Detection of Typical Pronunciation Errors in Non-native English Speech Using Convolutional Recurrent Neural Networks

Aleksandr Diment, Eemi Fagerlund,
Adrian Benfield, Tuomas Virtanen
Tampere University
Tampere, Finland
firstname.lastname@tuni.fi

***Abstract*—A machine learning method for the automatic detection of pronunciation errors made by non-native speakers of English is proposed. It consists of training word-specific binary classifiers on a collected dataset of isolated words with possible pronunciation errors, typical for Finnish native speakers. The classifiers predict whether the typical error is present in the given word utterance. They operate on sequences of acoustic features, extracted from consecutive frames of an audio recording of a word utterance. The proposed architecture includes a convolutional neural network, a recurrent neural network, or a combination of the two. The optimal topology and hyperparameters are obtained in a Bayesian optimisation setting using a tree-structured Parzen estimator. A dataset of 80 words uttered naturally by 120 speakers is collected. The performance of the proposed system, evaluated on a well-represented subset of the dataset, shows that it is capable of detecting pronunciation errors in most of the words (46/49) with high accuracy (mean accuracy gain over the zero rule 12.21 percent points).**

***Index Terms*—Computer-assisted language learning, pronunciation learning, computer-assisted pronunciation training CNN, GRU, CRNN.**

I. Introduction

Learning a foreign language has traditionally been associated with the need for face-to-face tutoring. Computer-assisted language learning (CALL), on the other hand, can be mobile, cost-efficient, and capable of providing individual tutoring using advances in machine learning. Flashcard-based vocabulary exercises are a common kind of CALL due to the established theory behind the optimisation of memorisation [1]. Other learning aspects have also been emerging in a CALL setting. In this work, we focus on computer-assisted pronunciation training (CAPT). Machine learning methods are crucial in this task, as they enable reaction to the correctness of the pronunciation, giving the learner individually tailored feedback and instructions for correcting the detected errors. CAPT is an active research area, and its applications have been shown to have a highly significant and measurable impact on language learning [2].

The following works on the subject have been published recently. In [3], the problem of particular pronunciation error detection in non-native Dutch speech has been addressed using decision trees and linear discriminant analysis (LDA). However, the study was limited to only three phonemes. In [4], an LDA-based system has been developed for quantity contrast between short and long vowels in Norwegian.

In [5], a long short-term memory (LSTM) based system has been used for predicting the so-called proficiency score, defined subjectively on the training set of non-native English speech. As features, the authors used the output of a SpeechRater [6]: a system developed using TOEFL Practice data to capture among others prosody information, phoneme content and speaker voice quality of the input speech. The features include average chunk length in words, articulation rate, duration of silences and long pauses, frequency of long pauses etc.

In [7], a method for pronunciation error detection using Hidden Markov models performing phoneme recognition has been proposed. In the training dataset, in the cases of errors, annotations have been provided indicating the true transcription of the utterances, with distortion-type errors represented by newly created phoneme symbols. In [8], an extended recognition network-based approach has been used to discover individual error patterns from a student's speech, as well as a classifier-based approach, consisting of a convolutional neural network (CNN) with two branches, one for audio and the other for textual input. The output indicates whether the provided audio and the transcription match. The method allows for efficient data generation by creating infinite pairs of mismatched utterances and transcriptions.

In this paper, we propose a method for detecting typical pronunciation errors made by native Finnish speakers when speaking English. The system does not rely on phoneme recognition and does not require detailed phoneme-level annotations, but is rather trained with binary annotations of expected common errors occurring in the given utterances. Since the kinds of typical errors can be defined for each word separately, tailored instructions for their correction become possible. We propose convolutional (CNN) and recurrent neural networks (RNN), as well as their combination (CRNN), for detecting typical pronunciation errors (Section II). We collect a dataset of real utterances from 120 speakers (Section III) and evaluate the proposed methodology on 49 words with various primary error types (Section IV), showing optimal topology and hyperparameters for each word and each error type.

II. Methodology

We formulate a supervised machine learning problem, where a system is trained with examples of errors typically made by native Finnish-speaking learners in a selected set of English words, as well as examples of utterances where there are no such errors. The obtained models are then used to detect the target errors in the utterances of new, previously unheard speakers.

A. Classification setup

The system is intended to be used in a language learning scenario, where the word, whose pronunciation correctness is to be assessed, is known in advance. Therefore, the classifiers can be built for each word separately, with certain typical error types defined for each word based on the collected data.

The proposed methodology consists of performing classification on the input sequence of feature vectors, described in more detail in Section II-B. Here, the sequence refers to a consecutive set of feature vectors obtained from the whole utterance of the word, which may or may not include a typical target error in a particular phoneme. The set of typical errors is defined by linguistic professionals with expertise in teaching English to Finnish students. The modelling of the errors in the non-segmented utterances of the words is performed with neural networks composed of convolutional layers and/or bidirectional gated recurrent units (for the details see Section II-C).

An alternative approach of isolating the phoneme in which the target error could occur prior to performing the classification was initially considered. Forced alignment could be employed in order to obtain the timestamps of the phonemes constituting a whole-word utterance. However, for the given problem of detecting erroneous pronunciation, the forced aligner, trained on mostly non-erroneous speech, may fail. Unreliable segmentation of the target phonemes is not suitable for use in an automatic system. Additionally, focusing only on the phoneme where the expected error could occur results in discarding information on co-articulation, potentially valuable for the classification task.

