Robust Phoneme Recognition with Little Data

Christopher Dane Shulby
Institute of Mathematical and Computer Sciences – University of Sao Paulo, Brazil
Samsung SIDI Institute, São Paulo, Brazil
www.nilc.icmc.usp.br
c.shulby@icmc.usp.br

Martha Dais Ferreira
Institute of Mathematical and Computer Sciences – University of Sao Paulo, Brazil
daismf@icmc.usp.br

Rodrigo F. de Mello
Institute of Mathematical and Computer Sciences – University of Sao Paulo, Brazil
mello@icmc.usp.br

Sandra Maria Aluisio
Institute of Mathematical and Computer Sciences – University of Sao Paulo, Brazil
sandra@icmc.usp.br

Abstract

A common belief in the community is that deep learning requires large datasets to be effective. We show that with careful parameter selection, deep feature extraction can be applied even to small datasets.We also explore exactly how much data is necessary to guarantee learning by convergence analysis and calculating the shattering coefficient for the algorithms used. Another problem is that state-of-the-art results are rarely reproducible because they use proprietary datasets, pretrained networks and/or weight initializations from other larger networks. We present a two-fold novelty for this situation where a carefully designed CNN architecture, together with a knowledge-driven classifier achieves nearly state-of-the-art phoneme recognition results with absolutely no pretraining or external weight initialization. We also beat the best replication study of the state of the art with a 28% FER. More importantly, we are able to achieve transparent, reproducible frame-level accuracy and, additionally, perform a convergence analysis to show the generalization capacity of the model providing statistical evidence that our results are not obtained by chance. Furthermore, we show how algorithms with strong learning guarantees can not only benefit from raw data extraction but contribute with more robust results.

2012 ACM Subject Classification Computing methodologies → Speech recognition

Keywords and phrases feature extraction, acoustic modeling, phoneme recognition, statistical learning theory

Digital Object Identifier 10.4230/OASIcs.SLATE.2019.4

Acknowledgements We would like to thank our NILC colleagues for their input and support. A great thanks goes to CEMAI. It was only possible to run these experiments thanks to the Euler super-computer cluster at the ICMC – University of Sao Paulo.

1 Introduction

Acoustic modeling, or the statistical representation of speech signals, is essential for parametric ASR (Automatic Speech Recognition). Generally PER (Phone Error Rate) or FER (Frame Error Rate) are used to measure model performance. FER is a more exact metric, since it shows exactly what the acoustic model is capable of and PER is more interesting for applications because in the end, this is the goal of the acoustic model. Still, some kind of PER smoothing technique is inevitably used. For the field of ASR, large companies have been positive on one hand, as they have matured the technology into production ready algorithms and increased the access to machine learning libraries like never before, but negative when it comes to publishing reproducible work. Often proprietary databases are used to train for benchmark results as in [6, 12] and PER or FER is rarely given. One of our principal objectives has been to be as transparent as possible and establish metrics for fair comparison of acoustic models. We also show that deep learning techniques, combined with careful engineering can be useful even for non-big-data scenarios, especially in the case of raw feature extraction (completely statistical-based). In this article, we experiment with context windows to better understand the performance of the model with respect to phonemic context and then, we demonstrate the robustness of the model with the learning guarantees provided by the Vapnik Chervonenkis (VC) theory [23]. It is important to accurately model acoustic properties so that errors do not propagate to other models. Reliable phoneme-level error detection is still a great need for automatic pronunciation training. This is not trivial, since acoustic models are still quite far from perfect and most ASR pipelines rely on pronunciation models (and sometimes language models) to overcome these deficiencies. In general, there is not an equal share of the wealth of data resources and the majority of the world's languages have not benefited from deep speech technology for a number of reasons: i) they require thousands of hours of transcribed/annotated data [17]; ii) they require a respectable infrastructure with large scale GPU-computing and often take weeks or months to train [5]; and iii) they do not offer solid learning guarantees [26]. More specifically, the amount of training data is a limiting factor and for more robust applications the amount of training data could grow exponentially with augmentation techniques like MCT (Multi-conditional Training) [13]. Secondly, infrastructures which can adequately support this type of training exists in very few university laboratories and is mostly limited to the private sector. Lastly, these vastly trained models which have been long suspected and shown by [26] to be great data memorizers, but poor generalizers and difficult to scale in real world applications, even in the best conditions. In this paper, we focus on reproducible results, showing that raw feature extraction can still be used for SotA (State of the Art) ASR in low-resource training environments, where careful parameter selection and good architecture choices can compensate for a deficit in data. We chose to use a CNN (Convolutional Neural Network) as the feature extractor for this work, because of raw feature extraction via convolution and dimension reduction by max pooling. Since human experts are capable of recognizing phonemes in a spectrogram given the concentration of visual formants, it seems reasonable to ask this of a computer vision algorithm which was inspired by a mammalian visual cortex. This is why we chose to treat this as an image processing problem, instead of a more classical signal processing approach which would use the Discrete Fourier Transforms directly. In Section 3, we describe the TIMIT database used in our experiments, the preprocessing steps taken for images used in the CNN feature extractor and the architecture proposed for the experiments. The experiments, results and discussions are presented in Section 4 followed by the convergence analysis in Section 5. Finally we will make our concluding remarks in argument for reproducible and robust acoustic models in Section 6.

