Hardware-aware Workload Distribution for AI-based Online Handwriting Recognition in a Sensor Pen

Fabian Kreß*, Alexey Serdyuk*†, Tim Hotfilter*,
Julian Hoefer*, Tanja Harbaum*, Jürgen Becker*
* *Karlsruhe Institute of Technology*
Karlsruhe, Germany
fabian.kress@kit.edu

Tim Hamann†
† *STABILO International GmbH*
Heroldsberg, Germany
tim.hamann@stabilo.com

***Abstract*—Time series-based applications such as recognition of handwriting benefit from using Deep Neural Networks (DNNs) in terms of accuracy and efficiency. Due to strict power and memory limitations of embedded platforms in the Internet-of-Things (IoT), the inference of such DNNs is usually performed on more powerful and less constrained devices. However, inference on mobile devices such as smartphones or tablets leads to high system requirements. In this paper, we present our approach for distributing computational workload between sensor pen and a mobile device for handwriting recognition. This not only results in lower system requirements for mobile devices, but also opens up the possibility of storing compressed sensor data on the pen when remote devices are not available or connected.**

Therefore, we first quantize weights of the DNN layers to significantly reduce the memory footprint and optimize a Connectionist Temporal Classification (CTC) algorithm to minimize the link bandwidth. The latter allows transmitting only a small matrix from the pen to the mobile device, on which the decoding is performed. To further improve the accuracy of handwriting recognition, we add a Language Model (LM) to the decoding algorithm executed on the mobile device.

As a result, we are able to reduce the memory footprint of the DNN weights by almost a quarter to 736.64 KB. Additionally, applying CTC-based beam search and an LM, we can reduce the link bandwidth by almost 80%. Based on the RISC-V Instruction Set Architecture (ISA), we finally define memory and performance requirements to enable in-pen DNN inference.

***Index Terms*—Embedded Systems, Handwriting Recognition, Recurrent Neural Network, Internet of Things**

I. Introduction

Studies have shown that handwriting significantly improves learning outcomes than just typing on a keyboard [1]. Moreover, handwritten work enhances creativity in problem-solving, leading to better concepts and results [2]. Mobile devices offer powerful tools for sharing or restructuring content, however, this is not available with handwritten notes on paper. As a result, sensor pens such as the STABILO Digipen [3] have already been developed. These combine the best of both worlds by recording user motion and transmitting sensor data to a mobile device over Bluetooth Low Energy (BLE). An overview of the system architecture used in this paper is shown in Figure 1.

Reliable online handwriting recognition using Deep Neural Networks (DNNs) in combination with Connectionist Temporal Classification (CTC) decoding is the key to a high usability and thus high acceptance. Inference of the DNNs requires processing a large amount of instructions, which is currently done on a connected mobile device exclusively. However, this leads to high performance requirements as well as uncertainties regarding latency. Furthermore, using the pen without active connection to the mobile device to store written text before sending is not possible due to limited memory resources. Consequently, executing the DNN on the pen itself is favorable, since it offers static latency that allows it to be used by a larger number of mobile devices, and since it is more independent of the BLE connection to the mobile device.

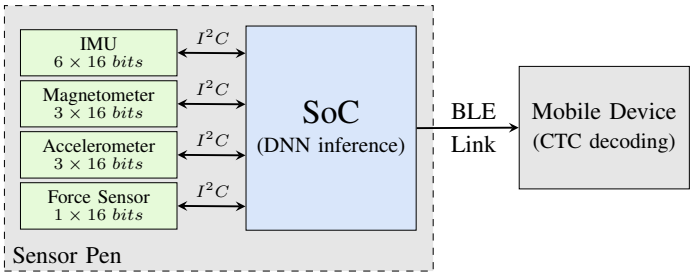

Fig. 1: Overview of the handwriting recognition system consisting of several sensors. The data of each sensor is transferred via I^2C to the central SoC which executes DNN inference and sends results over BLE to a mobile device for CTC decoding.

