Detecting the Intensity of Cognitive and Physical Load Using AdaBoost and Deep Rectifier Neural Networks

Gábor Gosztolya[1], *Tamás Grósz*[1], *Róbert Busa-Fekete*[1], *László Tóth*[1]

[1]MTA-SZTE Research Group on Artificial Intelligence, Szeged, Hungary

{ ggabor, groszt, busarobi, tothl } @inf.u-szeged.hu

Abstract

The Interspeech ComParE 2014 Challenge consists of two machine learning tasks, which have quite a small number of examples. Due to our good results in ComParE 2013, we considered AdaBoost a suitable machine learning meta-algorithm for these tasks, besides we also experimented with Deep Rectifier Neural Networks. These differ from traditional neural networks in that the former have several hidden layers, and use rectifier neurons as hidden units. With AdaBoost we achieved competitive results, whereas with the neural networks we were able to outperform baseline SVM scores in both Sub-Challenges.

Index Terms: speech technology, AdaBoost, deep neural networks, rectifier activation function

1. Introduction

Speech technology, besides speech recognition, includes several other tasks as well such as emotion detection [1], speaker recognition [2] etc. Several of these tasks involve the classification of the whole utterance (or a relatively longer part of it), which basically differs from traditional frame-based phoneme classification. In these tasks it is common to have thousands of features, but the main difference is the number of examples. As recording and annotating an audio database can be quite expensive, the number of examples is often just a few hundreds. In these tasks we can still use Gaussian Mixture Models (GMMs [3]) or Artificial Neural Networks (ANNs [4]), which are methods commonly employed for phoneme classification. These methods, however, work ideally when there are several thousands, or even millions of examples, which is not the case in these tasks; therefore it may worth evaluating other methods like SVM [5], AdaBoost.MH [6] and even decision trees [7].

The Interspeech 2014 Computational Paralinguistics Challenge (*ComParE* [8]) consists of two such tasks. In the **Cognitive Load Sub-Challenge**, using the *Cognitive Load with Speech and EGG* (CLSE) database [9], the ternary level of cognitive load (low, medium or high) has to be classified automatically using just the acoustic information. In the **Physical Load Sub-Challenge** (using the *Munich Bio-voice Corpus* [10]) the amount of physical load (low or high) has to be determined, also based on the acoustic data. For the above reasons, we decided to try to solve these tasks using AdaBoost.MH. In addition, we had another reason for doing so: in the ComParE 2013 Challenge [11] we also used this meta-learner algorithm, and we managed to achieve good results: we won one of the sub-challenges, and came second in another [12].

This publication is supported by the European Union and co-funded by the European Social Fund. Project title: Telemedicine-oriented research activities in the fields of mathematics, informatics and medical sciences. Project number: TÁMOP-4.2.2.A-11/1/KONV-2012-0073

However, ANNs have become the dominant method for (phoneme) classification in speech recognition recently, mostly due to the invention of Deep Neural Networks (DNNs [13]). State-of-the-art performance is usually achieved via DNNs [14, 15, 16], but it is also widely used in other areas such as image processing [17] as well. As our earlier experiments also show that they are capable of attaining quite high accuracy scores [18, 19], we thought that it would be worth testing this technique in this challenge too.

Both sub-challenges come with a standard feature set consisting of 6373 attributes. We treated both sub-challenges as simple classification tasks, and did not try to extract additional information from the `wav` files. However, we exploited some other information provided, like the speaker identifier and (in the Cognitive Load Sub-Challenge) the actual task performed, which were used to form groups of specific examples.

2. Learning with AdaBoost.MH

The AdaBoost.MH algorithm [6] is an efficient meta-learner algorithm, which seeks to build a strong *final classifier* from the linear combination of simple, scalar *base classifiers*. The simplest scalar base learner used in practice is the *decision stump*, a one-decision two-leaf decision tree having the form

$$\varphi_{j,b}(\mathbf{x}) = \begin{cases} 1 & \text{if } x^{(j)} \geq b, \\ -1 & \text{otherwise,} \end{cases} \tag{1}$$

where j is the index of a feature and b is the decision threshold. Although boosting decision stumps often yields satisfactory results, the state-of-the-art performance of AdaBoost.MH is usually achieved using *decision trees* as base learners, parametrized by their number of leaves. We also tested a base learner which optimizes *products* of simple base learners [20]; the base learner is parametrized by the number of terms.