2 Related Work

The CNN has proven to be useful for phoneme recognition. [1] proposed a hybrid CNN-HMM model using local filtering and max-pooling in the frequency domain where a strong benchmark of 21.6% PER is established. Their network is also pretrained on 18 hours of Google voice search data. [18] explore the optimal CNN architecture presenting the best results with large corpora (300-400 hours). While the results are promising for SotA applications using a CNN, both of these strategies require a great deal of resources to train and can be difficult for comparison by other researchers without access to these datasets or pre-trained weights. One can assume that some pre-processing on the images and data smoothing was used to generate PER. It is also likely a pronunciation and possibly language model was used after the acoustic model's posterior probabilities were generated, but these details are not clarified in those works. FER could be useful and is not given in most SotA papers like [16, 2, 18, 1, 11, 21]

Table 1 shows the SotA results we were able to compile, which use both FER and PER metrics. In the table, each study is listed with the method used, whether the paper provides sufficient information to reproduce it exactly (REP? - short for reproducible) and the PER and FER. Studies without both PER and FER were not included in the table.

Table 1 SotA Results using PER and FER as metrics.

Ref	**Method**	**REP?**	**PER**	**FER**
[10]	DBLSTM-RNN	N	**17.7**	27.9
[20]	CNN + CTC	N	29.4	**22.1**
[22]	DLSTM-RNN	Y	25.4	29.4

In 2013, [10] benchmarked the TIMIT corpus at 17.7% PER and 27.88% FER. This DBLSTM-RNN (Bidirectional Long Short-Term Memory-Recurrent Neural Network) included 3 Hidden layers with 250 units in each and pre-trained (from a larger dataset) CTC (Connectionist Temporal Classification) finite state transducers. The 50 speaker development set was used for fine-tuning and early stopping as well as a biphone language model for predictions. Interestingly, [22] did make a best-effort attempt to reproduce the network by [10]. The main difference was that a DLSTM instead of a DBLSTM is used. The author explains that this was because of the lack of availability of a Bi-directional LSTM within the TensorFlow library (as a project scope setting). Still, he does use a three layer LSTM network with 250 hidden units. The author used the openly available default initialization from TensorFlow based on [25], since the data from [10] was not available. His network used a mini-batch size of 6 sub-sequences of 20 frames and applied dropout regularization. The final results of this reproducible network were 29.43% FER and 25.36% PER. The FER results are quite comparable to [10] but the PER is much higher. This is probably due to the bi-phone language model. It also seems like the pretraining was rather similar (using the TensorFlow default) to the database that was used in the previous studies by [9] and [10]. As far as FER is concerned, the best work is [20], beating the mark set by [10]. This technical report is interesting since it uses a CNN as well with CTC and achieves a FER of 22.1% but does not break the mark for PER. While the authors do not offer any evidence for the generalization capacity, they do explain their training philosophy as stated in the article: "we train until the model begin[s] to overfit on the training set and the dev accuracy begins to fall. Much of the training is done on SAIL's Deep clusters, which uses nVidia GTX780 GPUs". The network with the best performance was the 25 frame windowed 128-256-384-384 CNN followed by 1024-512 dense layers. This means that the network is very large and likely makes use of a lot of data. They also do not give the hyper-parameters used in their CNN and explain that they used their own pretrained language model for predictions and a pretrained RNN-CTC from some dataset (also not described). Since none of these resources are made available, this study is unfortunately not reproducible either.

