INVESTIGATIONS ON BYTE-LEVEL CONVOLUTIONAL NEURAL NETWORKS FOR LANGUAGE MODELING IN LOW RESOURCE SPEECH RECOGNITION

Kazuki Irie, Pavel Golik, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
{irie, golik, schlueter, ney}@cs.rwth-aachen.de

ABSTRACT

In this paper, we present an investigation on technical details of the byte-level convolutional layer which replaces the conventional linear word projection layer in the neural language model. In particular, we discuss and compare the effective filter configurations, pooling types and the use of bytes instead of characters. We carry out experiments on language packs released by the IARPA Babel project and measure the performance in terms of perplexity and word error rate. Introducing a convolutional layer consistently improves the results on all languages. Also, there is no degradation from using raw bytes instead of proper Unicode characters, even on syllabic alphabets like Amharic. In addition, we report improvements in word error rate from rescoring lattices and evaluate keyword search performance on several languages.

Index Terms— language modeling, convolutional neural networks, speech recognition, keyword search

1. INTRODUCTION

Neural networks which process text on character-level have become increasingly popular for natural language processing. The motivation behind such an approach can be linguistic, considering characters as the atomic units of text, or technical, e.g. for an open-vocabulary automatic speech recognition (ASR). Another strong motivation is the success of neural networks which operate on raw level signal in other fields, such as image generation from pixels [1] or acoustic modeling from raw waveform [2, 3]. This trend encourages text related tasks to also operate on the level of characters or even bytes. In fact, many successful examples of character-level language processing with neural networks have been already reported, both with convolutional neural networks (CNN) [4] and with recurrent neural networks (RNN) such as long short-term memory (LSTM) [5]. Zhang et al. [6] have used character-level convolutional neural networks for text classification. For language modeling, word embeddings based on characters have been suggested: both the RNN-based approach by Ling et al. [7] and the CNN-based approach by Kim et al. [8] have been shown to perform better than the conventional word projection. These two architectures have also been applied to machine translation [9, 10].

Partially supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract no. W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

2. RELATED WORK

Kim et al. [8] introduced a CNN-based input layer for language modeling, which builds the word feature from the character sequence representing the word. The corresponding feature is fed to an LSTM language model and the prediction is done at word level. The effectiveness of such a technique has been confirmed by Jozefowicz et al. [11]. In this paper, we develop a convolutional layer over bytes. Certainly, characters define the atomic units of text in many languages. However, when the language uses characters outside ASCII, character-level tokenization might require hand-crafted pre-processing. In such a scenario, byte-level processing would be more attractive. Previously Gillick et al. [12] has shown a byte-level RNN to perform well for text labelling.

3. NEURAL LANGUAGE MODEL

3.1. Topology

The neural language model architecture we consider is based on Kim et al.'s topology [8], as depicted in Figure 1. The model consists of a byte-level 1D convolution layer, a pooling layer which is optionally followed by highway layers, a linear bottleneck layer and and an LSTM layer. The output word-level softmax layer is factorized with word classes for all models in this work. For the details of the LSTM language model, we refer to [13].

3.2. Byte-level convolutional layer

In the first layer of a regular word-level neural language model the input word represented as a one-hot vector is pro-

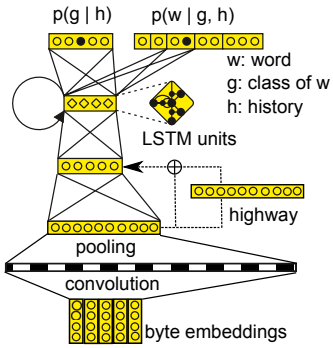

Fig. 1. *LSTM language model with CNN-based input layer.*

jected on a continuous low-dimensional embedding. In the model proposed by [8], the input word is first decomposed into characters, which are then projected on a sequence of low-dimensional character-embeddings. In this work we define a convolution over the continuous byte-embeddings. Again, the input and the output of the neural LM are defined for full words. The convolutional layer merely serves the purpose of internal word representation. The computations carried out through the convolutional layer and the max-pooling layer for a single filter are depicted in Figure 2. Similar to the character-level convolution in [8], we consider the byte vocabulary of size B. For each byte of index j $(0 \leq j < B)$, we associate a byte vector embedding v_j of dimension d. Any input word is first represented as the sequence of byte-indices $(j_1, j_2, ..., j_L)$ and then as the vector x formed by the concatenation of the byte embeddings: $x^\intercal = [v_{j_1}^\intercal, v_{j_2}^\intercal, ..., v_{j_L}^\intercal]$, resulting in a vector of dimension $d \cdot L$ where L is the length of the word in bytes. A byte-$n$gram filter f has a weight vector $w^{(f)}$ of size $K = d \cdot n$ and a bias scalar $b^{(f)}$. The 1D convolution between the word vector x and a byte-$n$gram filter f is computed for all positions i $(0 \leq i < L - K)$ by

