Towards the on-device Handwriting Trajectory Reconstruction of the Sensor Enhanced Pen

Alexey Serdyuk, Fabian Kreß, Micha Hiegle, Tanja Harbaum, Jürgen Becker, Florent Imbert, Yann Soullard, Romain Tavenard, Eric Anquetil, Jens Barth, et al.

Towards the on-device Handwriting Trajectory Reconstruction of the Sensor Enhanced Pen

Alexey Serdyuk*§, Fabian Kreß*, Micha Hiegle*, Tanja Harbaum*, Jürgen Becker*
* *Karlsruhe Institute of Technology*
Karlsruhe, Germany
alexey.serdyuk@kit.edu

Florent Imbert†, Yann Soullard‡, Romain Tavenard‡, Eric Anquetil†
† *IRISA, Université de Rennes, INSA Rennes, France*
{florent.imbert,eric.anquetil}@irisa.fr,
‡ *IRISA, CNRS UMR 6554 LETG, Université Rennes 2, Rennes, France*
{romain.tavenard,yann.soullard}@univ-rennes2.fr

Jens Barth§, Peter Kämpf§
§ *STABILO International GmbH*
Heroldsberg, Germany
jens.barth@stabilo.com

***Abstract*—Performing handwriting trajectory regression from inertial data using Deep Neural Network (DNN) on an embedded device is a very challenging task, since the network accuracy is prone to imperfections in the weights and needs a significant amount of parameters to be able to regress. In this work, we apply and compare different quantization techniques and Mitchell logarithmic multiplication approximation in order to enable the on-device inference. We show that it is possible to perform the inference of the TCN-based regression model using only 8-bit fixed-point quantization without significant reconstruction precision loss and that the accuracy degradation of the approximate multiplication can be partially compensated with Quantization-aware Training (QAT). Finally, we demonstrate that the compressed models can be integrated into an off-the-shelf commercial Systems-on-Chip with minimal use of FPU and requiring only 460 KB of the ROM size for the TCN-49 configuration.**

***Index Terms*—Embedded Systems, Human Machine Interfaces, Handwriting Reconstruction, Temporal Convolutional Network, Internet of Things**

I. Introduction

Handwriting remains one of the essential means in school education, improving cognitive development of the children. Studies [1] and [2] have shown, that schoolchildren achieve better academic performance, when they use pen and paper for their class- and homework, enhancing their abilities to solve creative problems and developing their finemotoric skills. The handwriting is also one of the easiest ways to capture notes and ideas. Digitizing handwriting can enhance these activities since it allows for simple and fast reorganization of handwritten texts and drawings. Unfortunately, existing solutions require to use of a sensor pen along with special tools, e.g. cameras, and special papers that serve as external reference. As a result, these systems offer less flexibility and are more complex to be used. From other hand, the existing solutions without external reference, like STABILO Digipen [3] are based on the motion data from Inertial Measurement Unit (IMU), which is hard to transform into precise trajectory due to position drift caused by variety of error sources like sensor noise, measurement bias and bias instability, axis misalignment and others.

A promising approach to address this is the use of Deep Neural Networks (DNNs) for trajectory reconstructions as shown in [4] and [5]. Running DNN inference on the device enables autonomous use of the sensor pen in the setups, where mobile devices are not appropriate or not easily reachable. Moreover, possibility to reconstruct the trajectory on device using IMUs opens up a wide spectrum of alternative applications, such as human pose estimation, indoor navigation [6] and dead reckoning.

Performing the inference stage on the device is a challenging task and imposes hard constraints on the target system [7]. Special care should be taken to deploy such models onto device. To achieve this goal we apply different quantization techniques and approximate multiplication to studied models.

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

- We evaluate existing trajectory reconstruction networks with different quantization techniques and demonstrate, that it is possible to efficiently quantize the DNNs, trained on a regression task.
- We show the effects of the approximate multiplication on the quality of the trajectory reconstruction.
- We evaluate our results of the optimized DNNs and estimate memory footprint for the studied models.

II. Background

In the following section we will briefly recapitulate existing DNN-based techniques of the trajectory reconstruction using inertial sensors and existing neural network optimization techniques.