3 Dataset, Features and Architecture

3.1 Dataset and Feature Extraction

We used the TIMIT Corpus. This corpus was used because it has been an industry benchmark for decades and is a small (ca. 5 hours) dataset which is phonetically balanced for English. As suggested in [14], we collapsed the phone set into 39 monophones. We did not; however, train the 61 phone set before collapsing so as not to dilute the training samples. To be consistent with other works, we also removed silence frames from the final predictions for FER and PER. We extracted the features from spectrogram images which we preprocessed with sox, from Hann windows of 25ms with a stride of 10ms. The spectrogram of each 25ms window has 5x128 color pixels according to the sox color palete. This means that each pixel in time direction corresponds to 5ms (200 pixels per second) and 128 frequency points are taken in the frequency direction (DFT size of 254). Afterwards, we searched for the best kernels to adequately represent our dataset, as explained in subsection 3.2.

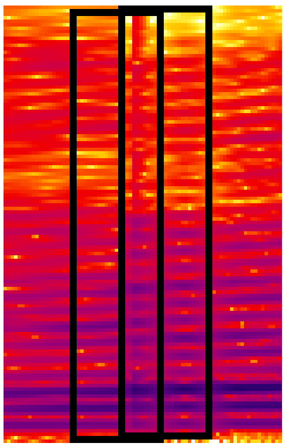

Figure 1 Spectrogram illustrating 25ms Hann windows with a stride of 10ms.

3.2 Architecture

3.2.1 CNN Feature Extractor

Our parameters network architecture, was estimated using an approach based on False Nearest Neighbors (FNN) proposed by [8]. This technique estimates the kernel sizes to best represent data patterns and found the most aggressive pooling layers to lower the dimensions of our feature maps since these would then be passed on to a SVM (Support Vector Machine). This method was used to generate the best five configurations and from those options, we chose the simplest configurations which was decided by using the one which presented the largest masks with the fewest neurons. The final parameters used were 3 Convolutional layers: 1. 36 units, kernel=15×2; 2. 31 units, kernel=15×1 + max-pooling=3×3, stride=2×2; 3. 15 units, kernel=8×1 + max-pooling=3×3, stride=2×2. ReLU activation was applied due to its widespread adoption in literature. We used a batch size of 128, a learning rate of 0.01 and trained over 31 epochs. The convolutional network required 8 hours using a single Titan-X GPU and 32GB of RAM. In the SVM, we augmented the context while skipping some frames. The defining principle in our frame skipping is that we left the two adjacent frames (1 right, 1 left) always intact, believing that these frames are the most important ones. After the three middle frames, we skipped every other frame until we reached the extremity of the window. For example if we have 11 overlaping frames (25ms/step of 10 ms) of context, we would only train the first, third, fifth, sixth, seventh, ninth and eleventh frames, considering the sixth as the central frame. This would be considered 1 instance for training and the next instances would follow the 10ms step size for the next frame until the end of the audio (the first and last 10 full seconds are padded with zeros). The full pipeline can be seen in Figure 2. We found that frame skipping greatly reduces the number of features with very little impact on results as later explained in section 4.