Instead, we treat the whole utterance of a word as the classification instance. Each of these instances is associated with the target value. We consider a binary classification scenario, where the target takes the value 1 if the typical error occurs in the utterance and 0 if it is absent. The whole system consists of independent classifiers, trained for each word separately.

A three-run Monte-Carlo cross-validation setup is used, with a training–validation–test split proportion of roughly 0.6–0.2–0.2. Since multiple word instances can be uttered by the same speaker in the dataset, the split is performed in terms of speaker IDs. This leads to a fair evaluation with no speaker being present in more than one subset. The splits are obtained by shuffling the user IDs multiple times until the resulting class balance in the subsets is similar (up to 10% imbalance mismatch was tolerated). The alternative, k-fold cross-validation setup, was deemed unrealistic due to the difficulty of class stratification when splitting the speakers into folds. The assignment of subsets to the utterances of all the words for all runs of cross-validation is performed in advance, so that all the hyperparameter search trials would naturally operate on identical data.

B. Features

Both Mel-frequency cepstral coefficients (MFCCs) and mel spectograms are supported in the proposed system. The original idea behind computing MFCCs in speech recognition is that it enables extraction of the information about the spectral envelope. Its importance is considered high as it allows the detection of formants, which characterise the speech content for speech recognition tasks [9], [10]. In this work, from 10 to 40 (an optimised hyperparameter) MFCCs are extracted from 40 to 120 mel bands computed from amplitude-normalised audio in frames of 46 ms with a 75% overlap. Features are then optionally (a hyperparameter) normalised to zero-mean, unit variance using scaling factors from the training subset. Additionally, in line with the developments in deep learning for speech analysis [11], mel spectrogram features (40 to 120 bins) are supported, with the same parameters for frame blocking. Whether to use MFCCs or mel spectrograms is also one of the hyperparameters of the system (see Section II-D).

The only data augmentation performed is random time shifting by zero padding of the input signals on both sides. All the signals were padded to be of two seconds duration as the maximum duration of an utterance in the collected dataset is 1.93 seconds (see Section III).

C. Neural network

The proposed architectures include either convolutional (CNN), recurrent layers (more specifically, gated recurrent units, GRU [12]) or their combination (a convolutional recurrent neural network, CRNN), with possible additional fully-connected (FC) layers. The translation-invariance properties of CNNs and the temporal modelling capabilities of GRUs allow for direct training on the whole utterance without the need of segmentation of the phonemes in which the target error is expected to occur. The importance of the GRUs is particularly apparent when taking into account the modelling of longer-term phenomena, such as co-articulation. We use the bidirectional [13] variant of the architecture (BiGRU), which allows using the data both from past and future frames of a sequence, thus increasing the amount of information available for the network at each time step. The CRNN architecture, consisting of a set of convolutional layers followed by the recurrent ones, aims at combining the benefits of both and has been successfully used for the tasks of speech [14], music [15] and environmental audio analysis [16].

The proposed architecture (Figure 1) consists of a variable number of CNN, BiGRU and FC layers, and a one-unit output layer, predicting the likelihood of a pronunciation error. The temporal dimension of the input is supported to be arbitrary (as long as it is constant within each minibatch), even though in our experiments the inputs were padded to be of the same length for simplicity. The recurrent layer activation is tanh and the output neuron activation is sigmoid.

An option of regularising the convolutional part of the network using dropout [17] is supported, with its rate being a hyperparameter. For the BiGRU layers, however, recurrent dropout [18] is not used as it is not supported in the cuDNN-

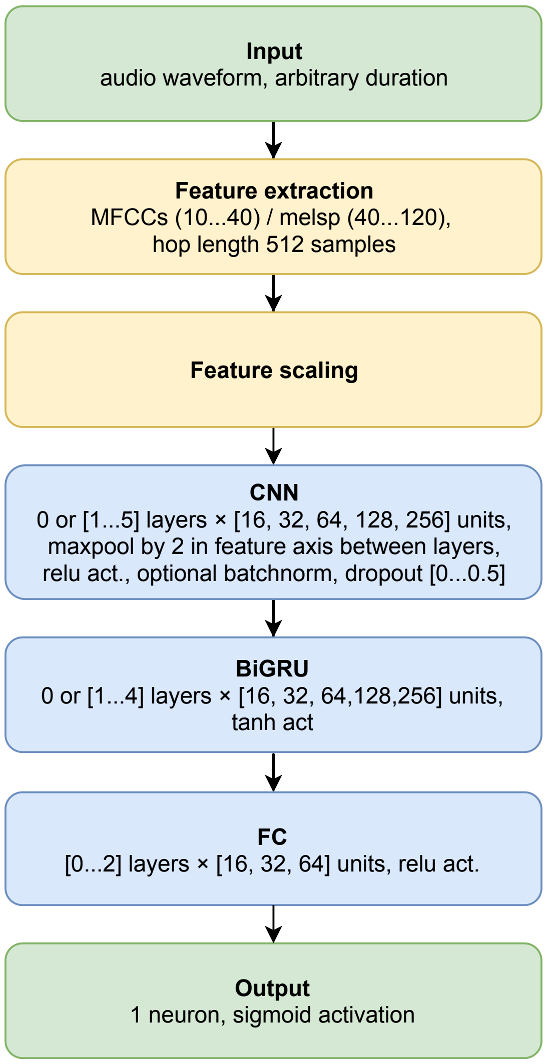
Input
audio waveform, arbitrary duration
Feature extraction
MFCCs (10...40) / melsp (40...120),
hop length 512 samples
Feature scaling
CNN
0 or [1...5] layers × [16, 32, 64, 128, 256] units,
maxpool by 2 in feature axis between layers,
relu act., optional batchnorm, dropout [0...0.5]
BiGRU
0 or [1...4] layers × [16, 32, 64,128,256] units,
tanh act
FC
[0...2] layers × [16, 32, 64] units, relu act.
Output
1 neuron, sigmoid activation

