3

Temporal Delta Layer: Exploiting Temporal Sparsity in Deep Neural Networks for Time-Series Data

Preetha Vijayan[1,2], Amirreza Yousefzadeh[2], Manolis Sifalakis[2], and Rene van Leuken[1]

[1]TU Delft, Netherlands
[2]imec, Netherlands

Abstract

Real-time video processing using state-of-the-art deep neural networks (DNN) has managed to achieve human-like accuracy in the recent past but at the cost of considerable energy consumption, rendering them infeasible for deployment on edge devices. The energy consumed by running DNNs on hardware accelerators is dominated by the number of memory read/writes and multiply-accumulate (MAC) operations required. This work explores the role of activation sparsity in efficient DNN inference as a potential solution. As matrix-vector multiplication of weights with activations is the most predominant operation in DNNs, skipping operations and memory fetches where (at least) one of them is a zero can make inference more energy efficient. Although spatial sparsification of activations is researched extensively, introducing and exploiting temporal sparsity has received far less attention in DNN literature. This work introduces a new DNN layer (called temporal delta layer) whose primary objective is to induce temporal activation sparsity during training. The temporal delta layer promotes activation sparsity by performing delta operation that is aided by activation quantization and l_1 norm based penalty to the cost function. As a result, the final model behaves like a conventional quantized DNN with high temporal activation sparsity during inference. The new layer was incorporated into the standard ResNet50 architecture to be trained and tested on the popular human action recognition

dataset, UCF101. The method resulted in a 2x improvement in activation sparsity, with a 5% reduction in accuracy.

3.1 Introduction

DNNs have lately managed to successfully analyze video data to perform action recognition [1], object tracking [2], object detection [3], etc., with human-like accuracy and robustness. Unfortunately, DNNs' high accuracy comes with considerable costs, in terms of computation and memory consumption, resulting in high energy consumption. This makes them unsuitable for always-on edge devices.

Techniques such as network pruning, quantization, regularization, and knowledge distillation [4] [5] have helped reduce model size over time, resulting in less compute and memory consumption overall. Sparsity is a prominent aspect in all of the aforementioned methods. This is significant because sparse tensors allow computations involving zero multiplication to be skipped. They are also easy to store and retrieve in memory. In the DNN literature, structural sparsity (of weights) and spatial sparsity (of activations) are well-studied topics [6]. However, while being a popular concept in neuromorphic computing, temporal activation sparsity has received less attention in the context of DNN.

This work applies the concept of change or delta based processing to the training and inference phases of deep neural networks, drawing inspiration from the human retina [7]. DNN inference, which processes each frame independently with no regard to the temporal correlation is dense and obscenely wasteful. Whereas, processing only the changes in the network can lead to zero-skipping in sparse tensor operations minimizing the redundant operations and memory accesses.

Therefore, the proposed methodology in this work induces temporal sparsity to theoretically any DNN by incorporating a new layer (named temporal delta layer), which can be introduced in a DNN at any phase (training, refinement, or inference only). This new layer can be integrated to an existing architecture by positioning it after all or some of the ReLU activation layers as deemed beneficial (see Figure 3.1). The inclusion of this layer does not necessitate any changes to the preceding or following layers. Furthermore, the new layer adds a novel sparsity penalty to the overall cost function of the DNN during the training phase. This l_1 norm based penalty minimizes the activation density of the delta maps (i.e., temporal difference between two consecutive feature maps). Apart from that, the new layer is compared

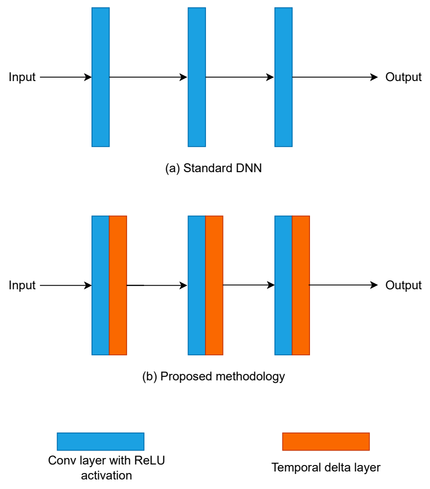


Figure 3.1 (a) Standard DNN, and (b) DNN with proposed temporal delta layer

in conjunction with two activation quantization methods, namely fixed-point quantization (FXP) and learned step-size quantization (LSQ).

3.2 Related Works

Although DNNs are in essence bio-inspired, they have not been able to find the balance between power consumption and accuracy yet, especially while dealing with computationally heavy streaming signals. On the other hand, the brain's neocortex handles complex tasks like sensory perception, planning, attention, and motor control while consuming less than 20 W [8]. Scalable architecture, in-memory computation, parallel processing, communication using spikes, low precision computation, sparse distributed representation, asynchronous execution, and fault tolerance are some of the characteristics of the biological neural networks that can be leveraged to bridge the energy consumption gap between the brain and DNNs [9]. Among these, the proposed methodology focuses on the viability of using sparsity within DNNs to achieve energy efficiency. During a matrix-vector multiplication between a weight matrix and an activation vector, zero elements in the tensor can be skipped leading to computational as well as memory access reduction (see Figure 1.2).