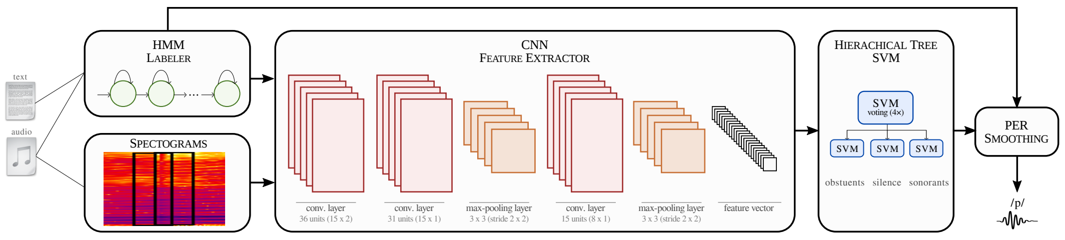

Figure 2 CNN-HTSVM architecture defined for the experiments.

3.2.2 HTSVM Classifier

The SVM parameters were found empirically after several experiments. The selected kernel for final experiments was a 4th order polynomial kernel with $coef0 = 1$ (as a non-homogeneous kernel) and a cost $C = 10,000$. The interested reader can find more details about the exact architecture of the hierarchical tree structure in [19]. One of the biggest issues of the TIMIT dataset is that it is not balanced with respect to its classes. For example: in the training set, the phoneme /k/ appears in $60,433$ frames, whereas /g/ is found in only $17,727$. In order to build a robust system, it is important to learn this phonemic distinction and minimize the influence of probability in the training set. We were able to deal with this by using the SMOTE [4] data augmentation technique. In order to obtain PER, the classified frames were converted to phonemes by taking the mode of all of the SVM classifications for each HMM boundary generated in the labeler.

4 Experiments and Results

We ran two types of experiments in this paper. The first experiments were designed to demonstrate the cost/benefit of window skipping and can be seen in Table 2. An asterisk symbol in the number of frames column denotes that window-skipping was used and in the classifier column indicates that stochastic gradient descent (SGD) was used, otherwise the solver was Adam. Once we completed the MLP (Multi-Layer Perceptron) experiments, we ran selected experiments with the SVM.

Table 2 Experiments on CNN features with window-widening.

Frames	Classifier	FER	PER	F1
1	MLP	0.49	0.54	0.36
3	MLP	0.45	0.50	0.41
5	MLP	0.43	0.48	0.45
7	MLP	0.41	0.45	0.48
9	MLP	0.40	0.44	0.49
11	MLP	0.39	0.43	0.51
11*	MLP	0.39	0.43	0.50
11*	MLP w/SGD	0.38	0.42	0.51
13	MLP	0.39	0.42	0.51
15*	MLP w/SGD	0.37	0.40	0.52
19	MLP w/SGD	0.37	0.39	0.52
1	SVM	0.42	0.47	0.44
3	SVM	0.41	0.45	0.48
11*	SVM	0.30	0.33	0.61
19*	SVM	0.28	0.32	0.63