A. Trajectory reconstruction from inertial data

The challenge of trajectory reconstruction from inertial data appears in different application areas, such as indoor localization, human pose capturing and handwriting acquisition. Naive integration of measured acceleration leads to drastic positioning error accumulation due to Earth gravity vector, sensor fabrication imperfections and bias instability. Applying the analytic methods as different variations of the Kalman Filter is a quite common practice for inertial navigation of the large-scale objects [8]. It is possible to apply these techniques to the handwriting reconstruction, as shown in [9], however, to achieve this precision, special care has to be taken to calibrate the sensors, what in turn increases the cost of the end-device.

Another possibility to regress the trajectory from IMU is to apply Machine Learning (ML) approach. For example, [10] and [6] introduce several DNN architectures to reconstruct the trajectory of the worn mobile devices equipped with IMU for the indoor navigation and accompanying data sets. Both works show better performance for the Temporal Convolutional neural Network (TCN) based architectures than the recurrent DNNs. These works achieve good results for the indoor localization task, however the accuracy of the trajectory reconstruction is not enough for handwriting processing applications, since they require higher precision ground truth data.

The first network for the pen trajectory reconstruction was introduced in the [4] and it used the Convolutional Neural Network (CNN) architecture to perform the regression of displacement vectors, which were then accumulated into the final trajectory. The data set used in this work allowed to perform the model training with the ground truth, containing fine-grained trajectories of the actual handwriting. The reconstructions presented in this work look promising, but it the primary goal of the paper was character and word recognition, whereas trajectories were only used as a part of pipeline. Apart of that, the model was trained on a slightly older version of the sensor pen, which offered a sampling rate for the inertial data of only 100 Hz.

B. Temporal Convolutional Network

TCN is a class of feedforward networks which are based on dilated convolutions with residual blocks. First introduced in [11], TCNs proved to be state of the art option for time-series processing. The use of a TCN has the advantage to be faster to train and less prone to vanishing gradient than Recurrent Neural Networks (RNNs), especially in the case of long sequences [12]. Dilated convolutions in combination with stacking of several layers allows to get a wide receptive field, which in turn enables the learning of long standing temporal connections in the input data. The receptive field defines, how many values of the input are considered in order to produce a new output.

C. Model optimization using quantization and approximate computing

As a result of the tight constraints of the pen in terms of low energy consumption and limited amount of the available on-chip constant memory, the execution of DNN inference requires several optimizations such as quantization of parameters, activation functions and feature maps, as shown in a previous work [13].

Quantization of the model parameters is quite common technique used in order to deploy the DNNs onto resource constrained devices. Post Training Quantization (PTQ) is by definition applied to the trained model, decreasing the resolution of the model parameters, activation functions and operators to the specific bitwidth. It usually leads to round-off errors, which degrades the accuracy of the network. Therefore, a slightly different approach is to perform Quantization-aware Training (QAT), which runs the training of the model with quantized weights and activation functions. This approach allows to adapt the model to the approximated parameters and to get closer to the base line precision. However, the training loop of the quantized models is usually less efficient, since not all quantization schemes are natively implemented in the typical hardware used to train networks. Moreover, the quantized networks are prone to the loss oscillations, due to round-off errors. In the subsection III-C we will consider different existing techniques to mitigate the influence of these effects.

The forward pass of the TCN involves many multiplication operations, which consequently take up most of the time and energy required for inference. Processing multiplications in hardware is resource intensive, hence, logarithmic multipliers can be implemented to reduce energy consumption, as shown in [14]. The logarithmic multipliers allow to replace the multiplications with addition and shift operations, which significantly reduces complexity and allows for efficient implementation on a custom hardware or on platforms, without hardware floating point unit support. For example, [15] introduces a low power Mitchell multiplier, which was evaluated on the MNIST classification task using CNN-based model architecture and achieves 76.6% power consumption reduction compared to the full precision multipliers without significant accuracy loss. However, to our knowledge, there have been no studies on the influence of approximate multiplication on the accuracy of regression neural networks, which is addressed in this paper.

III. Pen Trajectory Reconstruction Optimization

In order to optimize and deploy the TCN-based trajectory reconstruction model, as with any data-driven approach, first the training data should be considered. In the following section we will describe data sets, used for model training and optimization. Then we introduce studied TCN-based models and introduce our approach to the hardware-aware model optimization.