Pre-trained DNNs have to be optimized towards system requirements e.g. by quantization to reduce the memory footprint [4]. In addition, it is favorable to select a hardware design that offers low power consumption to ensure sufficient uptime as well as a reasonable inference latency and enough memory. DNN inference on Internet-of-Things (IoT) devices has already been addressed in numerous research. Bluche et al. presented a quantized Long Short-Term Memory (LSTM) network using CTC-decoding for keyword spotting on microcontrollers [5]. Their network offers a small memory footprint, however, they did not investigate the latency, which is important in the scope of online handwriting recognition. Gao et al. showed an accelerator for Recurrent Neural Networks (RNNs) for use in IoT computing platforms [6]. The LSTM offers low latency and a suitable energy consumption for time series-based applications. However, the accelerator is highly optimized towards RNNs. Since Convolutional Neural Networks (CNNs) achieve better results in online handwriting recognition than pure RNNs

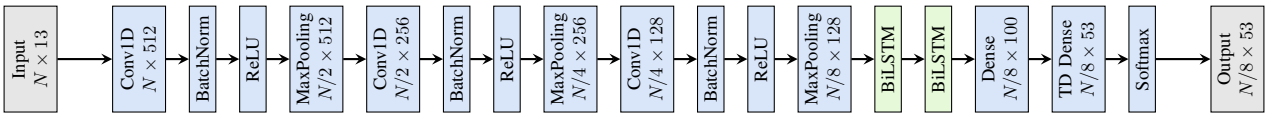

Fig. 2: Overview of the CNN + BiLSTM neural network inference layers used for handwriting recognition [7]. The structure consists of common CNN layers (1D Convolution, Batch Normalization, ReLU activation and 1D Max Pooling), followed by two bidirectional LSTMs, a Dense, a Time Distributed (TD) Dense layer and finally a Softmax function layer.

according to Wehbi et al. [7], the accelerator is not suitable for handwriting recognition applications. To the best of our knowledge, no work is addressing DNN inference for online handwriting recognition so far.

In summary, our contributions in this paper are as follows:

- We present our basic system setup and the DNN structure for fast and reliable online handwriting recognition in an IoT system.
- We apply weight quantization to the DNN architecture to meet hardware constraints.
- We add a Language Model (LM) to the beam search decoding on the mobile device to improve the Character Error Rate (CER) which is a highly relevant metric.
- We evaluate our results of the optimized DNN and define hardware requirements for memory size and performance.

II. Sensor System

Handwriting recognition based on the movement of a pen requires measuring multiple physical quantities such as acceleration and rotation. In addition, the system must detect whether the pen is in contact with a surface or not. The STABILO Digipen, a sensor-enhanced ballpoint pen which serves as our reference platform. It is equipped with an Inertial Measurement Unit (IMU), magnetometer and force sensor in the front and an additional accelerometer in the back [3]. Since the force sensor and all three dimensions of the other sensors have to be combined for reliable handwriting recognition, we have a total of 13 sensor inputs for DNN inference. These are acquired with a sampling frequency of 100 Hz and represented as 16-bit integer values.

A. Hardware Platform

Currently, a commercially available BLE System-on-Chip (SoC) with a general-purpose ARM Cortex M4F unit is integrated into the Digipen, but it is severely limited in terms of energy efficiency and available memory size. In particular, a substantial part of the available 512 KB Flash memory is already reserved for the BLE software stack. To enable DNN inference on the pen itself, we propose a holistic hardware/software co-design approach based on the system architecture shown in Figure 1. Our optimization is not limited to the DNN architecture, but also takes the hardware platform deployed in the pen into account. Thereby, the BLE interface has to be considered as well since it impacts the overall energy consumption and the latency of the system.

In general, IoT applications require taking several constraints into account for enabling DNN inference. Memory and latency requirements in particular are crucial for embedded applications such as handwriting recognition in a pen. Consequently, optimization of the DNN also has to be done on Instruction Set Architecture (ISA) and hardware level.

B. Neural Architecture

In this work, we use an LSTM for online handwriting recognition with CTC decoding on the mobile device from Wehbi et al. (Figure 2) [7]. The output of the model, also referred to as CTC matrix, consists of 53 label probabilities, which represent 26 uppercase, 26 lowercase latin letters and one for a special CTC blank character ('-'). One or more such matrices correspond to a single character. The special blank character is used to differentiate between several occurrences of the same character, e.g. 'aaa' $\rightarrow$ 'a' whereas 'aa-a' $\rightarrow$ 'aa'. In order to obtain the final prediction, a CTC decoding algorithm [8] has to be applied. In our work, we use best path and beam search algorithms with and without LM. The former algorithm just uses one single predicted class at each timestep from the CTC matrix with the maximum appearance probability. This approach strongly limits the size of the data needed to be transmitted, but does not consider alternative paths in decoding.