The second set of experiments were designed to compare our algorithm with the SotA. Here we compare our results to the replication study done by [22], our attempted replication of the famous CNN from [1] (without any pre-training or language model) and the traditional GMM-HMM as a baseline classifier which is generally used for classification on small datasets. As a traditional baseline, we used one of the most popular ASR toolkits for a database the size of TIMIT, the HTK toolkit. A triphone HTK model with 31 Guassians was trained on the same TIMIT training set used in our method and recognition was performed on the test set with a zero-gram language model and only the individual monophones as the pronunciation model in order to obtain only the posterior values from the acoustic model. We used 31 Gaussian components because this number is one which we have found useful in the past. It is higher than what is recommended by the voxforge tutorial [15], which uses 15 and is similar to the models used by Keith Vertanen [24], where he uses a maximum of 32 for the Wall Street Journal and TIMIT datasets together. The model was trained using MFCC 0DANZ acoustic features, where 0 uses the zeroth cepstral coefficient, D=delta coefficients, A=delta delta acceleration coefficients, N=absolute energy suppression and Z=zero mean normalization. The predictions were then segmented in the same fashion as the proposed method with 25ms sliding windows and a step of 10ms. In the case of the CNN replication, we used the closest parameters possible to those given in [1] as follows: conv layer 1: $1,000$ units, kernel size = 8×8; max-pooling 1: 5×5 / 2×2; conv layer 2: $1,000$ units, kernel size=8×8; max-pooling 2: 5×5 / 2×2, flattened into a 512 unit dense layer and softmax. This created a feature map with 4,224,000 dimensions. The first 15 epochs were trained with a learning rate of 0.08 and the last 10 with 0.002. A batch size of 128 and 11 frames for context and ReLu activation were used. One adaptation should be pointed out: the 6×6 max-pooling layer created negative divisions (due to an unknown pre-processing step) so as a conservative measure, we used the maximum possible which was 5×5. The network overfit the data quickly to 87% accuracy after the first 15 iterations. We used the same HMM labeler for all experiments and all other default parameters in the CNN were kept the same. Table 3 presents the FER, PER and F1 scores of the experiments.

Table 3 F1 Scores in Frames, Frame Error Rates and Phone Error Rates for each model.

Study	Classifier	FER	PER	F1
This work Baseline	GMM-HMM	0.76	0.75	0.16
This work Rep.[1]	HMM-CNN	0.58	0.52	0.31
[22] Rep.[10]	DLSTM RNN	0.29	**0.25**	n/a
This work	CNN-HTSVM	**0.28**	0.32	**0.63**

It should be pointed out that the poor performance of the HMM-CNN was expected on such a small dataset. The network was built to be trained using the weights already aquired from the larger private dataset used in that study. Also, the poor performance of the vanilla GMM-HMM model can be explained since the tied-state probabilities are highly dependent on the pronunciation and language models used in the traditional decoder which we did not use here. Our purpose was simply to show that this method does not perform well in frame-wise classification, even though it is widely used for speech-recognition tasks with small datasets. On the other hand the replication studies can be compared and show that when the proper tools are used, excellent results can be achieved, even when large amounts of data are not available.

Independent of the models accuracy, it is also important to understand what sort of errors the models are actually committing. Table 4 lists the 15 most frequent errors committed by each system, including the true values, predicted values and the confusion percentage. The GMM-HMM systems are known to produce rather jumbled posterior values since they typically rely on providing a ranking to the pronunciation model where these issues are normally solved. Still, this is not an adequate approach for phoneme recognition.

Table 4 Most Frequent FER Confusion percentages.

GMM-HMM			CNN-MLP			CNN-HTSVM		
True	Pred	Conf (%)	True	Pred	Conf (%)	True	Pred	Conf (%)
s	z	33.14	s	f	21.48	s	z	15.16
ih	uw	16.00	ih	ae	16.23	ay	ae	39.64
t	ch	17.58	iy	ae	14.99	ao	aa	26.58
er	r	32.46	z	f	28.01	r	er	18.84
ao	l	28.00	ay	ae	24.59	sh	s	26.01
iy	y	14.23	eh	ae	25.46	aa	ae	16.07
s	sh	10.09	aa	ae	20.96	ah	ae	14.79
ae	t	14.32	er	ae	14.32	t	s	7.61
ih	z	10.07	k	t	13.22	iy	ih	6.48
w	ao	45.52	ey	ae	25.29	er	r	7.64

