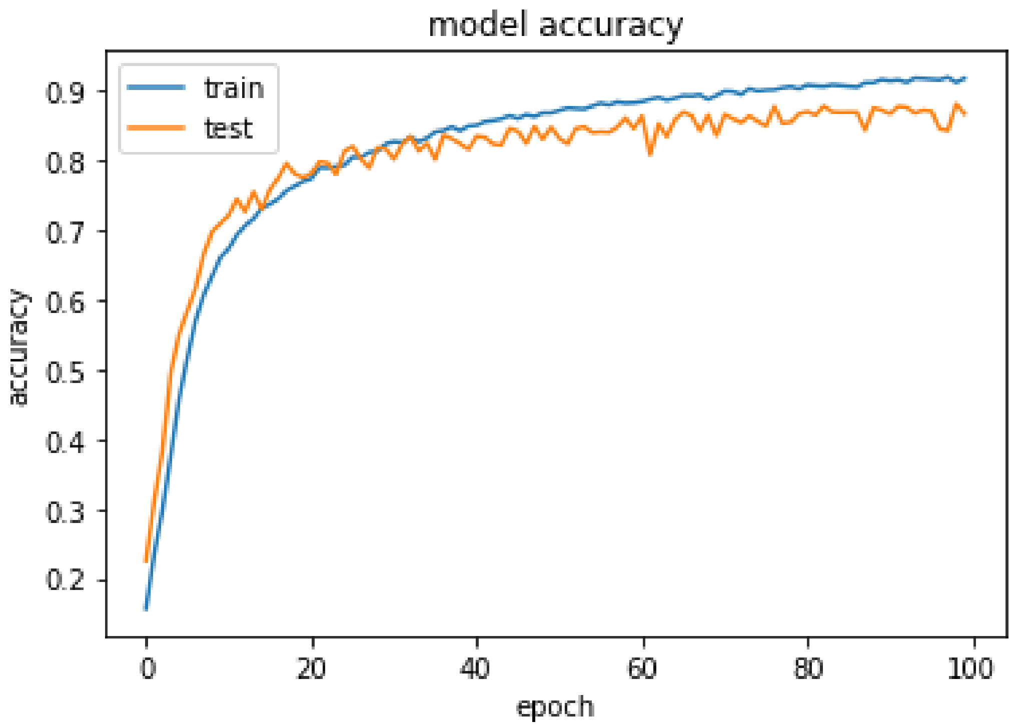

Figure 1: Model with convolutional and LSTM layers accuracy during training.

scale. Mel Scale is constructed such that sounds of equal distance from each other on the Mel Scale, also "sound" to humans as they are equal in distance from one another. We checked both of the representations and it indeed turned out that the second one was more informative for tested models

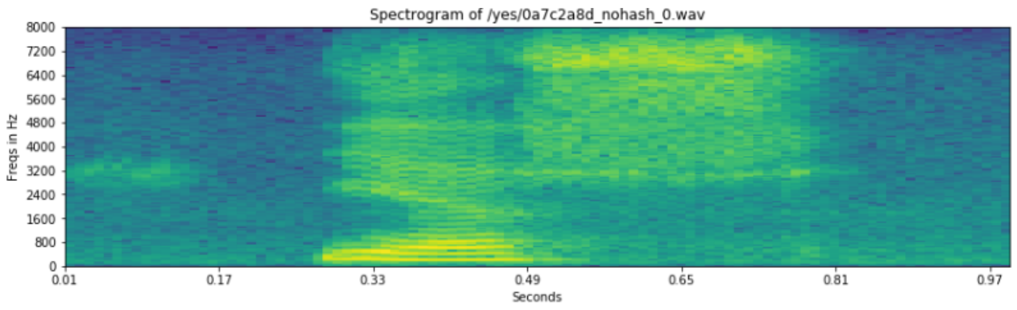

Figure 2: Standard spectrogram.