The beam search algorithm evaluates several possible CTC paths (beams). The number of evaluated paths is defined by the beam width parameter. The basic variant, also known as vanilla, does not use a character level LM. As an LM we use an n-gram character level LM [9]. Such a model represents a probability of occurrence of a single character after a sequence of n characters. In the beam search decoding, LMs are used to weight beams at every timestep, which allows to incorporate language characteristics to select more likely character combinations. The impact of the LM on the beam search may be scaled by the LM factor.

We apply post-training weight quantization to optimize memory footprint of the model for the pen hardware. Weights of all layers of the model were quantized to 8-bit integer values. Activation functions, intermediate feature maps, model input and output are coded in 32-bit float values.

C. BLE Link

The link is a crucial part of the whole system, since it transfers sensor data to the mobile device. BLE allows supporting a wide range of mobile devices and is a low-power communication protocol. According to Tosi et al. [10], the maximum BLE throughput in bit error free environment is 236.7 $kbit/s$. But the BLE link is constrained in available bandwidth, which in turn depends on the BLE stack constraints of the mobile device. Moreover, wireless communication can be subject to transmission errors, hence, retransmissions have to considered. Therefore, the amount of data to be transferred should be minimized.

III. Evaluation

Our model is trained using Keras and TensorFlow. The training dataset consists of 49,877 labeled sequences, which represent German words without umlauts. The training was performed for 30 epochs, with CTC loss and an Adam optimizer at a learning rate of 0.01. In total, the network consists of 731,853 parameters. As a prediction accuracy metric CER is used and evaluated on the test data set of 756 words. CER is calculated as the sum of Levenshtein distances between predicted and expected words divided by the number of all characters in the test data set. For evaluation, a 20-gram character LM is used, generated from a publicly available corpus of one million sentences from German news articles during the year 2015 without umlaut characters [11].

A. Architecture Optimization

1) Bandwidth Optimization: All sensor channels are acquired as 16-bit integer values with a sampling frequency of 100 Hz. The required link bandwidth for the transmission or the memory bandwidth for the local storage of the entire raw sensor data is thus 20.31 $kbit/s$. Due to the three max pooling layers applied over time dimension, transmissions of CTC matrices occur after every 8th input sample resulting in an output CTC-timestep rate of 12.5 Hz and thus a required bandwidth of 20.70 $kbit/s$. Hence, running inference on the sensor pen device without applying any further optimization is not beneficial. Depending on the CTC decoding algorithm, the amount of predicted classes can be reduced. For the best path algorithm, which has been used by Wehbi et al. [7], only one predicted class has to be transmitted. With this approach, which serves as a reference for our evaluation, a CER of 15.70% can be achieved and just a bandwidth of 0.39 $kbit/s$ is required.

With beam search, higher prediction accuracy can be achieved [8], but it requires more predicted classes from the output to be transmitted. Figure 3 shows the evaluation of the prediction accuracy depending on beam width for the vanilla beam search and for beam search with the LM using different factors. We evaluate the accuracy for beam widths from 1 up to 30. We observe that CERs for beam widths greater than 30 remain nearly constant, and therefore all values above 30 are ignored in our evaluation. Vanilla beam search does not significantly improve the prediction accuracy at any beam width value, whereas the LM improves its prediction accuracy starting from a beam width of four. The best prediction accuracy can be achieved with a beam width of 22 and a LM-factor = 2.0 (CER = 11.19%). However, using only eleven beams and a LM-factor = 1.0 reduces the required bandwidth by two times without significant accuracy loss (CER = 11.90%). This results in a reasonable trade-off between bandwidth and prediction accuracy. Required bandwidth in case of eleven beams is 4.30 $kbit/s$.

2) Model Size Optimization: To minimize the memory footprint of the model, we quantize weights of the model to 8-bit. Figure 4 shows a comparison of the original and quantized model. Due to slight overfitting in the original model, the prediction accuracy of the vanilla beam search improves by 0.12 percentage points on average. Using LM and LM-factor 1.0, the prediction accuracy degrades by only 0.53 percentage points. The size of the quantized model is 0.739 Mbytes, which is 3.7 times less than the original model size of 2.797 Mbytes.