Here one can see that the findings in studies like [3] can be confirmed that the SVM does get confused when phonemes are very similar. Still, this may be caused by the transcription of the 8 dialects where some sounds, especially vowel sounds can have some overlap. One observation between the SVM and MLP classifiers is that the MLP makes more erratic errors. Since the SVM is governed by a decision boundary function and the MLP is a probability of an argmax function, this makes sense. Even though the same CNN features were extracted, it seems that the MLP was more likely to develop a “trash” category where when in doubt it goes with a “more probable” class. In the example of /ae/, it was correctly classified 60% of the time so it became a go-to class when in doubt. In the case of /f/, the phoneme was classified correctly in 85% of the instances, so other fricatives were more likely to be classified as /f/ as well. The GMM-HMM is notoriously unsuccessful on the phoneme level which is why HMM-based engines use tied states to calculate the cost of substitutions to find the correct transcription in the pronunciation model. What is most evident is that often when it errs, it fails badly. The MLP is also not the most graceful failure, where the SVM is more robust in that its errors are more reasonable as pointed out in [3].

5 Convergence analysis

We believe that it is important to present statistical evidence that results are not found by chance. To save space, all formulas referred to in this section can be found below and are numbered for easy reference:

$$P\left(\sup_{f\in\mathcal{F}} |R(f) - R_{emp}(f)| \geq \epsilon\right) \leq \delta \tag{1}$$

$$\delta = 2\mathcal{N}(\mathcal{F}, 2n)e^{-n\epsilon^2/4} \tag{2}$$

$$R(f) \leq R_{emp}(f) + \sqrt{4/n(\log(2\mathcal{N}(\mathcal{F}, n)) - \log(\delta))} \tag{3}$$

$$R(f) \leq R_{emp}(f) + \sqrt{c/n(R^2/\rho^2 - \log(1/\delta))} \tag{4}$$

$$\begin{aligned} s(n) &= (198.4n^2 - 2574.8n + 6744.3)^{36} \\ &+ (150.6n^2 - 2000.9n + 5386.5)^{31} \\ &+ (89.3n^2 - 1200.5n + 3304.4)^{15} \\ &\lim_{n\to\infty} \frac{\log\{s(n)\}}{n} \approx 0, \end{aligned} \tag{5}$$

$$\begin{aligned} R(f) \leq 0.28 + \\ \sqrt{4/n(3,332,567^2/173,869,050^2 - \log(1/\delta))} \end{aligned} \tag{6}$$

SLT (Statistical Learning Theory) provides theoretical support for such convergence proofs in terms of how supervised learning algorithms generalize examples. The empirical risk minimization [23] defines the main principle of SLT. As seen in equation 1, that formulation intends to bound the divergence ϵ between the empirical risk R_{emp}, i.e., the error measured in a sample, and the expected risk $R(f)$, i.e., the expected error while assessing the joint probability distribution of examples and their respective classes, as the sample size n tends towards infinity. In the same equation, f refers to some classifier, and $\mathcal{F}$ is the space of admissible functions provided by some supervised algorithm, a.k.a. the algorithm bias as described in [23] and [7]. Additionally, Vapnik proved a bound for supervised learning algorithms considering the shattering coefficient $\mathcal{N}(\mathcal{F}, 2n)$. Such a coefficient is a measure function to compute the complexity of the algorithm bias, i.e., the cardinality of functions contained in the space $\mathcal{F}$ that produce different classification outputs, provided a sample size

n. Throughout our formulation, we employ the generalization bound defined in equation 3 to ensure that the expected risk is bounded by the empirical risk plus an additional term associated with the shattering coefficient and some probability δ. In the case of the SVM, the same bound is formulated as shown in equation 4, in which c is some constant, R corresponds to the dataset radius, and ρ represents the maximal margin. In this context, we assessed our CNN and SVM to understand the sample size they require to ensure learning in the context of speech recognition, allowing to estimate their expected risk value over unseen examples. Now, we proceed by computing the generalization bound for the CNN (equation 3). Considering $\delta = 0.05$, which represents a probability of 0.95 (i.e., 95%) to ensure that the empirical risk $R_{\text{emp}}(f)$ is a good estimator for the expected risk $R(f)$, meaning the error results measured for our classifier indeed work on unseen examples. The convergence curve (CC) obtained from the same equation is shown in Figure 3. Observe that our CNN requires at least $985,128$ examples to converge, while we had in practice $1,447,869$ examples in training set.