Running a full search for each boosting iteration step of AdaBoost.MH is prohibitively expensive, so we decided to run an accelerated version based on a multi-armed bandit (MAB) setup [21]. The general idea of accelerating the base learner is to partition the base classifier space $\mathcal{H}$ into (not necessarily disjunct) subsets $\mathcal{G} = \{\mathcal{H}_1, \ldots, \mathcal{H}_M\}$ and use MABs to learn the usefulness of the subsets. Each arm represents a subset, so, for each iteration, the bandit algorithm selects a subset. The base learner then finds the best base classifier *in the subset* (instead of searching through the *whole space* $\mathcal{H}$), and returns a reward based on this optimal base learner [22]. Based on the reward values, the MAB algorithm learns the quality or usefulness of the subsets. Here, we applied an adversarial bandit algorithm.

We employed an open source implementation – the tool called `multiboost` [23] – where this bandit-based approach is implemented along with the base learners mentioned above.

2.1. The Training Steps

To get an optimal performance from a machine learning method, we need to fine-tune the meta-parameters of the algorithm. AdaBoost.MH fortunately does not require preprocessing steps like feature selection or normalization, and usually no such steps are employed. Although it was reported for the Cognitive Load Challenge that applying specific normalization techniques leads to a significant improvement in the baseline accuracy scores, we adhered to standard practice and did not apply any normalization procedure.

After some preliminary tests using products and trees with different parameters for the base learners, we opted for simple decision stumps. We kept the original training, development and test sets, and used the same general scheme for each challenge and task. We trained 100 models for the training set; then evaluated all of them after each 100 iterations and combined their output via simple majority voting. We chose the iteration where the UAR score was the highest; this value was compared with that of baseline SVM on the development set. In the ComParE 2013 Challenge [11] we then trained 100 new models on the training and development sets altogether, evaluated them on the test set using the iteration count found, and voted them together [12]. Although with this strategy we were ranked first in one of the sub-challenges (*Emotion*), in another (*Autism*) we scored below the baseline score. We think this was due to incorrect iteration determination, since different values might be optimal for different training set sizes, when the development set is significant compared to the training set. As this was the case in both the ComParE 2013 and 2014 challenges, we turned to another solution, which is similar to the general leave-one-out scheme, only subject-wise ("leave-one-subject-out").

We made use of both the training and development sets, but left out the examples belonging to one subject. Then we trained 100 models this way, which were evaluated just for the given subject. We repeated this for all the subjects in the given sub-challenge, e.g. when we had 12 subjects in the training and development sets altogether, this led to 1200 models overall. Lastly, we chose the "optimal" iteration number as the one for which these models produced the best combined Unweighted Average Recall (UAR) score.

Having determined the "optimal" number of iterations, we chose not to train new models on the whole training and development sets together, but using the ones trained in the leave-one-subject-out–manner instead. Although all of them were trained with one subject left out, we had already found the best iteration for them, which may differ if we train models on more examples. So we only evaluated these models on the test database, and submitted these results to the Challenge.

Naturally, it may happen that by aggregating the output of the different base learner configurations (i.e. stumps, products, etc.), or even other machine learning methods via voting could improve the accuracy; however, we considered the number of training examples just too small to get a reliable weighting.

2.2. Cognitive Load Challenge

We followed the approach of the baseline paper, and split the training examples, depending on the task the subject had to perform while speaking (i.e. reading sentence, reading letter, stroop time pressure or stroop dual task), and discarded those that belonged to the reading span letter task. Then we treated these subsets as if they were independent; hence we trained separate models, chose an optimal iteration number etc. separately.

Two tasks (*stroop time pressure* and *stroop dual task*) were completely balanced: all three classes (L1, L2 and L3) had exactly the same number of examples. In the remaining one, however, class L3 was overrepresented. Unbalanced class distribution can also be handled by following different initial example weighting strategies in AdaBoost.MH, but as we could not really improve the accuracy, we downsampled L3 and kept an equal number of examples from all three classes. As we did not want to lose examples, we did this sampling on a per-training basis.