Fig. 1: The proposed system architecture and the hyperparameter search space.

based implementation, which was preferred due to its noticeable computational efficiency.

For the CNN layers, batch normalisation [19] is supported. A maxpooling operation is performed after each layer, reducing the feature dimension by half, while preserving the original time resolution. A global max pooling operation is then performed after the last convolution in the case of the CNN architecture. For the CRNN architecture, the feature dimension of the output of the last convolutional layer is maxpooled down to one, while the time dimension is left untouched and is represented by the time steps of the downstream BiGRU layers. In the case of the RNN architecture, the extracted features are fed directly to the BiGRU layers. The output of the last BiGRU layer at the last time step is then fed into the fully-connected layer afterwards.

Binary crossentropy loss is used as the training objective and is optimised with Adam [20], whose learning rate is also to be found in the hyperparameter search. The learning rate is reduced by half when the monitored validation loss does not decrease over the patience period of 50 epochs. The training is performed in up to 500 epochs with early stopping after 200 epochs of no improvement of the validation loss. The models are saved based on the best achieved validation loss.

D. *Hyperparameter optimisation*

For each word, hyperparameter optimisation experiments are performed over the duration of 35 hours on a GPU (Tesla K40 and Tesla P100 cards were used), resulting in an average of 193 trials per word. The objective is to maximise the average balanced validation accuracy (the average of recall obtained on each class) over the runs of cross-validation. When the balanced validation accuracy within one run of cross-validation yields 50%, the remaining runs with that search space sample are terminated to reduce computational cost.

The following hyperparameters are optimised (when applicable): architecture type (CNN, RNN, CRNN); feature type (MFCCs or mel spectrograms); number of mel bands; number of coefficients; feature normalisation flag; number of hidden units in the convolutional, recurrent and fully-connected layers; number of those layers; dropout rate and batch normalisation flag in the convolutional layers; and the initial learning rate (0.0001, 0.001 or 0.01). The search space of the parameters of the architecture is defined in Figure 1. Sequential model-based optimisation (SMBO), is used in the hyperparameter search as it yields better performance on the test set and requires fewer trials than the random or grid search strategies [21]. In SMBO, the objective function is modelled with a surrogate probabilistic surface, constructed using the information from each evaluation, and the next hyperparameter candidates are selected to maximise the expected improvement. In this work, the surrogate model is created using the tree-structured Parzen estimator [22], suitable for high search space dimensions and low evaluation budgets.

III. Data collection

To evaluate the proposed method, a dataset was collected. It includes recordings of 120 speakers pronouncing 80 distinct words twice or thrice, as well as annotations of typical errors being present or absent in each utterance. The following sections describe the collection, annotation and post-processing of the dataset.

A. *Selection of the words*

Prior to the recording, three English teachers, one of whom is a native English speaker, agreed upon a list of words whose pronunciation tends to be problematic for Finnish natives. The initial intention was to cover most of the common error types. However, the likelihood of their occurrence could not be guaranteed in the dataset of real utterances to be collected.

B. *Recording process*

There were 120 speakers in total (65 females, 55 males), 110 of whom were native Finnish speakers. The rest were native English speakers. The speakers were university and high school students aged between 17 and 30.

Each speaker was instructed to read through three or occasionally two lists of 80 words, each list containing the same words but in a different order. The word prompts were shown on a tablet. After reading each word aloud, the speaker would tap the screen, prompting the software to show the next word. We had randomised the list of words into 10 different orders. This was done to mitigate any possible bias in pronunciation that might result from speakers always reading the words in the same order (e.g. tiredness). Each speaker read the words in two or three different orders, thus providing two to three utterances per word.

The recordings were made in an acoustically treated noise-insulated room with dimensions 4.53 m $\times$ 3.96 m $\times$ 2.59 m and a reverberation time of 0.26 s. A single Røde NT55 condenser microphone and a Focusrite Scarlett 2i2 audio interface were used. The speakers were standing approximately 40 cm away from the microphone while recording, . The parameters of the recordings were: sampling frequency 44.1 kHz, bit depth 16 and PCM encoding. The statistics for the duration of the collected utterances are: mean 1.14, standard deviation 0.18, maximum 1.94, minimum 0.78 seconds.

C. Annotation

The single-word utterances from the continuous recordings of individual speakers were manually segmented into separate files. They were then grouped into playlists where all the utterances of the same word were combined.