There are broadly two types of sparsity available in DNNs: weight sparsity (related to the interconnect between neurons) and activation sparsity

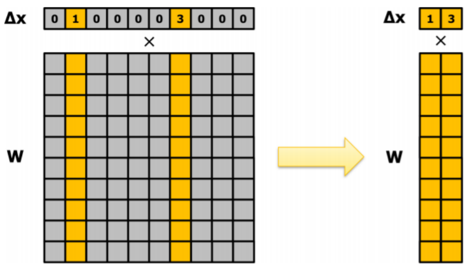


Figure 3.2 Sparsity in activation (Δx) drastically reduce the memory fetches and multiplications between Δx and columns of weight matrix, W, that correspond to zero [10].

(related to the number of neurons). Furthermore, activation sparsity can be categorised into spatial and temporal sparsity, which exploits the spatial and temporal correlation within the activations, respectively, [11]. Unlike weight and spatial sparsity [12, 13, 14, 15], exploiting the temporal redundancy of DNNs while processing streaming data as a means to reduce energy consumption is a relatively less explored idea. Exploiting temporal sparsity translates to skipping re-calculation of a function when its input remains unchanged since the last update.

One of the methods to exploit temporal sparsity is to use the compressed representation (like H.264, MPEG-4, etc.) of videos at the input stage itself. These compression techniques only retain a few key-frames completely and reconstruct others using motion vectors and residual error, thus using temporal redundancy [16] [17]. Another path includes finding a neuron model which is somewhere in between “frame-based DNN” and “event-based spiking neural networks”. This work is an attempt in the aforementioned direction. A similar work, CBInfer [18], proposes replacing all spatial convolution layers in a network with change-based temporal convolution layers (or CBconv layers). In this, a signal change is propagated forward only when a certain threshold is exceeded. Likewise, [19] tapped into temporal sparsity by introducing Sigma-Delta Networks, where neurons in one layer communicated with neurons in the next layer through discretized delta activations. An issue when it comes to CBInfer is the potential error accumulation over time as the method is threshold-based. If the neuron states are not reset periodically, this threshold can cause drift in the approximation of the activation signal and degrade the accuracy. Whereas, sigma-delta scheme experiments on smaller datasets like temporal MNIST, which might not be a reliable confirmation of the method's effectiveness.

3.3 Methodology

In video-based applications, traditional deep neural networks rely on frame-based processing. That is, each frame is processed entirely through all the layers of the model. However, there is very little change in going from one frame to the next through time, which is called temporal locality. Therefore, it is wasteful to perform computations to extract the features of the non-changing parts of the individual frame. Taking that concept deeper into the network, if feature maps of two consecutive frames are inspected after every activation layer throughout the model, this temporal overlap can be observed. Therefore, this work postulates that temporal sparsity can be significantly increased by focusing the inference of the model only on the changing pixels of the feature maps (or deltas).

3.3.1 Delta Inference

This work introduces a new layer that calculates the delta (or difference) between two temporally consecutive feature maps and quantifies the degree of these changes at only relevant locations in the frame. Since zero changes are not propagated through the layer, the role of this layer may be perceived as "analog event propagation". It is considered an "analog event" as it is not the presence of change, but the magnitude of change that is propagated through.

To better understand it mathematically, in a standard DNN layer, the output activation is related to its weights and input vector through Eq. 3.1 and 3.2.

$$Y_t = WX_t + B \tag{3.1}$$

$$Z_t = \sigma(Y_t) \tag{3.2}$$

where W and B represent the weights and bias parameters, Xt represents the input vector, and Yt represents the transitional state. Then, Zt is the output vector which is the result of $\sigma(.)$ - a non-linear activation function. t indicates that the tensor has a temporal dimension. However, in the temporal delta layer, weight-input multiplication transforms into,

$$\Delta Y_t = W\Delta X_t = W(X_t - X_{t-1}) \tag{3.3}$$

$$\begin{aligned} Y_t &= \Delta Y_t + Y_{t-1} \\ &= W(X_t - X_{t-1}) + W(X_{t-1} - X_{t-2}) + ... + Y_0, \;\; where\, Y_0 = B \\ &= WX_t + B, \end{aligned} \tag{3.4}$$

$$\Delta Z_t = Z_t - Z_{t-1} = \sigma(Y_t) - \sigma(Y_{t-1}), \; where\, \sigma(Y_0) = 0 \tag{3.5}$$

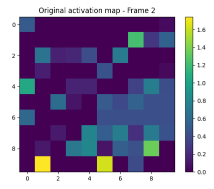
In Eq. 3.3, instead of using X_t directly, only changes or ΔX_t are multiplied with W. Using the resulting ΔY_t, the corresponding Y_t can be recursively calculated with Eq. 3.4, where Y_{t-1} is the transitional state obtained from the previous calculation. Eq. 3.5 is the final delta activation output that is passed onto the next layer.