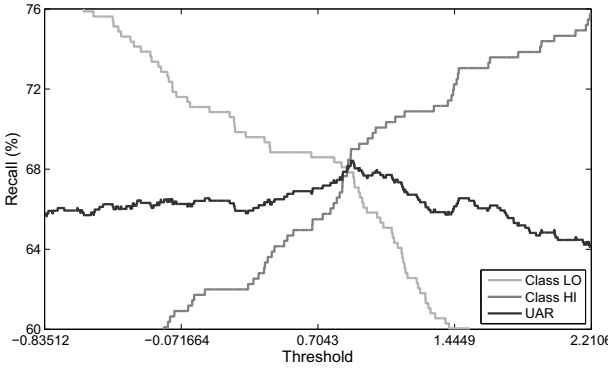

Figure 1: *The effect of adjusting decision threshold to class-wise recalls and the UAR score for AdaBoost.*

2.3. Physical Load Challenge

Although the class distribution was slightly unbalanced, this was a two-class dataset, for which it is quite easy to correct. A classifier chooses class one, when the appropriate posterior score is above a given threshold (in our case zero by default), and chooses class two otherwise. By adjusting this value, we can control the balance between the two classes (see Figure 1). Although there is no guarantee that the optimal value of UAR will be where the recall values for the two classes match, we found that it results in a robust threshold value. So we trained 100 models for each subject in the above mentioned way, resulting in 1200 models overall. Then, for each possible stopping iteration, all models were evaluated, and the threshold value was chosen for which the absolute difference of the recall values for the two classes was minimal. We chose the iteration number for which UAR score was maximal. Finally we evaluated all these models on the test set with this iteration and threshold value.

3. Using Deep Rectifier Neural Networks

Deep neural networks differ from conventional ones in that they consist of several hidden layers, while conventional ANN-based recognizers work with just one hidden layer. The application of a deep structure can provide significant improvements in speech recognition results compared to previously used techniques [13]. But modifying the network architecture also requires modifications to the training algorithm, since the conventional backpropagation algorithm encounters difficulties when training feedforward networks having several layers [24]. As a solution, Hinton et al. presented an unsupervised pre-training algorithm [25]; after this pre-training step, the backpropagation algorithm can find a much better local optimum. But as this pre-training algorithm is extremely CPU-intensive, several solutions have been proposed to alleviate or circumvent the computational burden of pre-training [26, 27, 28], one of them being

deep rectifier neural networks [29].

In the deep neural networks used here we employed rectified linear units as hidden neurons. These units apply the rectifier activation function $\max(0, x)$ instead of the usual sigmoid one [29]. The main advantage of deep rectifier nets is that they can be trained with the standard backpropagation algorithm, without any pre-training, which makes training more straightforward and faster, while maintaining a high level of accuracy.

We used our custom implementation for neural networks, which was developed for phoneme classification, but of course can be applied to other tasks as well [30]. On the TIMIT database, frequently used as a reference dataset for phoneme recognition, we achieved the best accuracy known to us with a phonetic error rate of 16.7% on the core test set [31].

For neural network training, the actual learning rate is crucial; the simplest strategy is to use a pre-defined value for a predefined number of iterations. (During one iteration we usually train the network on all the samples once.) Another possibility is to have a sequence of this process with different numbers of iterations and learning rate values. We followed a more flexible approach: we held out a separated set of training examples, on which the neural network was evaluated after each iteration. When the accuracy failed to improve, we lowered the learning rate; the training was terminated, when the improvement in the error was less than 0.1% in two subsequent iterations.

Neural networks are reported to be sensitive to unnormalized data, so we always normalize our input for zero mean and unit variance (*standardization*). Although in the baseline paper for some sub-challenges slightly better results were reported by normalizing the input into the interval $[0, 1]$, we decided to stick to our former practice. The initial learn rate was set to 0.01. We used 1000 rectifier neurons in each of the 3 hidden layers, and softmax ones in the output layer.