A. Sensor Pen and Reference System

The STABILO Digipen [3] is equipped with an IMU sensor, Magnetometer, extra accelerometer in the rear part and the pen tip force sensor, which helps to segment the captured trajectory into separate strokes and also serves as an input channel for trajectory reconstruction. The sensor data is sampled at the frequency of 400 Hz and transmitted wireless to the user device. All raw sensor data channels are represented as 16-bit integer values.

To be able to train and evaluate trajectory reconstruction models, the raw data has to be matched with some reference trajectory. It is obtained using a modified version of the Digipen, which has an EMR-Wacom stylus instead of standard ball point ink insert, and a Wacom-enabled tablet, with reference data acquisition app. Recorded trajectory data consists of normal on-surface and the hovering strokes, when the pen tip is slightly lifted above the surface of the tablet. Wacom-based systems sample on-surface strokes with constant frequency, which allows to use them as robust labels. In contrast, the hovering strokes are sampled irregularly, requiring sophisticated preprocessing pipeline. Therefore, in this work we limit our

scope to training on on-surface strokes only. The full data acquisition setup is described in [4] and [5].

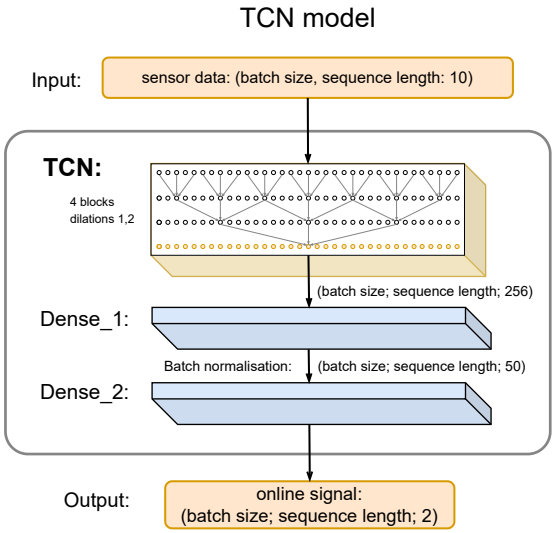

Fig. 1: TCN-49 model proposed in [5]

B. Neural Architecture

In a previous work [5], we designed a neural network architecture for the online handwriting trajectory reconstruction from the raw sensor data of the Digipen. The model is trained to predict the displacement vectors of the pen trajectory, which are integrated later into the trajectory. This approach allows to mitigate the biases of the absolute trajectories making it possible for the Neural Network (NN) to generalize. The network architecture is based on several blocks of a non-causal TCN layer followed by two dense layers. Table I lists the variations of the trained models and Figure 1 illustrates the network architecture of the TCN-49, where the number denotes the width of the models receptive field. The main goal of these networks is to reconstruct the shapes as close to ground truth as possible, ignoring the absolute scale and trivial rotations. The common used similarity measure for this goal is Fréchet distance [16], since it considers relative location and the ordering of the trajectory points. We apply this measure to evaluate the reconstruction quality of the models.

The network is designed as part of a complete pipeline with preprocessing and a training strategy [5]. First, the signals are divided into spans corresponding to the written samples, where the parts before first stroke and after the last stroke are deleted, since they do not contain any useful information. Both the input signal and the ground truth signal must be of the same length to train the network. For this purpose, an alignment approach based on the Dynamic Time Warping algorithm (DTW) was used. The DTW algorithm respects the writing dynamics as much as possible, in contrast to linear alignment as in [4]. Finally, signals are split into strokes and the network is trained on touching strokes only as the variability of hovering strokes has an impact on the quality of training.

TABLE I: Studied NN

Model	Number of parameters	Kernel size	Dilations	Number of stacks
TCN-49	467,452	3	(1,2)	4
TCN-85	528,452	3	(1, 2, 4)	3
TCN-169	870,452	5	(1, 2, 4)	3
TCN-373	894,452	3	(1, 2, 4, 8, 16)	3

After preprocessing, these networks were trained on the full IRISA-KIHT dataset, which consists of 4,110 labeled handwriting samples, which were split into train dataset, containing 3,770 samples without hovering strokes, and the test dataset, containing 320 samples with hovering strokes. The training was performed on a batch size of 16 samples using Adam optimizer and a learning rate of 1e-4 on a MSE loss over 200 epochs. Trained models were evaluated using an average Fréchet distance, obtained on the test data set. Fréchet distances were normalized to the lengths of corresponding reference curves. In the following section we will consider the model optimization techniques used to enable the model deployment onto the embedded hardware.

