Dilated Convolutional Neural Networks for Lightweight Diacritics Restoration

Bálint Csanády **András Lukács**

AI Research Group, Institute of Mathematics, Eötvös Loránd University

csbalint@protonmail.ch, andras.lukacs@ttk.elte.hu

Highlights

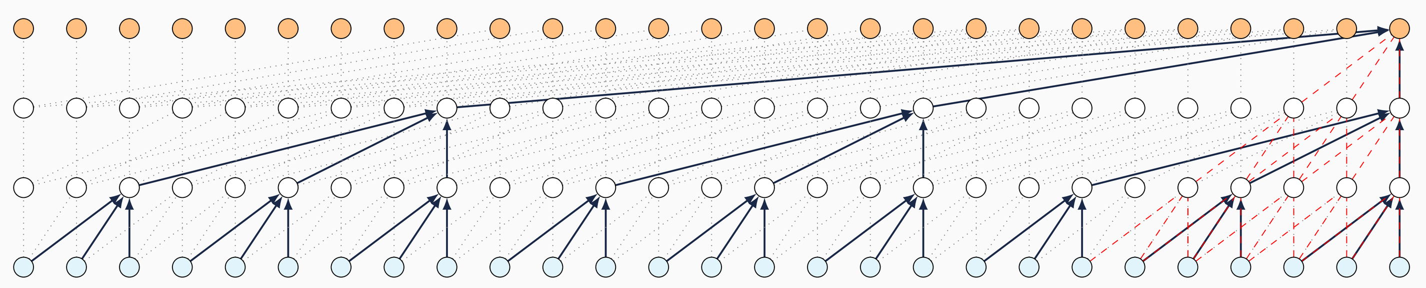
- **Diacritics restoration** is a ubiquitous task in NLP and on the Internet.
- We describe a **small footprint approach**, using a neural network (A-TCN) which operates at a character-level and is based on 1D dilated convolutions.
- Our solution surpasses the performance of similarly sized models and is also competitive with larger models.
- A feature of our solution is that it **runs locally in a browser**.
- Our model was evaluated on multiple corpora.
- We analyzed the errors to understand the limitation of the **self-supervised** training.
- We provide links to our online demo and our source code.

Diacritics Restoration

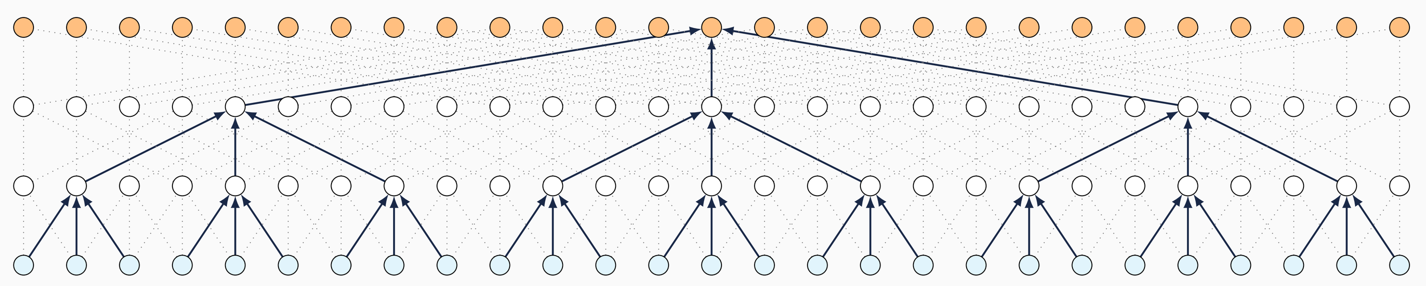
- Many languages derive some of the characters in their alphabet from a base alphabet (such as the Latin alphabet) using **diacritical marks**.
- In Hungarian for example all of the vowels can receive diacritical marks.
- The **goal** of diacritics restoration is to restore these marks given an input text without the proper marks.
- Ambiguity: *koros* $\mapsto$ {*körös, kóros, kórós, koros, kőrös*}.

Temporal Convolutional Networks (TCN)

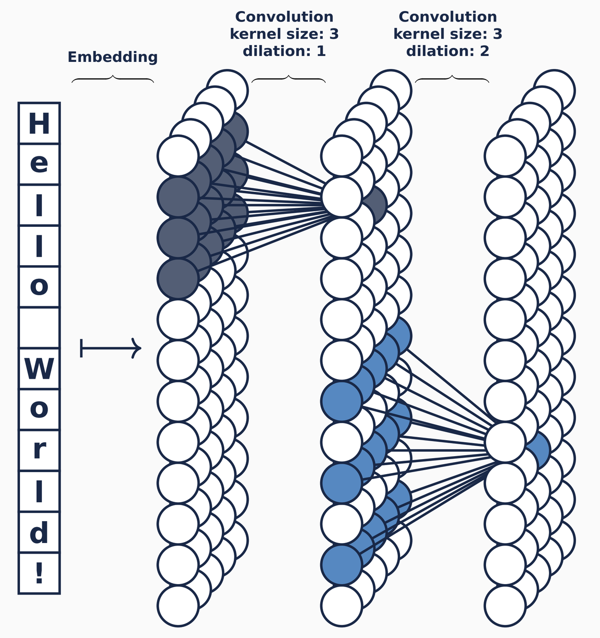
- TCN is a neural network architecture based on **1D convolutions**.
- The **dilation factor** of the convolutions increases exponentially by the depth of the network.
- The network is built of **residual blocks**.
- **A-TCN** (Acausal TCN) is an architecture similar to TCN, where information can flow form both temporal directions.