Recall that our neural networks required a separate "hold out" set for setting the actual appropriate training rate, which examples were lost to the back-propagation algorithm. To overcome this drawback, we attempted to fix the number of iterations and learning rates: we withheld the examples of one speaker at a time, and inspected the actual learning rates applied. These, however, varied to such an extent that no such generalization was possible. So we trained ten models for each speaker used as this hold out set, and performed classical backpropagation learning on the rest, then evaluated them and aggregated their output via a voting method. We did this first for the training set, and the results obtained on the development set were compared to the appropriate SVM scores; then we trained our classifiers this way on the training and validation sets united, evaluated them on the test set, and the resulting class labels were submitted for the two sub-challenges.

Note that with this approach, when training on the training set only, we were handicapped compared to the baseline SVM, as we did not "peek" at all into the development set, whereas for SVM this set was used for parameter selection.

3.1. Probabilistic Sampling

Neural nets are sensitive to class imbalances, and tend to behave inaccurately on classes having only a few examples. Our aim was to balance the class distribution by presenting more examples taken from the rarer classes to the learner, for which we applied the training scheme called probabilistic sampling [32]. It is a simple two-step sampling scheme: first we select a class, then randomly pick a training sample from the samples of this class. Selecting a class can be viewed as sampling from a multinomial distribution after we assign a probability to each class:

$$P(c_k) = \lambda \frac{1}{K} + (1 - \lambda) Prior(c_k), \tag{2}$$

where $Prior(c_k)$ is the prior possibility of class c_k, K is the number of classes and $\lambda \in [0, 1]$ is a parameter. If λ is 1, then we get a uniform distribution over the classes, and with $\lambda = 0$ we get the original class distribution. Choosing a value between 0 and 1 for λ allows us to linearly interpolate between these two distributions. In this study we either used the setting $\lambda = 1$ (so we sampled from a uniform class distribution), or we did not use sampling at all, as we found that these settings performed best on the development set.

3.2. Model Output Aggregation

Training a neural network is a non-deterministic procedure due to the random initial weight values. To reduce this effect of uncertainty, it is common to train several models with the same parameters, and aggregate their output in some way. Perhaps the most commonly used way of aggregating outputs is via simple majority voting: we choose the class label which was supported by the largest number of models. It can be readily applied to neural networks, but it is well known that ANNs are able to produce accurate posterior scores, which information is lost during simple majority voting. So we tested another simple aggregation method: for each example and each class we averaged out the output posterior values of all models, and chose the class where this mean value was the highest. We will refer to this method as *probabilistic voting* later on.

3.3. The Cognitive Load Challenge

Our approach for this challenge generally followed that of the authors of the baseline paper [8]: the examples were divided into three groups, based on the task performed by the speaker, and separate DNNs were trained on these subsets (with the exception of the *Reading span letters* task, which was omitted). For each task, we optimized for the UAR score of that subset; although this does not necessarily lead to an optimal UAR score for the whole challenge, we viewed this as the most straightforward way of meta-parameter optimization.

Although the best baseline accuracy scores were obtained via normalization on a per-speaker basis, this information was not available for the test set. The second-best score was got via normalization separately for training and validation sets, in contrast with normalizing all the data together. This can probably be explained by the fact that this strategy lies closer to per-speaker normalization, hence we followed this approach.

We tested both sampling and both voting strategies for all three tasks, although sampling was not strictly required for the tasks *stroop time pressure* and *stroop dual task*. Finally we chose the one which worked best on the development set.

3.4. The Physical Load Challenge

In this sub-challenge the class distribution was only slightly biased towards one of the classes, so it was not clear which sampling strategy would work better. Hence we trained DNNs with both sampling strategies on the training set, and evaluated the resulting models on the validation set, again with both output aggregation algorithms.