$$y_i^{(f)} = \sigma(\sum_{k=0}^{K} w_k^{(f)} \, x_{k+i \cdot d} + b^{(f)}) \quad (1)$$

where σ is an element-wise sigmoid function. Then the vector $y^{(f)}$ is reduced to a single scalar $h^{(f)}$ via sub-sampling over positions:

$$h^{(f)} = \max_{0 \leq i < L-K} y_i^{(f)} \quad (2)$$

These operations are carried out for all filters. Therefore, the output of the max-pooling layer over positions has a dimension that is equal to the number of filters. We discuss the technical details of these operations and the corresponding experiments in Sec. 4. For an efficient implementation with matrices, the byte-embedding is zero-padded to a fixed length set to the maximal length over all words seen in training.

3.3. Highway layer for adaptive feature combination

Highway layers on top of convolution and pooling were reported to be a key component for improvements in [8].

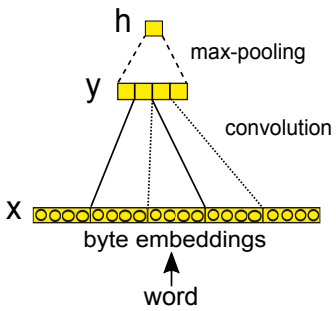

Fig. 2. *Illustration of byte-convolution for a single filter. In this example, the length of the word is 5 bytes (L=5), each byte embedding has a dimension 4 (d=4) and the filter covers byte-bigrams (n=2, K=8). Pooling is carried out over all positions within one word.*

Though the highway network is originally designed for training deep networks, it can also be used as a layer which adaptively interpolates the features across layers. For further details, we refer to [8, 14, 15]. In this work, this is only used in Sec. 4.6.

Table 1. *Statistics of training and testing corpora.*

ID Language	Set				Lexicon size
	Training		Dev		
	amount of speech [h]	running words		OOV [%]	
104 Pashto	37.2	452k	108k	2.3	14k
306 Igbo	40.3	460k	110k	3.8	17k
307 Amharic	40.2	295k	65k	11.7	35k
403 Dholuo	41.4	372k	84k	5.3	18k

4. TEXT-BASED EXPERIMENTS

In this section, we discuss the technical details of the byte convolution. Experiments are carried out on dev sets of 4 languages from the IARPA Babel Option Period 3 (OP3) data release: Pashto, Igbo, Amharic and Dholuo[1]. The corresponding descriptions and data statistics are presented in Table 1. The perplexities for the baseline single-layer LSTM language models can be found in the first row of Table 8. The number of LSTM units was optimized between 100 and 500 for each language. The number of word classes is set to 100 for all languages.

4.1. Bytes vs. characters

As mentioned in Sec. 2, processing bytes can be more attractive than characters for some languages. However, bytes do not define a proper language unit. We might still expect the CNN to discover patterns in the byte sequence, though this needs to be verified experimentally. In this work, we present experiments on Pashto and Amharic.

[1]Official data set identifiers: IARPA-babel104b-v0.4bY, IARPA-babel306b-v2.0c, IARPA-babel307b-v1.0b, IARPA-babel403b-v1.0b

In the Pashto alphabet each of the 47 characters is encoded with two bytes. Therefore, by mapping Pashto characters to 1-byte coded ASCII characters, we can evaluate a character-level CNN and compare it to the byte-level model. The corresponding byte-level CNN has the byte set size of 48. In both cases we use 15-dimensional embeddings. The perplexity results are shown in Table 2. Both character and byte model outperform the word-level baseline with the byte-level model having a slightly better perplexity. The optimal configuration is obtained with filters of size 3 for character and 6 for bytes (which also corresponds to 3 characters).

Table 2. *Bytes vs. characters. Perplexities for **Pashto**. Perplexity of the baseline LSTM: 130.5*

	Character			Byte		
Filter Size	Number of filters			Number of filters		
	500	1000	2000	500	1000	2000
3	126.7	125.7	**125.3**	127.9	127.5	127.0
6	126.2	126.2	126.0	125.6	124.9	**124.6**
12	126.6	126.1	126.2	125.6	124.7	124.8

It is not straight-forward to perform the same text preprocessing on Amharic, since there are 282 characters (consonant+vowel pairs) which can not be encoded by one byte. Therefore, to work on character-level we had to explicitly code that each Amharic character is 3-byte long. The results are shown in Table 3. Again, there is no degradation in working with raw Unicode bytes instead of characters.

For Igbo and Dholuo, all characters in the transcripts provided in the Babel project are encoded with one byte, so there is no difference in convolving over bytes or characters.