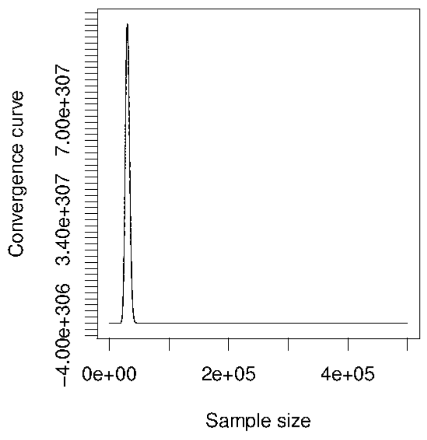

Figure 3 CC, from eq. 5.

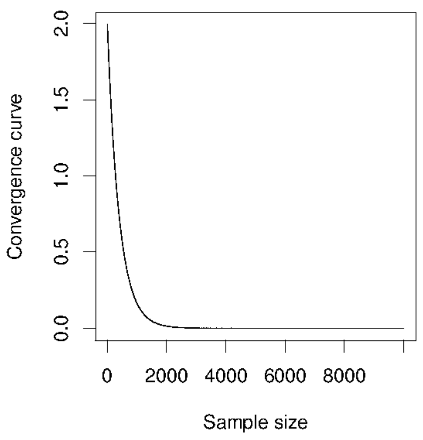

Figure 4 CC, from eq. 6.

We can employ another result from Vapnik to prove that our CNN converges. Equation 5 simplifies the previous equation and considers only the most relevant term to prove learning convergence. Notice that as the term $\frac{\log CNN(n)}{n}$ approaches zero, term $\sqrt{4/n(\log(2CNN(n)) - \log(0.05))}$ goes to zero, remaining the empirical risk as an assessment measure of the learning performance. Next, the SVM is also analyzed considering equation 4. In this case, we have an accuracy of 0.72 leading to $v(f) = 1 - \text{accuracy} = 1 - 0.72 = 0.28$, $R = 3,332,567$ (the radius we estimated for the whole dataset), and $\rho = 173,869,050$ as the maximal margin found. Consequently, the generalization bound for our SVM is defined in equation 6. Also, considering $\delta = 0.05$ as before, and $c = 4$ as taken in the default formulation (Equation 3), Figure 4 illustrates the convergence curve obtained for the SVM. Notice our SVM requires at least $11,981$ examples to converge, while we had in practice $1,447,869$ examples in the training set. Based on [7], we conclude the convergence analysis for our CNN and SVM, ensuring learning in the context of speech recognition.

6 Conclusion

In this paper, we have focused on a feature extraction approach from two biases: 1.) a domain bias where we took great care in preprocessing the data in a way which would minimally preserve the most important acoustic information; and 2.) a statistical learning bias where we checked in every step that we had the most simple approach which would guarantee learning and would lead to the least complex hypothesis space possible. This work shows that even without pretraining or fine-tuning, we achieve better FER results than the SotA replication by [22] and we are not so far from the reported SotA which cannot be faithfully reproduced [10, 20] and better than our reproduction tests of the CNN by [1]. Large networks like these cannot generalize well for small datasets without pretraining or heavy regularization as we can see a large difference in the training and test scores as well as the F-measure. As for our CNN, we would like to point out that although the best results were obtained with 19 frames, we believe that 11 frames seems to be the most cost-effective number, since larger configurations demand a much larger number of feature maps and offer only little improvement. The multiple max-pooling layers were great facilitators for the window widening technique since without this dimension reduction the SVM would have been prohibitive to train and even the MLP would have been very difficult on our laboratory work-station. Therefore, while more layers in the CNN architecture do not seem necessary for better feature extraction, they can help reduce dimensionality which aids in classification. We show that careful parameter selection and attention to supervised learning guarantees are essential for building robust models with little data. We also believe that the results from the convergence analysis further evidences the robustness of this strategy and show that with enough examples we are able to guarantee learning. Without this guarantee, it is possible that the results could have been obtained by chance. We hope that this work will inspire others to seek statistical evidence to support their models and that they will describe them in a way which can be reproduced.