Method		Sentence	Time	Dual
DNN	voting	59.97%	73.02%	60.32%
	p. voting	59.88%	74.60%	58.73%
DNN + p.s.	voting	62.91%	71.43%	61.90%
	p. voting	62.96%	73.02%	63.49%
SVM (baseline)		61.20%	74.60%	63.50%

Table 1: *UAR scores obtained for the validation set of the Cognitive Load Challenge with DNNs, for the different sampling (no sampling / probabilistic sampling) and voting (simple majority voting / probabilistic voting) techniques tested.*

Task	Method	Valid.	Test
Reading sentence	AdaBoost	63.76%	—
	DNN	62.96%	—
	SVM	61.20%	61.50%
Time pressure	AdaBoost	76.19%	—
	DNN	74.60%	—
	SVM	74.60%	66.70%
Dual task	AdaBoost	77.78%	—
	DNN	63.49%	—
	SVM	63.50%	56.90%
Cognitive Load	AdaBoost	66.59%	59.35%
	DNN	64.21%	63.05%
	SVM	63.20%	61.60%

Table 2: *UAR scores got for the Cognitive Load Challenge.*

4. Results

The UAR scores achieved on the validation set with DNNs, using the different sampling and voting techniques in the **Cognitive Load Sub-Challenge** can be seen in Table 1. The first thing to notice is that probabilistic sampling clearly helped on the *Reading sentence* task, but its effect is not so beneficial on the task *Stroop time pressure*. This is understandable since the class distribution is already balanced for the latter task; in this case using this sampling technique just introduces a further random factor in the training process, decreasing model robustness. On the other hand, although task *Stroop dual task* looks similar to *Time pressure*, we found the models trained using the probabilistic sampling less sensitive to different subjects. This can be due to that the order of training examples is also important (especially for such tiny datasets), and by probabilistic sampling we reshuffle the training set. Finally we opted for using probabilistic sampling in tasks *Reading span sentence* and *Stroop dual task*. Probabilistic voting also helps in most cases, which can be explained by the good probability estimation capabilities of neural networks. Also note that these scores were obtained by training on the training set only, whereas the baseline SVM score was got by peeking into the validation set.

The results for *stroop time pressure* and *stroop dual task* show signs of overfitting; this is not surprising, however, considering how tiny these two datasets are with only 99 examples for the three classes overall in the training set and 63 in the development one. It may be also the case that neural networks are just too complex for these tasks.

The combined results on the development and test sets (see Table 4) show that deep neural networks consistently performed better on the two sets than baseline SVM. (Unfortunately there were no detailed results for the different tasks.) On the other hand, AdaBoost.MH shows signs of overfitting: although it produced significant improvements over the baseline score on the validation set, we could not carry these over the test set. It was not that surprising, though, since most of the development set scores were peaks in the UAR curve. Still, judged from the results, both methods can be clearly applied as we were able to produce results quite similar to those of the baseline.

Method		Valid.	Test
DNN	voting	62.48%	—
	prob. voting	65.11%	—
DNN + p.sampling	voting	67.12%	—
	prob. voting	68.16%	73.03%
AdaBoost		68.45%	71.43%
SVM (baseline)		67.20%	71.90%

Table 3: *UAR scores obtained for the Physical Load Challenge.*

The results obtained on the **Physical Load Sub-Challenge** are listed in Table 3. It can be seen that for DNNs probability voting strategy is clearly superior to simple majority voting in this sub-challenge. Another observation is that even this slight (cca. 10%) bias towards one of the classes is significant for our Deep Neural Network, at least for such a small dataset, but this shortcoming can be fixed by using probabilistic sampling. The improvement of over 1% over the baseline score could be transferred to the test set again.

As for AdaBoost.MH, we scored slightly below baseline, which means one or two further misclassified items, compared to baseline SVM. This amount of error may come from the non-optimal tradeoff between the two classes as well; nevertheless, this performance still could be worth applying in practice.

We also note that from the base learner configurations tested on AdaBoost.MH, stumps proved to be the best. This can probably be explained by the size of the datasets: all four databases are quite small, but the *Stroop time pressure* and *Stroop dual task* sets have an extremely small set of examples. In this case, decision trees can be one of the best methods, which is practically the same as AdaBoost.MH using decision stumps.