Two experts in linguistics performed the annotations. They first went briefly through the examples of each word to get the big picture and to define the primary and possible secondary error types. Next, they listened through the whole dataset, independently performing the annotations of all the utterances, marking whether they were correct, contained the specified primary or secondary error, or some other kind of error. Additionally, they had an option to mark the utterance as to be discarded due to some technical issue (e.g. noise or distortion). All the samples of a word were automatically played back in sequence with a four-second pause in between. However, the annotators had the option of listening multiple times to an utterance and to compare it with the other speakers' utterances of the same word.

D. Dataset organisation

The focus of this work was to detect the most common pronunciation errors. Detecting secondary errors with a machine learning system was deemed unfeasible due to the insufficient amount of data in which those occur (only 2.4% of the samples). Therefore, the annotations of the secondary errors were ignored in this work, and the system output only predicted the presence or absence of a typical error. Additional post-processing was performed by discarding samples over which the annotators had disagreed. Such cases were most likely due to the non-binary scale of the error and a degree of subjectivity of the problem. The classifier was only trained with the agreed samples whose certainty was thus maximised, in line with the binary label format. The possibility of training the network with uncertain data and advanced methods of combining contradictory annotations [23] are left for future investigation.

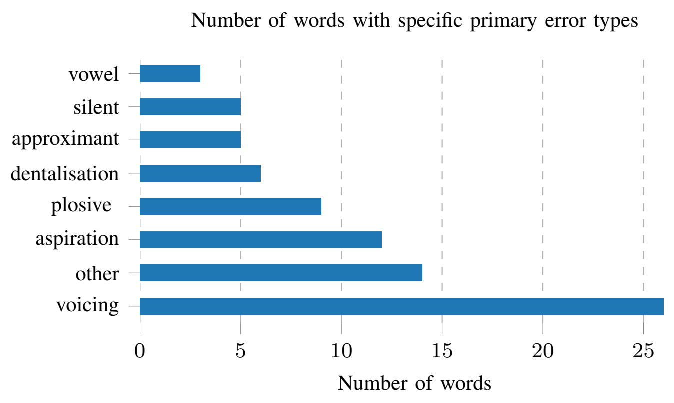

Fig. 2: Primary groups of errors discovered in the dataset and corresponding counts of unique words with such errors as primary ones. The error classes are defined as the follwing:

- *vowel*: a vowel error (e.g. vowel [ɪ] becoming [e] on the first syllable of the word *begin*),
- *dentalisation*: bilabial [w] becomes labio-dental [v] (words *worse*, *wet*, *wine*),
- *silent*: silent letters (b in *bomb*, p in *psychology*, l in *walk*, h in *hour*) being pronounced,
- *approximant*: phoneme [v] is rather an approximant than a fricative (words *very*, *vet*, *vine*, *voice*, *verse*),
- *plosive*: ð not dental enough (tongue should be between the teeth), pronounced like Finnish [t] (with slight aspiration) (words *through*, *than*, *thin*, *weather*),
- *aspiration*: missing aspiration in phonemes [p] (*push*, *prince*, *pull*, *pear*), [t] (*fantastic*, *tie*), [k] (*cave*, *control*, *card*, *coat*)
- *voicing*: voiced consonants becoming voiceless ([z] in *goes*, ([ð] in *bathe*, [dʒ] in *joke*, [d] in *bed*)
- *other*: various kinds of other errors, shared across many words in the dataset ("rolling r", affricate [tʃ] or fricative [ʃ] becomes like plosive [t] in the word *structure*, alveolar [s] instead of palate-alveolar [ʃ] in the words *education* and *fish*, affricate [tʃ] becoming more like combination [ts] in the word *rich*.

Finally, the counts of agreed correct and target error samples were computed for all the collected 80 words. As a rule, most of the words showed significant class imbalance, with the correct pronunciation occurring much more often than errors. This issue does raise challenges for the data-driven approach. For training and evaluation of the proposed system, a subset of 49 words was selected based on two criteria: at least 10 samples of either class needed to be present in all the training, validation and test sets for the three distinct splits, and a rough (10% tolerance) stratification should be achieved after 10 000 shuffling attempts at most. This way, we were able to discard the words for which we expected it to be unfeasible to develop machine learning methods due to the lack of available data.

The word-wise details on the target errors, the agreement rate between the annotators and the counts of samples belonging to either class are presented later in the summary Table II. For the aggregated statistics on the discovered primary error types in the collected dataset, see Figure 2. We see that the voicing error turned out to be the dominant one across the words, which could be explained by the scarcity of voiced consonants in the Finnish language.

IV. Evaluation

A. Metrics

Within each run of the search, in addition to computing the balanced accuracy on the validation set as the hyperparameter optimisation objective, evaluation metrics were computed on the test set and stored for the final analysis. Test set performance was kept apart from all the experimentation and had no effect on the decisions made by the experimenter. The following metrics were computed: F_1-score, area under a receiver operating characteristic curve (AUC), accuracy and zero rule. The latter, defined as the accuracy of always predicting the majority class, is useful for gaining a better insight into the meaning of the accuracy score in an unbalanced problem.