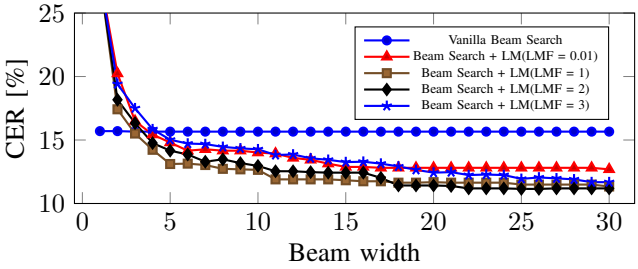

Fig. 3: Comparison of CER between vanilla beam search and CTC Beam Search accompanied by language model. The results show that vanilla beam search does not benefit from using more than a single input. In contrast, CTC Beam Search with language model performs better for 5 beams.

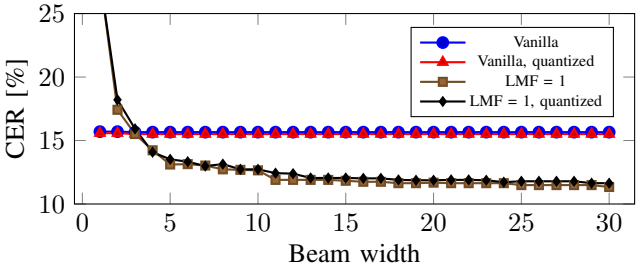

Fig. 4: Comparison of CER between original model and quantized model accompanied by language model. The results show that accuracy of the quantized model degrade slightly.

B. System-on-Chip Platform Requirements

A critical aspect for hardware implementations is the memory footprint of the DNN application since memory consumes a lot of area. Apart from the weights, intermediate feature maps, the LSTM cell states and the DNN program code have to be stored on the SoC. Since we assume sequential processing of the DNN, the largest layer input and output feature map define the memory requirement. In our case, the second 1D convolution layer takes an $N/2 \times 512$ tensor as input and outputs $N/2 \times 256$ values. The minimum number of time steps N required to generate a valid CTC matrix is 24, due to three 1D max pooling with a sliding window size of two and 1D convolution layers with kernel sizes five, three and three. Consequently, to store 32-bit floating-point feature maps, 36,864 bytes must be available. In addition, a total of 131,072 bytes of memory is required for the states of the two bidirectional LSTM layers, each with 64 cells and 128 inputs.

The size of the DNN program code depends on the used ISA. In the following, we use RISC-V ISA due to its extensibility, which easily allows the analysis of different hardware architectures. We evaluate the memory footprint of the weight quantized DNN using a RISC-V cross-compiler for RV32IMFC ISA offering not only base integer (I), integer multiplication and division (M) and compressed (C) instructions but also

TABLE I: DNN metrics for the ARMv7E-M and three different RISC-V ISAs including target hardware performance to achieve required latency below $80\ ms$.

	ARMv7E-M	**RV32IC**	**RV32IMC**	**RV32IMFC**
Program Size	105.28 KB	305.58 KB	302.86 KB	291.90 KB
Instruction Count	$0.73 \cdot 10^9$	$6.75 \cdot 10^9$	$1.29 \cdot 10^9$	$0.34 \cdot 10^9$
Performance Target	$>$ 9.13 GFLOPS	$>$ 84.38 GOPS	$>$ 16.13 GOPS	$>$ 4.25 GFLOPS

single-precision floating-point operations (F). Furthermore, we compare this with the DNN program size for the reference ARMv7E-M, for the RV32IC and the RV32IMC ISA. In the latter two cases, floating point instructions are implemented as soft-float operations. The results are shown in table I. Since the compiler was configured to optimize the code for size rather than latency, there is only a minor difference between the three RISC-V ISAs configurations. Compared to ARMv7E-M, however, these require significantly more memory, which is due to additional instruction set extensions in the ARM ISA. Adding up the aforementioned memory requirements including the weights, results in at least 1.2 MB of memory, which is necessary for deploying the DNN using RV32IMFC.