7 Future Work

For future work, it would be interesting to test our method on other datasets, especially those with accented speech, noisy speech and different recording conditions, since we believe that the CNN features could be robust enough to deal with them.

It would also be interesting to see what effect more data has on the classifier to see how well this method performs when data is not so limited. Of course, this would also mean devising strategies to train more data with a SVM, which would need to employ sample reduction techniques.

References

1 Ossama Abdel-Hamid, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

2 Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

3 Rimah Amami and Noureddine Ellouze. Study of phonemes confusions in hierarchical automatic phoneme recognition system. *arXiv preprint*, 2015. `arXiv:1508.01718`.

4 Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

5 Xie Chen, Adam Eversole, Gang Li, Dong Yu, and Frank Seide. Pipelined Back-Propagation for Context-Dependent Deep Neural Networks. In *Interspeech*, pages 26–29, 2012.

6 Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Katya Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint*, 2017. `arXiv:1712.01769`.

7 R. Fernandes de Mello, M. Dais Ferreira, and M. Antonelli Ponti. Providing theoretical learning guarantees to Deep Learning Networks. *ArXiv e-prints*, 2017.

8 Martha Dais Ferreira, Deborah Cristina Correa, Luis Gustavo Nonato, and Rodrigo F de Mello. Designing Architectures of Convolutional Neural Networks to Solve Practical Problems. *2017 Elesvier pre-print*, 2017.

9 Alex Graves and Navdeep Jaitly. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *ICML*, volume 14, pages 1764–1772, 2014.

10 Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

11 Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs. *arXiv preprint*, 2014. `arXiv:1408.2873`.

12 Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. Cache based recurrent neural network language model inference for first pass speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6354–6358. IEEE, 2014.

13 Tae Yoon Kim, Chang Woo Han, Sangha Kim, Donghoon Ahn, Seokyeong Jeong, and Jae Won Lee. Korean LVCSR system development for personal assistant service. In *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, pages 93–96. IEEE, 2016.

14 K-F Lee and H-W Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

15 Ken MacLean. Tutorial: Create Acoustic Model - Manually, 2018. Accessed: 2018-03-01. URL: `http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial/triphones/step-10`.

16 Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

17 Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

18 Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618. IEEE, 2013.

19 Christopher Dane Shulby, Martha Dais Ferreira, Rodrigo F de Mello, and Sandra Maria Aluisio. Acoustic Modeling Using a Shallow CNN-HTSVM Architecture. *2017 Brazilian Conference on Intelligent Systems*, pages 85–90, 2017.

20 William Song and Jim Cai. End-to-end deep neural network for automatic speech recognition. *Technical Report*, 2015.

21 László Tóth. Phone recognition with hierarchical convolutional deep maxout networks. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):25, 2015.

22 Timo van Niedek, Tom Heskes, and David van Leeuwen. Phonetic Classification in TensorFlow. *Bachelor's Thesis, Radboud University*, 2016.

23 Vladimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

24 Keith Vertanen. HTK Acoustic Models, 2018. Accessed: 2018-03-01. URL: `https://www.keithv.com/software/htk/us/`.

25 Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint*, 2014. `arXiv:1409.2329`.

26 Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint*, 2016. `arXiv:1611.03530`.