B. Overall results

The results are illustrated in Figure 3 and in more detail in Table II. We see that for the words with class priors closest to equal, i.e. with large numbers of samples belonging to both classes, the performance of the proposed system is most prominent (an average 17.84 percentage point accuracy gain over the zero rule for the 20 most represented words). For the words with highly unbalanced classes, the problem becomes more challenging. Still, the system beats the zero rule in 46 cases of all the 49 evaluated words and produces an average accuracy gain over the zero rule of 12.21 percentage points.

C. Hyperparameter-specific results

Interesting observations can be made about the results of the hyperparameter search. Figure 4 presents the counts of systems with certain values of hyperparameters, selected for each word as yielding the best validation accuracies. There is a certain preference of RNNs over CNNs and CRNNs. This can be explained by the fact that "classical" features were supported, and so the benefit of convolutional layers as a feature extractor was thus not as prominent. Temporal modelling with BiGRUs appeared to be beneficial for the majority of the words.

MFCCs were selected more often than mel spectrograms, contrary to recent trends in machine listening [11]. The dimensionality reduction properties of MFCCs may explain this. While the network should be able to learn from the higher-dimensional mel spectrograms, and the hyperparameter optimisation should be able to select the optimal number of features, the system may overfit more easily given the insufficient amount of data, thus rendering MFCCs a more appropriate feature choice.

Concerning the depth of the neural networks, all the architectures benefit from a larger depth of convolutional layers, while the number of RNN layers was preferred to be moderately large for purely RNN architectures. Batch normalisation in the CNN layers was useful in the majority of the cases. Moderate rates of dropout were preferred. In the case of CRNN architectures, half of the systems preferred not to perform any dropout at all.

Aggressively large learning rates during the initial stages were shown to be beneficial. Naturally, since the reduction of the learning rate on a validation loss plateau was always supported, the potential drawbacks of excessively large learning rates were controlled.

TABLE I: Error type-wise details of the preferred parameters and performance.

error type	preferred architecture	preferred features	avr. # epochs	avr. gain over ZR
approximant	RNN, 3/3 cases	MFCC, 2/3	52	16.93
aspiration	CRNN, 4/8	MFCC, 5/8	54	11.44
plosive	CNN, 3/7	MFCC, 4/7	60	11.16
other	CNN, 4/7	melsp, 4/7	61	12.39
silent	RNN, 3/4	MFCC, 4/4	88	10.80
voicing	CNN, 8/17	MFCC, 12/17	83	12.49
dentalisation	RNN, 2/2	MFCC, 2/2	74	17.88

D. Error-specific results

Next, we consider the performance and parameters of the systems in relation to the types of pronunciation errors (see Table I). The vowel error type, represented in the dataset by the word "begin" is excluded as the gain over zero rule could not be achieved with this word, rendering the hyperparameter values irrelevant. The most noticeable improvement over the zero rule was observed with [v] and [w] related errors, with other consonant errors also being detected with high accuracies.

Among the proposed architectures, RNNs were selected most often for the [v] and [w] related error types, as well as for silent phonemes. Aspiration-related errors were detected most often with a CRNN, and the rest of the error types found CNN to be sufficient. MFCCs were a clear primary choice of features except for the "other" error types. All the networks reached best validation scores very fast, in under 100 epochs.

E. Bayesian optimisation performance

The best validation accuracy, used for selecting the hyperparameters, was achieved on average after 60% of the trials (translating into an average trial number of 120). The mean mismatch between the validation and corresponding test accuracies of the selected systems was 6.73 percentage points. If knowing which hyperparameters would achieve the best test accuracy was possible (the so-called oracle case), this mismatch would have been 3.45 percent points. This suggests that the systems generalise well, the defined setup is reasonable, and further optimisation is unlikely to produce any noticeable improvement in performance.

V. Conclusion

Systems based on CNN, RNN and CRNN architectures were proposed for detecting common pronunciation errors in isolated word utterances. Optimal hyperparameters were obtained for each word and type of error. The performance was evaluated with a collected annotated dataset of utterances from 120 people. The system showed good performance: for the twenty words with the most available data for each class, it achieved an average 17.84 percentage point accuracy gain over the zero rule. Overall, it has beaten the zero rule in 46 out of the 49 evaluated words. The method is capable of solving the problem of pronunciation error detection without resorting to hand-crafted features or massive datasets (successful performance was observed with as few as 24 examples of target errors in the dataset).