C. Hardware-aware Quantization

First, we quantized the pretrained models with QKeras using fixed point quantization schemes from [17]. Since the studied NN uses only ReLU activation functions, which are quite easy to calculate, we have quantized weights only. The second step was to train and evaluate quantized networks. We implemented the quantized versions of the TCN-layers using Keras-TCN [18] and QKeras. Training quantized model is vulnerable to the loss oscillations due to the rounding errors especially during the back propagation. Therefore, QKeras leverages the straight-through estimator (STE), which performs the forward pass of the training applying the quantization functions and back propagation with non-quantized weights, assuming the quantization is the identity function in the backward pass, as the quantization function is not differentiable [19], [20].

Running QAT with randomly initialized weights usually is less efficient and stable than training the baseline model with full precision weights, especially taking into account, that the baseline model was trained on 200 epochs. To accelerate the QAT, we initialized the weights of models with values from their baseline counterparts, as proposed in [21]. Furthermore, we used the predictions of the corresponding baseline counterparts as labels for the QAT. With this approach, we can further reduce training time and define clean task separation between baseline model training and model deployment, which enables us to integrate the optimization pipliene into an MLOps platform, like MLFlow [22].

In order to further optimize networks for the hardware, we propose to apply a piecewise polynomial approximation of multiplication, which is achieved through the binary logarithm introduced by Mitchell [23]. As presented in a previous work

TABLE II: Average Fréchet distance between ground truth and predicted trajectories using different fixed-point precision for model weights with PTQ and QAT evaluated with QKeras and selected trajectory reconstruction TCN models.

Model	Quantization	Without Mitchell				With Mitchell			
		PTQ	QAT			PTQ	QAT		
			1 Epoch	5 Epochs	10 Epochs		1 Epoch	5 Epochs	10 Epochs
TCN-49	Baseline				**0.210**				
	(5,8)	0.256	0.267	0.244	**0.213**	2.008	0.317	0.275	0.274
	(8,16)	0.236	0.276	0.231	0.218	0.325	0.316	0.285	**0.267**
	(8,24)	**0.232**	0.272	0.234	0.213	0.443	0.320	0.302	0.274
TCN-85	Baseline				**0.247**				
	(5,8)	0.346	0.376	0.283	0.264	1.131	0.289	0.283	0.276
	(8,16)	0.347	0.328	0.271	0.254	0.509	0.300	0.293	**0.272**
	(8,24)	0.348	0.398	0.297	**0.245**	0.552	0.355	0.338	0.281
TCN-169	Baseline				**0.204**				
	(5,8)	0.252	0.347	0.247	**0.234**	4.372	0.324	0.340	0.340
	(8,16)	0.259	0.338	0.260	0.248	0.409	0.276	**0.259**	0.269
	(8,24)	0.255	0.362	0.244	0.256	0.453	0.368	0.281	0.267
TCN-373	Baseline				**0.170**				
	(5,8)	0.366	0.300	0.247	0.247	4.796	0.334	0.295	0.261
	(8,16)	0.282	0.302	0.235	0.247	0.417	0.415	**0.258**	0.274
	(8,24)	0.285	0.333	**0.230**	0.233	0.424	0.383	0.290	0.326

[17] and in [24], implementing a DNN accelerator that uses a hardware structure based on Mitchell's algorithm for multiplication provides lower hardware resource utilization on an FPGA or area consumption on an ASIC, decreased latency, and low energy consumption compared to full precision multipliers. In order to evaluate the influence of the Mitchell multiplier on the accuracy of the model, we implemented the custom Conv1D TensorFlow Kernel and integrated it into the training loop of the quantized TCN-models. Since we use the STE, we apply the original precision Conv1D operation to calculate the gradients for the backpropagation.

As it was shown in the [25], the accuracy of the AlexNet model tends to significantly degrade, when the Mitchell multiplication applied to all layers of the network. The most part of the impact on the accuracy comes from the last dense layers, since they accumulate the multiplication errors from all previous convolutional layers. For this reason we left out the Mitchell approximation for two last dense layers of studied networks.

The QAT for all models and evaluations were performed using the same data set as for baseline models. We used almost the same hyperparameter values, except the learning rate, which was tuned down to 1e-5 to improve the convergence of the model, and limited the number of the training epochs to 10, assuming that the weights of the pretrained models were already on plateau.