6.3 Data augmentation - random noise

Since we observed that the results on Kaggle are much worse (up to 10 percentage points) than our validation results, we realized the need for data augmentation. The goal is to make our classifier more robust by making it see more train examples. We achieved this by adding random level of randomly selected background noise form the data. We implemented it using tensorflow's tf.data.Dataset functionality and used it in each training example. After that, we observed significant increase in

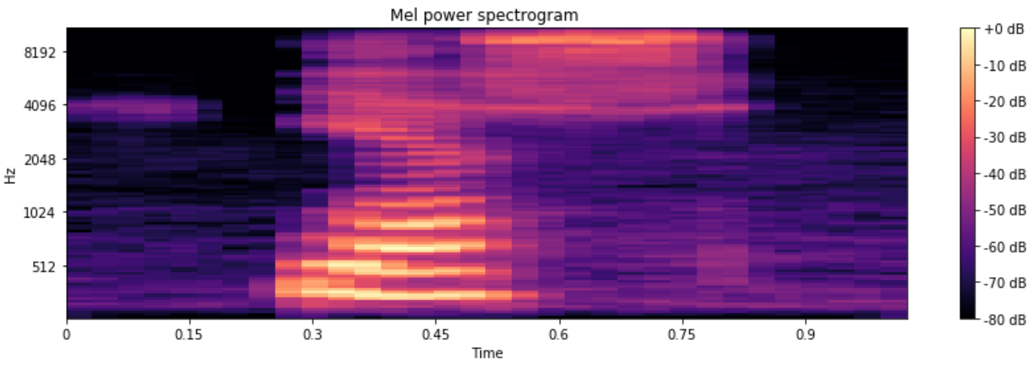

Figure 3: Mel spectrogram.

accuracy, from 75% without augmentation to 80% using the same architecture. Because of that, we decided to use data this data augmentation in our following experiments, even at the cost of slower data processing (for example single epoch took 3 minutes instead of 30 seconds).

6.4 Deep Speech model variations

After checking basic architectures we wanted to evaluate more complex state-of-the-art models from recent years. One of the most popular models for speech recognition are Deep Speech ([3]) and Deep Speech 2 ([7]) architectures. We have implemented variations of Deep Speech 2 model architectures based on *Rafał Rolczyński Automatic-Speech-Recognition* repository. Our implementations had two to three convolution layers at the beginning followed by k Bidirectional LSTM layers, one dense and softmax output layer. In experimentation phase we wanted to test whether deep or wide architectures cope better with posed problem.

6.4.1 Pyramid structure of LSTM layers

One of the evaluated variation of Deep Speech architecture was one with pyramid structure of LSTM layers. This kind of layers structure is widely popular for CNN and feature extraction, however, it turned out that it does not work as well for recurrent layers. Achieved accuracy was less than 80% with random noise augmentation on standard spectrograms.

```
...
x = layers.Bidirectional(layers.LSTM(128, return_sequences=True,
                                     dropout=drop_out_rate))(x)
x = layers.Bidirectional(layers.LSTM(64, return_sequences=True,
                                     dropout=drop_out_rate))(x)
x = layers.Bidirectional(layers.LSTM(32, return_sequences=True,
                                     dropout=drop_out_rate))(x)
x = layers.Bidirectional(layers.LSTM(16, return_sequences=True,
                                     dropout=drop_out_rate))(x)
...
```

6.4.2 Wide structure of LSTM layers

Wide in terms of cells per LSTM layer architecture proved to cope better with posed problem. It may be that is is only reasonable to scale vertically neural networks if dataset and problem is difficult enough. For this challenge with only 12 distinct classes shallow and wide architecture seems appropriate. We suppose that we need enough wide LSTM layers to capture complex relations of frequencies in time but no deeper than 2 to 3 layers, because of relatively easy task to classify small number of words.

```
...
x = layers.Bidirectional(layers.LSTM(512, return_sequences=True,
                                     dropout=drop_out_rate))(x)
x = layers.Bidirectional(layers.LSTM(512, return_sequences=True,
                                     dropout=drop_out_rate))(x)
...
```

6.4.3 Attention mechanism

Sequence neural models (like RNNs, including GRUs and LSTMs), produce a sequence of hidden states which serve as a summary of a sequence up to a given point in time. The final output is usually based on a last hidden state vector, which is supposed to be the best possible representation of the whole sequence. However, it is also possible to build models, that use an average of all hidden states in a sequence. Attention is more general, it allows us to take a weighted average average (with weights computed by a model) as a representation. Since this model is more general, we hoped that it would help improve performance of a classification. However, it turned out that we didn't obtain significant gains in performance after applying it.