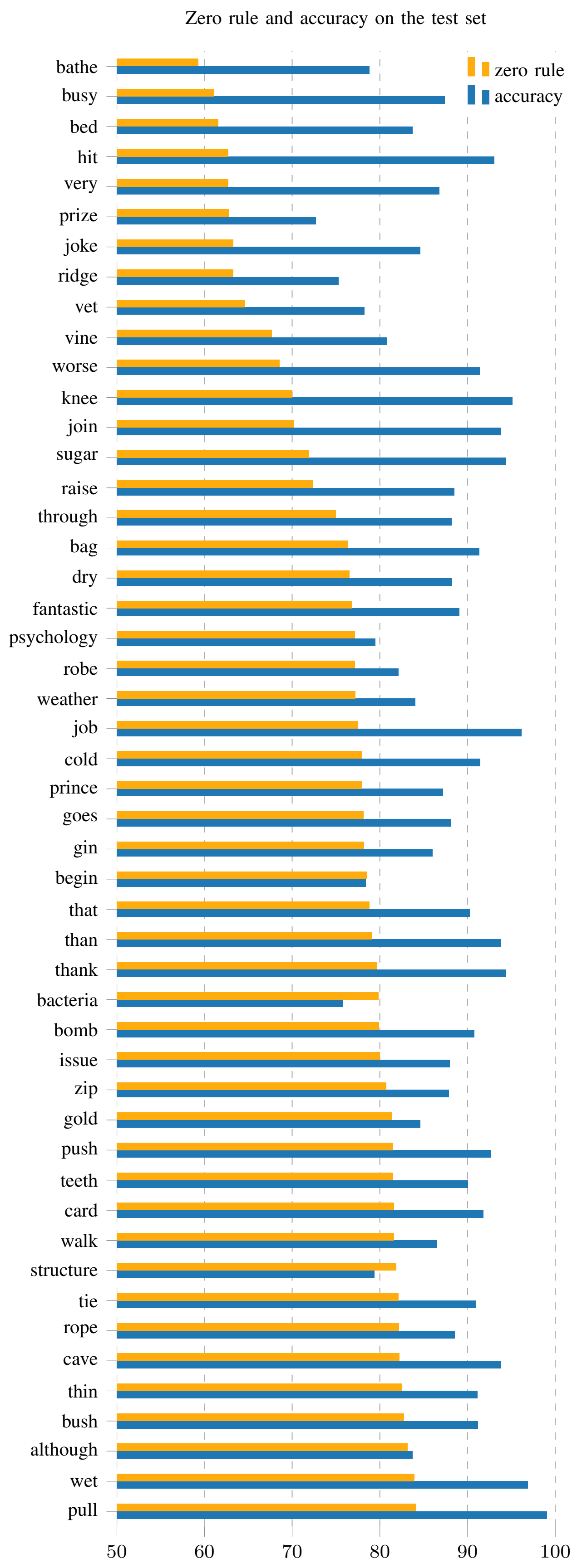

Fig. 3: Test set performance of the best architectures selected based on the validation scores. The words are in the increasing order of class imbalance.

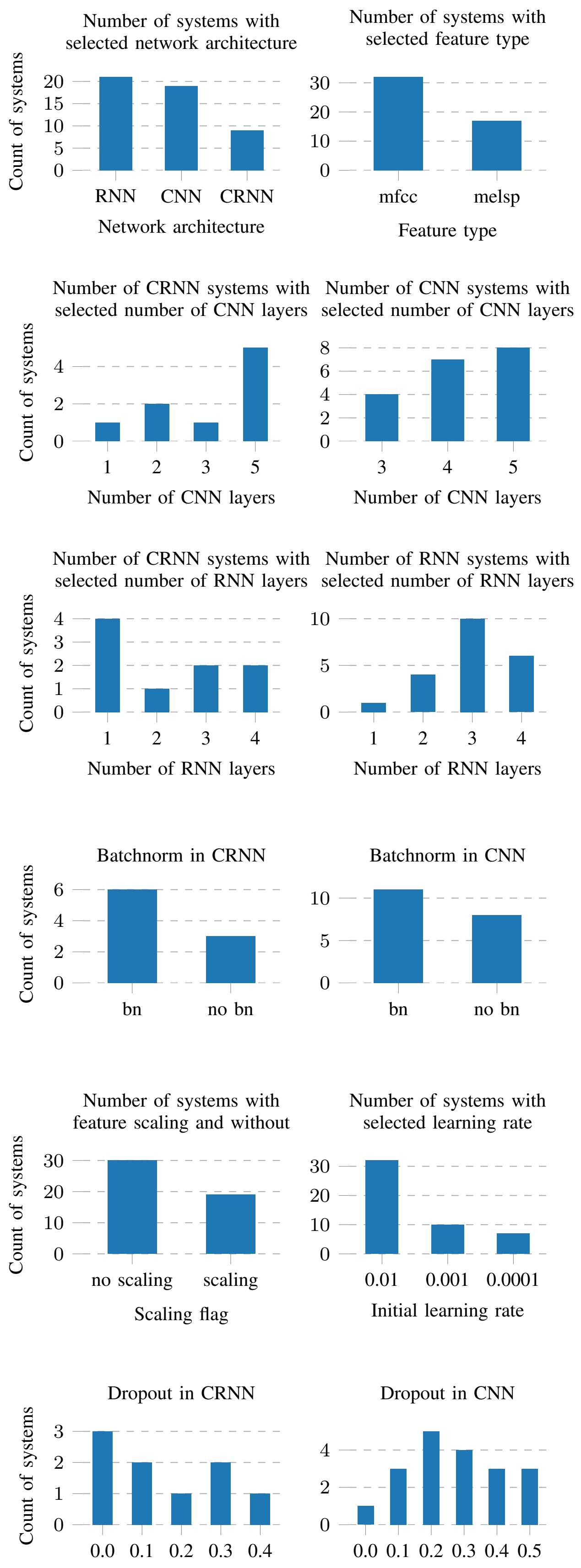

Fig. 4: Number of systems with corresponding hyperparameter values.