IV. Experimental results

In order to find out the best suitable optimization strategy for the deployment of the trajectory reconstruction algorithms onto the pen hardware, we quantized presented models and performed the QAT with and without Mitchells multipilcation approximation applied in the forward path of the training loop. For the model training we used a dedicated GPU server with NVIDIA RTX A6000 with 48 GB RAM and AMD EPYC 9554 128-Core CPU. The summary of the evaluations is presented in the Table II and accompanying reconstruction samples can be found in the Figure 2.

A. Effects of quantization schemes

As can be observed the naive PTQ of the models leads to the degradation of the reconstruction accuracy. Herewith the bigger models are more affected by the quantization errors, since the error is accumulated proportionally to amount of model layers. Only TCN-85 model does not follow this trend since the baseline model initially had lower accuracy than other models. The reduction of bit-width expectably leads to the worse reconstruction. Therefore the best reconstruction accuracy could be achieved with the (8,24) fixed-point quantization for TCN-49, which is supported by our results.

B. Effects of QAT

QAT was performed for 10 epochs with logged checkpoints at 1st and 5th epochs. Overall, we can see, that the application of QAT leads to the model improvement in all cases, since

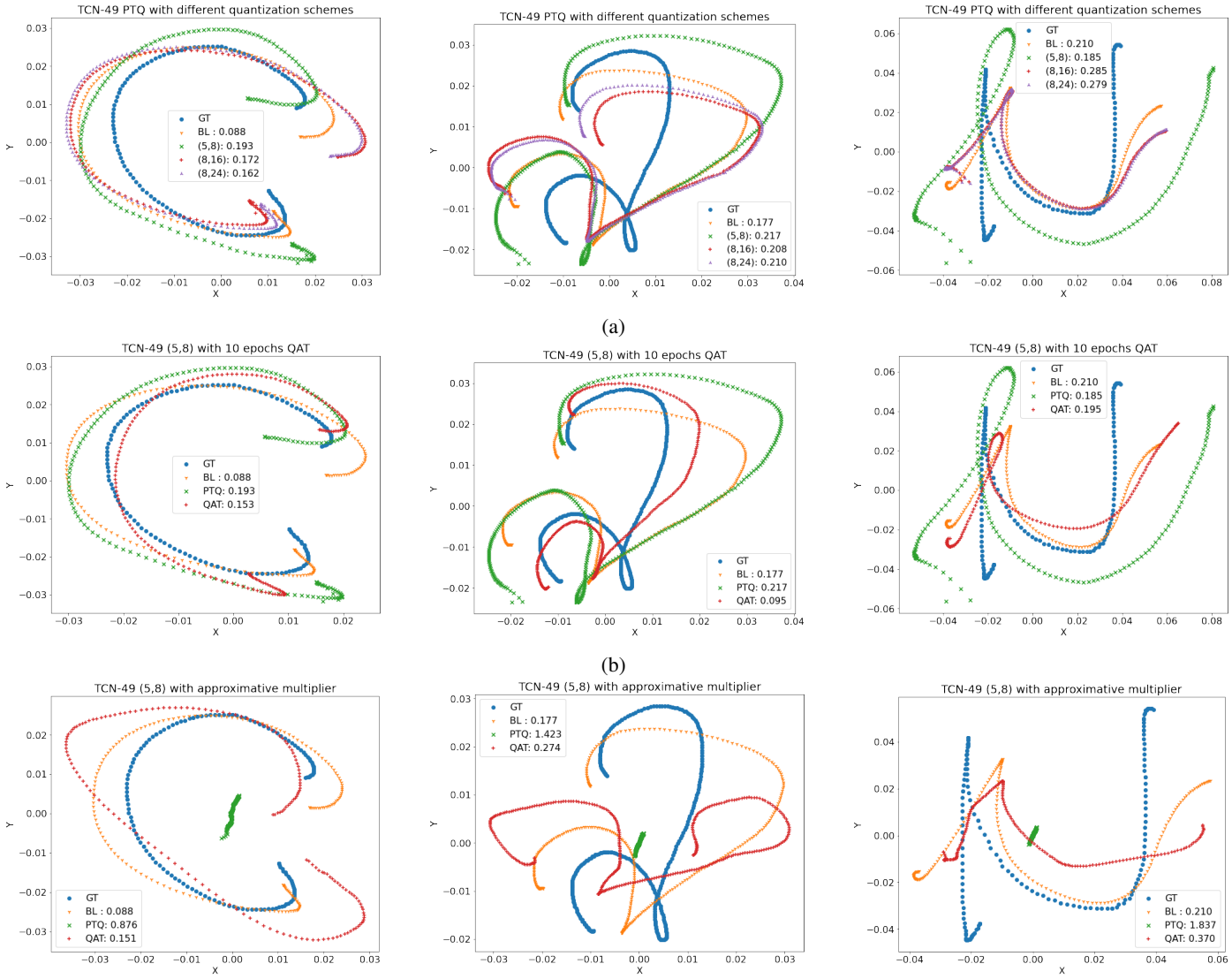

Fig. 2: Examples of reconstructions produced by Baseline (BL) and quantized models with corresponding Fréchet distance to the Ground Truth (GT) in normalized XY coordinates: (a) Comparison between different fixed-point quantization schemes. (b) Comparison between Post Training Quantization and Quantization-aware Training with 10 epochs. (c) Comparison between PTQ and QAT using Mitchell's approximate multiplication.

the quantized weights converged closer towards original ones, whereby, the model improvement is first observed after several epochs. The first epoch of QAT in generally leads to even less accuracy than of PTQ models.

After the 5th epoch, we observe different behaviour of studied models. By the 10th epoch, the TCN-49 achieves the accuracy close to the baseline counterpart accuracy for all quantization schemes. The TCN-85 steadily improves the accuracy over 10 epochs of QAT achieving even slightly better reconstruction accuracy, than its baseline counterpart. However, it is still worse, comparing to other studied models. The accuracy of TCN-169 and TCN-373 does not come that close to the baseline as the TCN-49. Accuracy of the TCN-373 (8,24) and TCN-169 (8,16) even start to oscillate after the 5th epoch.