6.5 ResNet

As mentioned before, since spectograms allow us to represent sound data as 2 dimensional objects (as opposed to 1 dimensional sequence of amplitude values), we can treat them kind of like images. Because of that, we can also use computer vision classification models in this task. As learned by experience in our previous project, one of the most successful architectures is ResNet.

It is an architecture described by Microsoft researches in [8], that made training huge networks easier by using residual connections . It is simply a layer outputting both the transformed vector as well as original one, that is $\mathcal{H}(x) = \mathcal{F}(x) + x$, where x in an input vector, $\mathcal{F}(x)$ is the layer's transformation and $\mathcal{H}(x)$ final (residual) transformation. In this way, we make it explicitly easier for the model to output previous vector representations if it decides that more transformations are useless. Combined with proper regularization, we can now train much deeper networks. This approach won 2015's ImageNet challenge.

We used it in our experiments and it achieved good result of 83% which we improved by using LSTM on top of it to 84%.

6.6 Ensembling

Since we trained multiple independent models with significantly different architectures and using different data formats, we decided to use ensembling to improve performance. We used single dense layer on top of outputs of all the models to predict final class.

In this way we achieved our highest score of 87.06% accuracy as shown in figure 4.

7 Conclusions

Speech recognition problem for many yeas had been occupied with hand-engineered domain knowledge, however, in recent years it has become much more approachable for deep learning practitioners because ene-to-end algorithms significantly outperformed previous solutions. It surprised us how easy it is to achieve reasonably high accuracy on task like this one. Our key conclusion from TensorFlow

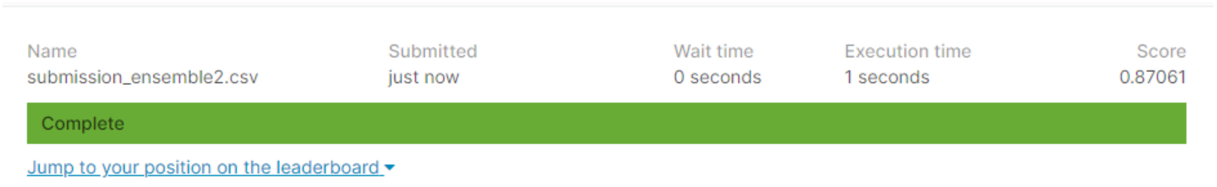
Name submission_ensemble2.csv
Submitted just now
Wait time 0 seconds
Execution time 1 seconds
Score 0.87061
Complete

Figure 4: Our final submission - ensemble of models.

Speech Recognition Challenge is coming up with new ideas for data representation. In this case representing audio with spectrograms was the most influential factor on final result. However, data augmentation turned out to be also crucial for making our models much more robust. It is also important to mention that for temporal problems like speech recognition recurrent neural networks should be the first choice for the solution. The can considerably leverage effectiveness of CNN by using better structured representation of data distilled by CNN. To conclude, we explored only basics of wide and deep fields which are speech recognition and recurrent neural networks, however, we gained fundamental knowledge in these areas.

8 Literature

- One of great books on the subject is ”Deep Learning” by Ian Goodfellow, Yoshua Bengio, and Aaron Courville ([9]).
- Recurrent neural networks (including LSTMs) were first described many years ago in [1] and [2].
- DeepSpeech ([3]) architecture was one of the first approaches to neural speech recognition. It showed, that deep learning can outperform traditional systems with carefully handcrafted features.
- DeepSpeech2 ([7]) is an improvement of the previous architecture.
- Data collection process and proposition of evaluation metrics are described in [6].

References

[1] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[3] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[6] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018.

[7] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* The MIT Press, 2016.