Table 3. *Bytes vs. characters. Perplexities for **Amharic**. Perplexity of the baseline LSTM: 202.7*

	Character			Byte		
Filter Size	Number of filters			Number of filters		
	500	1000	2000	500	1000	2000
3	193.8	193.4	194.8	-	-	-
6	194.6	193.7	**192.8**	193.1	192.5	**191.5**
9	194.4	195.1	195.3	194.4	192.1	194.0
12	197.4	197.7	198.5	194.0	194.7	195.2

4.2. Filter size

In [8] the authors used filters of different sizes. In our experiments, the perplexity results in Table 4 indicate that it is sufficient to set the size of all filters to a single value.

Table 4. *Comparison of using filters of mixed sizes vs. single size. Perplexity results on **Igbo**. Perplexity of the baseline LSTM: 103.4*

		Number of filters		
	Filter size	500	1000	2000
Mixed	2 - 8	96.9	96.1	**95.7**
Single	3	96.6	95.6	**95.5**
	8	96.4	96.4	96.0
	16	98.7	98.9	98.9

4.3. Dimension of byte embedding

We also compare different sizes of embeddings associated with each byte. The perplexity results are shown in Table 5. For both Igbo and Pashto, we found that higher dimensions work better. In addition, we found that the number of filters can be reduced if byte embedding dimension is higher.

Table 5. *Perplexities for different byte embedding size. Filter size is 3 for Igbo and 6 for Pashto.*

		Number of filters		
	Dim.	500	1000	2000
Igbo	15	96.6	95.6	**95.5**
	30	96.2	**94.8**	95.1
Pashto	15	125.6	124.9	**124.6**
	30	124.7	**124.5**	125.1

4.4. Pooling: max over positions vs. max over filters

The model described in Sec. 3.2 is a direct application of the convolution and pooling to text, which has been proven useful in other fields. It is therefore important to discuss the meaning of such operations for language processing. By *pooling over positions*, the CNN finds the byte-ngram patterns which appear across different words, but totally ignores the position within the word at which the pattern occurs. The translation-invariance might be a useful property for some sub-word units, but for others, especially longer words, the position might matter. Therefore, it might make sense to consider a pooling operation which preserves information on the absolute position of the byte-level patterns. With this question in mind, we evaluate a pooling which sub-samples its input by taking the *maximum over all filters* at each position in the word. The results for Igbo are shown in Table 6. The reason for the bad perplexity might be the fact that by pooling over filters only one filter per position is used and all others do not get any feedback. For our future work, we plan to investigate a combination of these two pooling strategies.

Table 6. *Effect of pooling type. Perplexity results on **Igbo***

Pooling type	Size	Number of filters		
		500	1000	2000
Over positions	3	96.6	95.6	**95.5**
Over filters	3	140.2	**130.8**	139.8
	6	**127.9**	128.4	137.0

4.5. Do improvements come from the CNN?

In order to verify whether the improvements come from the CNN architecture and not from a larger word feature or an additional non-linearity, we compare the convolutional layer to a larger linear and sigmoid layer (up to 2000 nodes). The results in Table 7 show that even though a larger input layer leads to better perplexities, it does not reach the CNN.

Table 7. *Effect of first layer type. PPLs for* ***Igbo*** *and* ***Pashto***

	PPL	
Input layer type	Igbo	Pashto
Baseline linear	103.4	130.5
Large linear	99.0	127.7
Large sigmoid	98.9	127.6
Convolutional	**94.8**	**124.5**

4.6. Effect of highway layer

The effect of adding a highway layer is shown in Table 8. In our experiments, we found that its impact on the perplexity is marginal if the filters of CNN are well configured.

Table 8. *Effect of LM topology on perplexity*

LM topology	PPL			
	Pashto	Igbo	Amharic	Dholuo
Baseline LSTM	130.5	103.4	202.7	144.8
+ CNN	**124.5**	**94.8**	**191.5**	136.9
+ Highway	125.2	95.9	194.4	**135.8**

5. ASR AND KEYWORD SEARCH EXPERIMENTS

5.1. Acoustic modeling setups

For acoustic modeling, we trained bidirectional LSTM neural networks with 3 hidden layers using the `RETURNN` toolkit [16]. There are 500 LSTM units in the hidden layers and the output layer models generalized triphone state posterior probabilities for 4500 CART labels. The input is a 115-dimensional CMLLR transformed concatenation of Gammatone, pitch, voicedness and multilingual bottleneck features [17]. The multilingual bottleneck feature extractor is trained on 28 languages (1793 hours of audio) and fine-tuned to the target language. The final acoustic models are sequence discriminatively trained according to the MPE criterion.

The keyword search (KWS) performance is measured in *maximum term weighted value* (MTWV), an accuracy measure for keyword spotting that uses a single global decision threshold optimized on the dev set [18]. The higher the value, the better.

According to the evaluation condition of the final Babel project period, the pronunciation lexica are derived automatically from text, assuming a one-to-one mapping from characters to phones. The corpus statistics are shown in Table 1.