TCN architecture (dilation factors: 1,3,9). Red dashed: without dilation.



A-TCN architecture (kernel size: 3, dilation factors: 1,3,9).



- **Competitive** with LSTM-s and other recurrent architectures.
- Fast growing receptive field by network depth.
- No hard limit on input length.
- Runs well in the browser.

Datasets

- We used the datasets provided by Náplava et al. (LINDAT), for training on **Czech**, **Hungarian**, **Polish** and **Slovak**.
- We also trained on a dataset built from **Hungarian Webcorpus 2.0** (HunWeb2).
- Further evaluation: the earlier Hungarian Webcorpus
- To augment the data instead of removing all diacritic marks, we kept a certain percentage in each epoch.

	Train			Dev		
Language	Sequences	Avg.seq.len.	Characters	Sequences	Avg.seq.len.	Characters
Cze	946 k	107.6	101.8 M	14.5 k	114.4	1.66 M
Hun	1287 k	108.3	139.3 M	14.7 k	120.7	1.77 M
Pol	1063 k	116.2	123.6 M	14.8 k	121.3	1.80 M
Svk	609 k	106.7	65.1 M	14.9 k	114.7	1.71 M

Statistics of the LINDAT datasets.

Dataset		Seqs	Avg. seq. len.	Chars
HunWeb1	Dev	10 k	409.3	4.09 M
HunWeb2	Train	6.16 M	474.0	2.92 G
	Dev	10 k	474.1	4.74 M

Statistics of the Hungarian datasets.

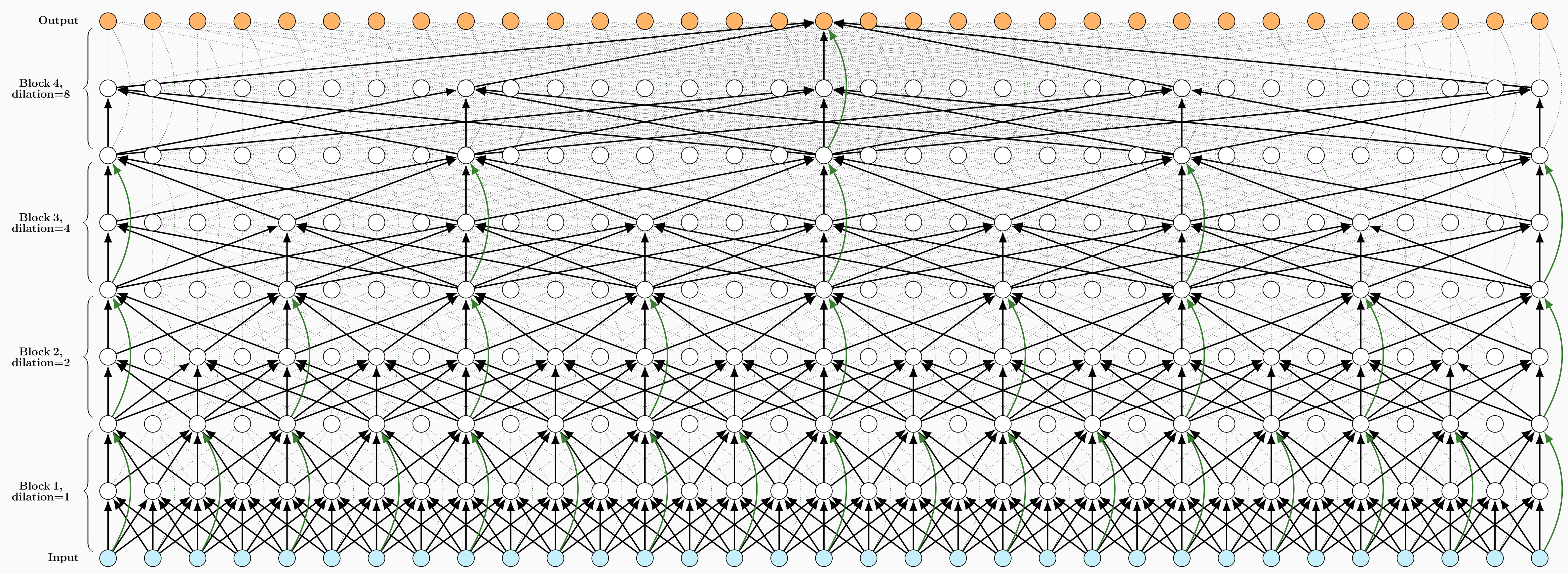
			Unambiguous		Ambiguous		Ratio	
Corpus	Sequences	Words	Words	Bases	Words	Bases	Words	Bases
HunWeb1	649 k	35.7 M	18.2 M	979 k	17.6 M	29.3 k	1.032	33.5
HunWeb2	6.16 M	403.0 M	118.6 M	4.51 M	284.4 M	179.2 k	0.417	25.2

Word ambiguity statistics of the Webcorpus-based datasets for Hungarian.