After reading sensor values for the first 24 time steps, the inference can be executed. Subsequently, only eight succeeding time steps are required to compute the next DNN output. Hence, to avoid data loss or having to buffer incoming sensor values, the maximum latency of the inference at a sampling frequency of $100\ Hz$ must be less than $80\ ms$. For defining the performance requirement of the hardware, we evaluate the number of instructions which have to be executed for the different ISAs using OVPsim [12] and Spike [13], respectively. Based on the results, we can also derive the minimal hardware performance. The performance requirement for each ISA is shown in table I. The results clearly show that using the RV32IC ISA leads to a very high number of instructions and is therefore not suitable for IoT applications. Even though RV32IMC requires fewer instructions, there is currently no state-of-the-art SoC available offering the required performance to meet our latency requirements. Similarly, there is currently no ARM Cortex-M4F-based system that provides sufficient on-chip memory for DNN inference. In contrast, according to the current state-of-the-art, there are already SoCs supporting RV32IMFC, which satisfy the performance target and memory requirements such as [14]. Consequently, since RISC-V is open source and can be extended with more efficient custom instructions, this ISA appears to be the most suitable for online handwriting recognition in an IoT device such as the Digipen.

IV. Conclusion

In this paper, we presented our hardware-aware DNN workload distribution approach for online handwriting recognition, which now enables executing inference in a sensor pen. Consequently, running DNN inference in the pen results in lower requirements for mobile devices and opens up the possibility of storing compressed sensor data on the pen itself when there is no remote device available. We reduce the memory footprint to almost a quarter of the original model size by quantizing the weights of the LSTM. Moreover, by introducing CTC decoding using a beam search algorithm and a character level Language Model (LM) executed in the mobile device, we were able to find a suitable trade-off between accuracy and required BLE bandwidth. Both metrics, memory footprint and BLE bandwidth, are crucial for deploying DNN in the sensor pen. Finally, we defined requirements for an SoC considering memory size and performance. In summary, the evaluation shows promising results for memory footprint, DNN accuracy and performance requirements. In the future, we plan to further optimize the DNN architecture considering energy constraints and to design an optimized SoC according to the requirements defined in this paper.

Acknowledgment

This work was funded by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS21097A (KIHT). The responsibility for the content of this publication lies with the authors.

References

[1] P. A. Mueller and D. M. Oppenheimer, "The pen is mightier than the keyboard: Advantages of longhand over laptop note taking," *Psychological Science*, vol. 25, no. 6, pp. 1159–1168, 2014, pMID: 24760141. [Online]. Available: https://doi.org/10.1177/0956797614524581

[2] S. Oviatt, A. Cohen, A. Miller, K. Hodge, and A. Mann, "The impact of interface affordances on human ideation, problem solving, and inferential reasoning," *ACM Trans. Comput.-Hum. Interact.*, vol. 19, no. 3, oct 2012. [Online]. Available: https://doi.org/10.1145/2362364.2362370

[3] STABILO International GmbH. The stabilo digipen. [Online]. Available: https://stabilodigital.com/

[4] I. Walter *et al.*, "Embedded face recognition for personalized services in the assistive robotics," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Cham: Springer International Publishing, 2021, pp. 339–350.

[5] T. Bluche, M. Primet, and T. Gisselbrecht, "Small-footprint open-vocabulary keyword spotting with quantized lstm networks," *arXiv preprint arXiv:2002.10851*, 2020.

[6] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Real-time speech recognition for iot purpose using a delta recurrent neural network accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[7] M. Wehbi, T. Hamann, J. Barth, P. Kaempf, D. Zanca, and B. Eskofier, "Towards an imu-based pen online handwriting recognizer," in *Document Analysis and Recognition – ICDAR 2021*, J. Lladós, D. Lopresti, and S. Uchida, Eds. Cham: Springer International Publishing, 2021, pp. 289–303.

[8] H. Scheidl. CTC Decoding Algorithms. [Online]. Available: https://github.com/githubharald/CTCDecoder

[9] D. Jurafsky and J. H. Martin, "N-gram language models," in *Speech and Language Processing*, 2021.

[10] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, "Performance evaluation of bluetooth low energy: A systematic review," *Sensors*, vol. 17, no. 12, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/12/2898

[11] D. Goldhahn, T. Eckart, and U. Quasthoff, "Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages," in *Proceedings of the 8th International Language Resources and Evaluation (LREC'12)*, 2012.

[12] Imperas Software Limited. OVPsim. [Online]. Available: https://ovpworld.org

[13] RISC-V foundation. Spike RISC-V ISA Simulator. [Online]. Available: https://github.com/riscv-software-src/riscv-isa-sim

[14] D. Rossi *et al.*, "Vega: A ten-core soc for iot endnodes with dnn acceleration and cognitive wake-up from mram-based state-retentive sleep mode," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, 2022.