However, the systems were trained and evaluated in a somewhat idealistic scenario. The signals were noiseless and recorded with the same equipment, and the samples with contradictory annotations were not used. The future work directions include methods for noise, room and device robustness, as well as for learning from contradictory annotations. Additionally, joint training of multiple words as opposed to the proposed separate classifiers could confer the benefit of an increased amount of training data. Finally, transfer learning methods will be studied, e.g. by using models, pre-trained on a speech recognition task.

Acknowledgment

Computing resources were provided by CSC—IT Center for Science, Finland. Shriram Nandakumar's contribution to the data collection is acknowledged. We would also like to thank the authors of the software used in this work: keras [24], TensorFlow [25], hyperopt [26], scikit-learn [27], numpy [28], pandas [29], matplotlib [30], and matplotlib2tikz [31].

References

[1] R. Godwin-Jones, "Emerging technologies from memory palaces to spacing algorithms: approaches to second-language vocabulary learning," *Language, Learning & Technology*, vol. 14, no. 2, pp. 4–11, 2010.

[2] E. M. Golonka, A. R. Bowles, V. M. Frank, D. L. Richardson, and S. Freynik, "Technologies for foreign language learning: a review of technology types and their effectiveness," *Computer Assisted Language Learning*, vol. 27, no. 1, pp. 70–105, 2014. [Online]. Available: https://doi.org/10.1080/09588221.2012.700315

[3] K. Truong, A. Neri, C. Cucchiarini, and H. Strik, "Automatic pronunciation error detection: an acoustic-phonetic approach," in *InSTIL/ICALL Symposium*, 2004.

[4] I. Amdal, M. H. Johnsen, and E. Versvik, "Automatic evaluation of quantity contrast in non-native norwegian speech," in *International Workshop on Speech and Language Technology in Education*, 2009.

[5] Z. Yu, V. Ramanarayanan, D. Suendermann-Oeft, X. Wang, K. Zechner, L. Chen, J. Tao, A. Ivanou, and Y. Qian, "Using bidirectional LSTM recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec 2015, pp. 338–345.

[6] K. Zechner, D. Higgins, X. Xi, and D. M. Williamson, "Automatic scoring of non-native spontaneous speech in tests of spoken english," *Speech Communication*, vol. 51, no. 10, pp. 883–895, 2009.

[7] R. Ai, "Automatic pronunciation error detection and feedback generation for call applications," in *Learning and Collaboration Technologies*. Springer, 2015, pp. 175–186.

[8] A. Lee, "Language-independent methods for computer-assisted pronunciation training," Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[9] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 4, pp. 357–366, 1980.

[10] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, ser. Prentice Hall Signal Processing Series. PTR Prentice Hall, 1993. [Online]. Available: http://books.google.fi/books?id=XEVqQgAACAAJ

[11] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft." IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), May 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/recent-advances-in-deep-learning-for-speech-research-at-microsoft/

[12] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, 2014, pp. 103–111. [Online]. Available: http://aclweb.org/anthology/W14-4012

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[14] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.

[15] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2392–2396.

[16] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, June 2017.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv e-prints*, Dec. 2014.

[21] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *30th International Conference on Machine Learning (ICML 2013)*, 2013.

[22] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *NIPS*, 2011.

[23] F. Rodrigues and F. Pereira, "Deep learning from crowds," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018, pp. 1611–1618.

[24] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[26] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. I–115–I–123. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3042832

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing. http://numpy.org/, 2006–, [Online; accessed 2018-12-13].

[29] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[30] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[31] N. Schlömer, "matplotlib2tikz: a python tool for converting matplotlib figures into pgfplots (tikz) figures," https://github.com/nschloe/matplotlib2tikz, [Online; accessed: 2018-12-13].

TABLE II: Evaluation results and dataset statistics for the 49 evaluated words. Correct count and tar. er. count stand for the numbers of ground truth correct utterances and target error utterances in the agreed subset of the dataset. The test accuracies are in bold face when they exceed the zero rule.