5. Conclusions

We applied two state-of-the-art machine learning methods in the Interspeech 2014 Computational Paralinguistics Challenge: AdaBoost.MH and Deep Rectifier Neural Networks. Our results showed that, although with the former algorithm we scored slightly below baseline, both methods produced competitive performances, and DNNs consistently managed to outperform baseline SVM. We experienced that the two methods gave quite different outputs: with neural networks we had a hard time classifying the two tiny tasks, whereas AdaBoost.MH we could only handle the unbalanced class distribution of the *Reading span sentence* task by downsampling. Therefore it may worth mixing the two learning methods in the Cognitive Sub-Challenge, solving different tasks with different algorithms; another idea worth testing is evaluating both algorithms and combining their output. These are, however, considered as future work, in the case if the datasets will be released to the public in the future.

6. References

[1] L. Vidrascu and L. Devillers, "Detection of real-life emotions in call centers," in *Proceedings of Interspeech*, Lisboa, Portugal, 2005, pp. 1841–1844.

[2] D. A. van Leeuwen, A. F. Martin, M. A. Przybocki, and J. S. Bouten, "NIST and NFI-TNO evaluations of automatic speaker recognition," *Computer Speech & Language*, vol. 20, no. 23, pp. 128 – 158, 2006.

[3] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.

[4] C. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[5] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[6] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.

[7] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[8] B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The INTERSPEECH 2014 computational paralinguistics challenge: Cognitive & physical load," in *Proceedings of Interspeech*, 2014.

[9] T. F. Yap, "Speech production under cognitive load: Effects and classification," Ph.D. dissertation, University of New South Wales, 2012.

[10] B. Schuller, F. Friedmann, and F. Eyben, "Automatic recognition of physiological parameters in the human voice: Heart rate and skin conductance," in *Proceedings of ICASSP*, 2013, pp. 7219–7223.

[11] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The Interspeech 2013 Computational Paralinguistics Challenge: Social signals, Conflict, Emotion, Autism," in *Proceedings of Interspeech*, Lyon, France, 2013.

[12] G. Gosztolya, R. Busa-Fekete, and L. Tóth, "Detecting autism, emotions and social signals using AdaBoost," in *Proceedings of Interspeech*, Lyon, France, Aug 2013, pp. 220–224.

[13] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[14] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," in *Proceedings of ICASSP*, Vancouver, Canada, 2013, pp. 6985–6989.

[15] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proceedings of ICASSP*, Vancouver, Canada, 2013, pp. 8599–8603.

[16] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On rectified linear units for speech processing," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3517–3521.

[17] S. Zhou, Q. Chen, and X. Wang, "Convolutional deep networks for visual data classification," *Neural Processing Letters*, vol. 38, no. 1, pp. 17–27, 2013.

[18] T. Grósz and L. Tóth, "A comparison of deep neural network training methods for large vocabulary speech recognition," in *Proceedings of TSD*, Plzen, Czech Republic, 2013, pp. 36–43.

[19] L. Tóth, "Convolutional deep rectifier neural nets for phone recognition," in *Proceedings of Interspeech*, Lyon, France, 2013, pp. 1722–1726.

[20] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *Proceedings of ICML*, vol. 26, Montreal, Canada, 2009, pp. 497–504.

[21] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[22] R. Busa-Fekete and B. Kégl, "Fast boosting using adversarial bandits," in *Proceedings of ICML*, vol. 27, 2010, pp. 143–150.

[23] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, "MultiBoost: a multi-purpose boosting package," *Journal of Machine Learning Research*, vol. 13, pp. 549–553, 2012.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS*, 2010, pp. 249–256.

[25] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[26] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011, pp. 24–29.

[27] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary conversational speech recognition," Dept. Comp. Sci., University of Toronto, Tech. Rep., 2012.

[28] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of AISTATS*, 2011, pp. 315–323.

[30] T. Grósz and I. Nagy T., "Document classification with neural networks and probabilistic sampling," in *Proceedings of TSD*, 2014.

[31] L. Tóth, "Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition," in *Proceedings of ICASSP*, 2014.

[32] L. Tóth and A. Kocsor, "Training HMM/ANN hybrid speech recognizers by probabilistic sampling," in *Proceedings of ICANN*, 2005, pp. 597–603.