For the QAT, we can observe as well that the TCN-49 is more stable after the quantization, achieving the best accuracy among studied models for the (5,8) quantization.

C. *Effects of Mitchell multiplication*

Combining the approximate multiplication applied to convolutional layers with PTQ drastically reduces the reconstruction quality for all studied models, making them completely unreadable (Figure 2c). However, the QAT allows us to recover part of the initial reconstruction quality.

The accuracy of TCN-49 and TCN-85 are more affected by the approximate multiplication than of bigger models. Each of these models converges to same value of the evaluation metric, showing slightly better accuracy for the (8,16) quantization. TCN-169 and TCN-373 converge faster, getting to their best results after 5 epochs. The consequent training epochs did not

show any improvement to these models. The best reconstruction accuracy can be achieved with TCN-169 (8,16) and TCN-373 (8,16) models.

We have observed the significant increase in training time for the models with Mitchell's approximate multiplication on our hardware. Running the Mitchell multiplication on non-specialized hardware like a GPU is less efficient than float32 multiplications, since it requires multiple bit operations.

D. Feasibility for hardware integration

Combining the different approaches for the model quantization we can significantly compress the models. The best quantized model with full precision multiplication would take only 460 KB of ROM, assuming that the memory is split into 4 KB chunks. In order to use of the approximate multiplication, the trade-off between accuracy and resource consumption should be met, since the best accuracy can be achieved with the TCN-169 (8,16) and TCN-373 (8,16) models, which in turn requires up to 1,752 KB ROM space for weights.

V. Conclusion and Outlook

In this work, we demonstrated, that it is possible to optimize the DNNs trained to solve regression task using QAT and approximate Mitchell multiplier. It allows to deploy these models onto the resource constrained devices and on the SoCs using accelerators for the approximate multiplications, minimizing the use of FPU and without significant model accuracy loss.

In this paper we limited our scope only to fixed quantization schemes. In the future work the application of the the hybrid quantization schemes, using tools such as AutoQKeras [19], should be explored, since it can help to achieve better trade-off between model accuracy and required resources. Another direction of study is the search for other training losses, which can better correlate with the model accuracy from one hand and requiring less computational resources than the Fréchet distance, in order to be able to integrate it into the training loop.