word	phoneme	error type	annot. agr., %	correct count	tar. er. count	ZR, %	acc., %	AUC	F_1, %	# params	feature	archit.
although	[ð]	voicing	92.99	221	28	83.16	**83.73**	0.60	25.18	201.73K	MFCC	CRNN
bacteria	[b]	voicing	85.20	224	23	79.83	75.79	0.51	11.91	4.94M	MFCC	CNN
bag	[g]	other	89.56	215	33	76.39	**91.32**	0.85	74.17	26.42K	melsp	CNN
bathe	[ð]	voicing	76.17	79	124	59.29	**78.79**	0.79	77.60	29.31K	MFCC	RNN
bed	[d]	voicing	76.17	152	88	61.57	**83.73**	0.83	79.70	451.20K	melsp	RNN
begin	[ɪ]	vowel	89.41	244	40	78.47	78.42	0.65	45.74	106.18K	melsp	CNN
bomb	silent b	silent	92.83	228	64	79.91	**90.76**	0.88	78.74	30.21K	MFCC	RNN
bush	[b]	voicing	92.52	249	27	82.77	**91.20**	0.80	68.03	215.30K	MFCC	RNN
busy	[z]	voicing	80.53	139	115	61.04	**87.41**	0.87	88.51	317.50K	MFCC	CNN
card	[k]	aspiration	92.83	270	27	81.58	**91.78**	0.83	74.35	1.67M	melsp	CRNN
cave	[k]	aspiration	95.33	274	32	82.25	**93.84**	0.91	84.99	2.24M	melsp	CRNN
cold	[k]	aspiration	87.07	198	32	77.96	**91.42**	0.80	74.78	2.88M	melsp	RNN
dry	[d]	voicing	79.60	186	58	76.52	**88.25**	0.84	75.24	3.41M	MFCC	CRNN
fantastic	[t]	aspiration	85.05	230	42	76.80	**89.04**	0.83	70.75	1.27M	MFCC	CNN
gin	[dʒ]	voicing	86.14	208	58	78.18	**86.02**	0.85	72.43	2.82M	MFCC	RNN
goes	[z]	voicing	86.45	40	233	78.12	**88.15**	0.76	92.76	3.42M	MFCC	CNN
gold	[g]	voicing	91.28	237	25	81.35	**84.60**	0.74	53.87	700.67K	MFCC	RNN
hit	[ɪ]	other	67.76	147	70	62.68	**93.04**	0.93	90.53	1.62M	MFCC	RNN
issue	[ʃ]	other	93.15	240	29	79.98	**87.98**	0.73	57.68	437.76K	MFCC	CNN
job	[dʒ]	voicing	73.68	203	30	77.52	**96.14**	0.91	89.24	104.03K	melsp	CNN
join	[dʒ]	voicing	83.96	198	70	70.15	**93.76**	0.92	90.38	6.86M	melsp	CRNN
joke	[dʒ]	voicing	76.48	148	95	63.27	**84.63**	0.85	79.77	4.92M	MFCC	CNN
knee	silent k	silent	96.42	223	85	70.02	**95.13**	0.94	92.40	54.27K	MFCC	CNN
prince	[p]	aspiration	88.01	246	35	78.00	**87.18**	0.79	67.75	2.80M	MFCC	RNN
prize	[z]	voicing	83.64	142	119	62.84	**72.69**	0.76	70.60	1.68M	melsp	CNN
psychology	silent p	silent	69.94	133	39	77.16	**79.47**	0.82	66.60	3.98M	MFCC	RNN
pull	[p]	aspiration	97.82	279	35	84.12	**99.05**	0.98	97.52	439.87K	MFCC	CRNN
push	[p]	aspiration	94.70	234	56	81.51	**92.65**	0.86	79.32	414.91K	MFCC	CNN
raise	[z]	voicing	80.22	81	174	72.40	**88.47**	0.84	91.83	78.27K	MFCC	CNN
ridge	[dʒ]	voicing	60.59	92	87	63.29	**75.29**	0.72	67.86	80.39K	melsp	CNN
robe	[b]	voicing	87.69	218	50	77.17	**82.12**	0.70	51.42	184.10K	MFCC	RNN
rope	[r]	other	95.33	269	27	82.17	**88.52**	0.77	54.26	1.05M	melsp	CRNN
structure	[tʃ]	other	94.24	268	27	81.85	79.36	0.48	0.00	3.28M	melsp	CNN
sugar	[ʃ]	other	96.26	219	88	71.94	**94.36**	0.94	90.30	103.39K	melsp	CNN
teeth	[θ]	plosive	92.21	260	34	81.52	**90.02**	0.82	73.62	704.51K	MFCC	RNN
than	[ð]	plosive	86.76	168	43	79.04	**93.83**	0.90	84.90	1.65M	melsp	CNN
thank	[θ]	plosive	94.24	260	39	79.67	**94.40**	0.93	86.32	1.70M	melsp	CRNN
that	[ð]	plosive	88.01	197	38	78.79	**90.24**	0.79	70.37	724.61K	MFCC	RNN
thin	[θ]	plosive	94.39	256	39	82.52	**91.15**	0.81	72.56	3.29M	MFCC	CNN
through	[θ]	plosive	91.59	194	73	74.98	**88.18**	0.89	80.66	3.98M	MFCC	RNN
tie	[t]	aspiration	91.90	266	24	82.13	**90.91**	0.77	56.62	1.55M	MFCC	CRNN
very	[v]	approximant	55.30	103	55	62.72	**86.79**	0.87	84.14	3.11M	melsp	RNN
vet	[v]	approximant	54.21	73	52	64.64	**78.26**	0.77	71.12	4.05M	MFCC	RNN
vine	[v]	approximant	60.90	88	42	67.66	**80.77**	0.75	63.07	997.12K	MFCC	RNN
walk	silent l	silent	91.28	261	27	81.61	**86.54**	0.63	35.52	256.67K	MFCC	RNN
weather	[ð]	plosive	89.10	237	37	77.19	**84.02**	0.73	59.57	313.25K	melsp	CNN
wet	[w]	dentalisation	97.82	275	36	83.94	**96.86**	0.90	88.97	423.73K	MFCC	RNN
worse	[w]	dentalisation	89.88	213	67	68.56	**91.41**	0.89	85.65	2.86M	MFCC	RNN
zip	[ts]	other	92.21	242	30	80.74	**87.89**	0.76	63.77	278.91K	MFCC	RNN