Results

Lang.	Char.	Relevant Char.	α-word	Seq.
Cze	0.9966	0.9944	0.9783	0.7344
Hun	0.9975	0.9925	0.9824	0.7890
Pol	0.9987	0.9970	0.9903	0.8810
Svk	0.9966	0.9947	0.9784	0.7420

Architecture



Error Analysis

Error class	Ratio
1. Corpus error	0.062
2. Corrected corpus error	0.128
3. Word Ambiguous Input	0.186
4. Grammar Ambiguous Input	0.158
5. Context Ambiguous Input	0.124
6. Named Entity	0.256
7. Incorrect Output	0.126

Results for Hungarian Language

- We built a **dictionary** from the words in the HunWeb2 training dataset. The dictionary JSON (uncompressed) is about 170 MB.
- **Hunaccent** was also considered as a baseline since it is also a lightweight model which runs in a browser (hunaccent.js is ~ 12 MB).
- Náplava et al. reports an alpha word accuracy of 0.9902 on Hungarian (LINDAT) with an LSTM-based solution of around 30 MB.

Model	Model size	Train data	Eval data	Character	Vowel	Alpha-word	Sequence
Copy			HunWeb1	0.8979	0.6929	0.4768	0.0000
			HunWeb2	0.9020	0.7042	0.4997	0.0000
			LINDAT	0.9043	0.7134	0.5093	0.0269
Hunaccent	12 MB	HunWeb1	HunWeb1	0.9886	0.9657	0.9207	0.0398
			HunWeb2	0.9855	0.9563	0.9049	0.0087
			LINDAT	0.9834	0.9509	0.8934	0.2732
Dictionary	170 MB	HunWeb2	HunWeb1	0.9960	0.9879	0.9772	0.3511
			HunWeb2	0.9965	0.9894	0.9791	0.3329
			LINDAT	0.9942	0.9831	0.9698	0.6551
A-TCN	13.5 MB	HunWeb2	HunWeb1	**0.9987**	**0.9961**	**0.9907**	**0.6574**
			HunWeb2	**0.9988**	**0.9964**	**0.9916**	**0.6424**
			LINDAT	0.9974	**0.9941**	**0.9862**	**0.8087**
A-TCN	13.5 MB	LINDAT	HunWeb1	0.9950	0.9850	0.9649	0.2683
			HunWeb2	0.9945	0.9834	0.9621	0.1556
			LINDAT	**0.9975**	0.9925	0.9824	0.7890

Accuracy comparison for Hungarian diacritics restoration between the baseline (Hunaccent) and our model (A-TCN).
The numbers indicate the results on non-augmented, fully dediacritized input.

Online Demo with ONNX Runtime Web

- Available at: https://web.cs.elte.hu/~csbalint/diacritics/demo.html?lang=en
- Client-side inference: the model runs locally in the browser.
- The whole application is a single html file (~ 13.5 MB).
- Pytorch $\mapsto$ ONNX $\mapsto$ ONNX Runtime Web.

```
Arvizturo tukorfurogep
Csuszdazo mubor kulonitmeny
Haztuznezougynok-busito
Joizu felaru sutotok
Jott arviz, tuzvesz, rut gumokor.
Kover fulu siturazo no
Kulonallo muutepito
Nyulfulvago terkozsurito
Sos hust sutsz tan, vizkopo Szucsne?
Tobb hutohazbol kertunk szinhust.
Tiz budos legy husz mucsotanyt foz.
```

$\Rightarrow$

```
Árvíztűrő tükörfúrógép
Csúszdázó műbőr különítmény
Háztűznézőügynök-busító
Jóízű félárú sütőtök
Jött árvíz, tűzvész, rút gümőkór.
Kövér fülű sítúrázó nő
Különálló műútépítő
Nyúlfülvágó térközsűrítő
Sós húst sütsz tán, vízköpő Szűcsné?
Több hűtőházból kértünk színhúst.
Tíz büdös légy húsz műcsótányt főz.
```

Further Goals

- More general spell-correcting.
- Train a larger, but still browser-compatible model.
- Consider more diacritics-heavy languages.
- Clean up the corpora with the help of the model.
- Apply the architecture on other, possibly non-NLP tasks.
- Gain more insight in the architecture, to optimize the hyperparameters.

Github repo

Acknowledgments

The research was partially supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program, the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008 and the Thematic Excellence Programme TKP2021-NKTA-62.
The second author was supported by project "Application Domain Specific Highly Reliable IT Solutions" implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.
We would like to thank Dániel Varga for drawing our attention to the problem of lightweight diacritics reconstruction, Judit Ács for her help with NLP issues, and neptune.ai for providing a convenient platform to log our experiments.