Acknowledgment

SPONSORED BY THE
Federal Ministry of Education and Research

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] M. Wehbi, D. Luge, T. Hamann *et al.*, "Surface-free multi-stroke trajectory reconstruction and word recognition using an imu-enhanced digital pen," *Sensors*, 2022.

[5] W. Swaileh, F. Imbert, Y. Soullard, R. Tavenard, and É. Anquetil, "Online handwriting trajectory reconstruction from kinematic sensors using temporal convolutional network," *International Journal on Document Analysis and Recognition (IJDAR)*, pp. 1–14, 2023.

[6] H. Yan, S. Herath, and Y. Furukawa, "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods," May 2019, arXiv:1905.12853 [cs]. [Online]. Available: http://arxiv.org/abs/1905.12853

[7] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "Tinyml: Current progress, research challenges, and future roadmap," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1303–1306.

[8] L. Wang, Z. Zhang, and P. Sun, "Quaternion-based kalman filter for ahrs using an adaptive-step gradient descent algorithm," *International Journal of Advanced Robotic Systems*, vol. 12, no. 9, p. 131, 2015. [Online]. Available: https://doi.org/10.5772/61313

[9] Y. Bu, L. Xie, Y. Yin, C. Wang, J. Ning, J. Cao, and S. Lu, "Handwriting-assistant: Reconstructing continuous strokes with millimeter-level accuracy via attachable inertial sensors," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 4, dec 2022. [Online]. Available: https://doi.org/10.1145/3494956

[10] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to Cure the Curse of Drift in Inertial Odometry," Jan. 2018, arXiv:1802.02209 [cs]. [Online]. Available: http://arxiv.org/abs/1802.02209

[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," Sep. 2016, arXiv:1609.03499 [cs]. [Online]. Available: http://arxiv.org/abs/1609.03499

[12] M. Nan, M. Trăscău, A. M. Florea *et al.*, "Comparison between recurrent networks and temporal convolutional networks approaches for skeleton-based action recognition," *Sensors*, vol. 21, no. 6, p. 2051, 2021.

[13] F. Kreß, A. Serdyuk, T. Hotfilter, J. Hoefer, T. Harbaum, J. Becker, and T. Hamann, "Hardware-aware workload distribution for ai-based online handwriting recognition in a sensor pen," in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, 2022, pp. 1–4.

[14] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 614–625, 2021.

[15] M. S. Kim, A. A. Del Barrio, R. Hermida, and N. Bagherzadeh, "Low-power implementation of Mitchell's approximate logarithmic multiplication for convolutional neural networks," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2018, pp. 617–622, iSSN: 2153-697X.

[16] H. Alt, C. Knauer, and C. Wenk, "Comparison of Distance Measures for Planar Curves," *Algorithmica*, vol. 38, no. 1, pp. 45–58, Jan. 2004. [Online]. Available: https://doi.org/10.1007/s00453-003-1042-5

[17] F. Kreß, A. Serdyuk, M. Hiegle, D. Waldmann, T. Hotfilter, J. Hoefer, T. Hamann, J. Barth, P. Kaempf, T. Harbaum, and J. Becker, "ATLAS: An Approximate Time-Series LSTM Accelerator for Low-Power IoT Applications," 2023, in press.

[18] P. Remy, "Temporal convolutional networks for keras," https://github.com/philipperemy/keras-tcn, 2020.

[19] C. N. Coelho Jr., A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, A. A. Pol, and S. Summers, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," 2021.

[20] B. Moons, K. Goetschalckx, N. V. Berckelaer, and M. Verhelst, "Minimum energy quantized neural networks," 2017.

[21] A. Fasoli, C.-Y. Chen, M. Serrano, X. Sun, N. Wang, S. Venkataramani, G. Saon, X. Cui, B. Kingsbury, W. Zhang, Z. Tüske, and K. Gopalakrishnan, "4-bit Quantization of LSTM-based Speech Recognition Models," Aug. 2021, arXiv:2108.12074 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2108.12074

[22] "Mlflow: A machine learning lifecycle platform," https://github.com/mlflow/mlflow, 2023.

[23] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.

[24] H. Kim, "A low-cost compensated approximate multiplier for bfloat16 data processing on convolutional neural network inference," *ETRI Journal*, vol. 43, no. 4, pp. 684–693, 2021.

[25] M. S. Kim, A. A. D. Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient mitchell's approximate log multipliers for convolutional neural networks," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 660–675, 2019.