5.2. Language modeling and lattice rescoring setups

Our baseline systems for speech recognition and keyword search are based on Kneser-Ney smoothed bigram count models. In fact, Knill et al. [19] has shown that the bigram lattice topology offers a better, more diverse search space for keyword search purposes. Therefore, our goal is to rescore lattices with an RNN LM to replace the LM scores while preserving the bigram lattice structure. For that we rescore lattices with the *replacement approximation* suggested by Sundermeyer et al. [20]: for each arc in the original lattice, the new LM scores are taken from the best path in the traceback tree which contains the corresponding arc. For efficiency reasons we pre-compute and store the CNN-based representations in a word-indexed lookup table. The training of all neural language models and rescoring was done using the toolkit `rwthlm` [21].

5.3. Results

First of all, we report the effect of standard LSTM rescoring for low resource ASR and keyword search. The LSTM LM were trained on the audio transcripts (cf. Table 1). The ASR results obtained with CNNs are shown in Table 9. The byte-level CNNs slightly improve the ASR performance on Igbo, Dholuo and Amharic. For Igbo and Dholuo, KWS was also carried out. Improvements in KWS are shown in Table 10.

Table 9. *Word error rate results.*

ID	Language	WER [%]		
		2gr	+LSTM	+CNN
104	Pashto	47.4	45.6	**45.6**
306	Igbo	56.8	56.0	**55.9**
307	Amharic	42.8	42.1	**42.0**
403	Dholuo	38.1	37.0	**36.9**

Table 10. *Effect of rescoring for keyword search performance*

ID	Language	MTWV		
		2gr	+LSTM	+CNN
306	Igbo	0.3759	0.3733	**0.3801**
403	Dholuo	0.6228	0.6245	**0.6253**

6. CONCLUSION AND FUTURE WORK

In this paper, we extended the notion of convolution on character level to the more simple setup based on raw Unicode bytes. This approach slightly reduced the perplexity and allowed to improve both the word error rate and the keyword search accuracy on several difficult low-resource tasks. We investigated the choice of parameters for the convolutional layer (number and size of filters) and pooling operation (pooling over position vs. pooling over filters). We found that a highway layer did not consistently improve the perplexity, and we were able to show that the convolution operation is indeed more important than just a larger input layer. In the future, we plan to further investigate different pooling strategies and consider to predict bytes instead of words.

7. REFERENCES

[1] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. Int. Conf. on Machine Learning (ICML)*, New York City, NY, USA, Jun. 2016, pp. 1747–1756.

[2] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR," in *Proc. Interspeech*, Dresden, Germany, Sep. 2015, pp. 26–30.

[3] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 890–894.

[4] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2015, pp. 649–657.

[7] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," in *Proc. of Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, Sep. 2015, pp. 1520–1530.

[8] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Proc. AAAI Conference on Artificial Intelligence*, Phoenix, AZ, USA, Feb. 2016, pp. 2741–2749.

[9] W. Ling, "Machine translation 4 microblogs," Ph.D. dissertation, Carnegie Mellon University, 2015.

[10] M. R. Costa-Jussà and J. A. Fonollosa, "Character-based neural machine translation," in *Proc. Association for Computational Linguistics (ACL)*, Berlin, Germany, Aug. 2016, pp. 357–361.

[11] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[12] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, "Multilingual language processing from bytes," in *Proc. North American Chap. of the Assoc. for Comput. Ling. on Human Lang. Tech. (NAACL-HLT)*, San Diego, CA, USA, June 2016, pp. 1296–1306.

[13] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[14] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, Dec. 2015, pp. 2368–2376.

[15] K. Irie, Z. Tüske, T. Alkhouli, R. Schlüter, and H. Ney, "LSTM, GRU, highway and a bit of attention: An empirical overview for language modeling in speech recognition," in *Proc. Interspeech*, San Francisco, CA, USA, Sep. 2016, pp. 3519–3523.

[16] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, and H. Ney, "RETURNN: the RWTH extensible training framework for universal recurrent neural networks," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, 2017.

[17] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Multilingual features based keyword search for very low-resource languages," in *Interspeech*, Dresden, Germany, Sep. 2015, pp. 1260–1264.

[18] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proc. of ACM SIGIR Workshop on Searching Spontaneous Conversational Speech*, 2007, pp. 51–57.

[19] K. Knill, M. J. Gales, S. P. Rath, P. C. Woodland, C. Zhang, and S.-X. Zhang, "Investigation of multilingual deep neural networks for spoken term detection," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Olomouc, Czech Republic, Dec. 2013, pp. 138–143.

[20] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 661–665.

[21] M. Sundermeyer, R. Schlüter, and H. Ney, "rwthlm the RWTH Aachen University neural network language modeling toolkit," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 2093–2097.