Another notable difference between the standard DNN layer and the proposed layer is the role of bias. In delta based inference, bias is only used as an initialization for the transitional state, Y_0 in Eq. 3.4. However, since bias tensors do not change over time, their temporal difference is zero and is removed from Eq. 3.3.

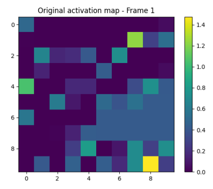
Now, as the input video is considered temporally correlated, the expectation is that ΔX_t and by association ΔZ_t are also temporally sparse. In essence, the temporal sparsity between consecutive feature maps is cast on the spatial sparsity of the delta map that is propagated. Additionally, Y_t in Eq. 3.1 and 3.4 are always equal. This indicates that as long as the input is the same, both standard DNN and temporal delta layer based DNN provide the same result at any time step.

3.3.2 Sparsity Induction Using Activation Quantization

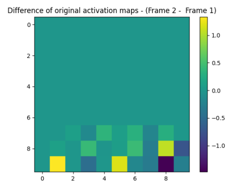
As shown in Figure 3.3, there is temporal redundancy evident in feature maps of two consecutive frames. However, if looked closely, it can be observed that these feature maps are similar but not identical as shown in Figure 3.3a and 3.3b. Therefore, if two such consecutive feature maps are subtracted, the resulting delta map has many near zero values, thus restricting the potential increase in temporal sparsity (Figure 3.3c). This is mainly due to the higher precision available in the floating point representation (FP32) of the activations. For example, in IEEE 754 representation, a single-precision 32-bit floating point number has 1 bit for sign, 8 bits for the exponent and 23 bits for the significant. It, not only, leads to a very high dynamic range, but also, increases the resolution or precision for numbers close to 0. The number nearest to 0 is about ± 1.4 x 10^{-45}. Therefore, due to high resolution, two similar floating point values have difficulty going to absolute zero when



(a)



(b)



(c)

Figure 3.3 Demonstration of two temporally consecutive activation maps leading to near zero values (rather than absolute zeroes) after delta operation.

subtracted. A plausible solution to decrease the precision of the activations is to use quantization.

In this work, a post-training quantization method (fixed point quantization [20]) and a quantization aware training method (learnable step size quantization [21]) are considered for comparison as a temporal sparsity facilitator for the new layer.

3.3.2.1 Fixed Point Quantization

In this method, the floating point numbers are quantized to integer or fixed point representation [20]. Unlike floating point, in fixed point representation, the integer and the fractional part have fixed length. This limits both range and precision. That is, if more bits are used to represent the integer part, it subsequently decreases the precision and vice versa.

Method:

Firstly, a bitwidth is defined to which the 32-bit floating parameter is to be quantized, BW. Then, the number of bits required to represent the unsigned integer part of the parameter (x) is calculated as shown in Eq. 3.6.

$$I = 1 + \lfloor log_2 \left(\max_{1<i<N} |x| \right) \rfloor \tag{3.6}$$

A positive value of I means that I bits are required to represent the absolute value of the integer part, while a negative value of I means that the fractional part has I leading unused bits. Now, it is known that 1 bit is for sign, so the number of fractional bits, F, is given by Eq. 3.7.

$$F = BW - I - 1 \tag{3.7}$$

Considering the parameters, BW - bitwidth, F - fractional bits, I - integer bits, and S - sign bit, Eq. 3.8 maps the floating point parameter x to the fixed point by,

$$Q(x) = \frac{C(R(x.2^F), -t, t)}{2^F} \tag{3.8}$$

where $R(.)$ is the round function, $C(x, a, b)$ is the clipping function, and t is defined as,

$$t = \begin{cases} 2^{BW-S}, & BW > 1 \\ 0 & BW \leq 1 \end{cases}$$

Possible Drawback of Fixed Point Quantization:
Fixed point quantization, as shown above, is a fairly straightforward mapping scheme and is easy to be included in the model training process during the forward pass before the actual delta calculation. However, it poses a limitation to the extent of quantization possible without sacrificing accuracy. Typically, an 8-bit quantization can sustain floating point accuracy with this method, but if the bitwidth goes below 8 bits, the accuracy starts to deteriorate significantly. This is because, unlike weights, activations are dynamic and activation patterns change from input to input making them more sensitive to harsh quantization [22]. Also, quantizing the layers of a network to the same bitwidth can mean that the inter-channel behaviour of the feature maps are not captured properly. Since the number of fractional bits is usually selected depending on the maximum activation value in a layer, this type of quantization tends to cause excessive information loss in channels with a smaller range.

3.3.2.2 Learned Step-Size Quantization

Quantization aware training is the most logical solution to the aforementioned drawback as it can potentially recover the accuracy in low bit tasks given enough time to train. Therefore, a symmetric uniform quantization scheme is considered called Learned Step size Quantization (LSQ). This method considers the quantizer itself as a trainable parameter which is trying to minimize the task loss using backpropagation and stochastic gradient descent. This serves two purposes: (a) step size, which is the width of quantization bins, gets to be adaptive through the training according to the activation distribution. It is vital to find an optimum step size because, as shown in Figure 3.4, if the step size is too small or too large, it can lead to the quantized

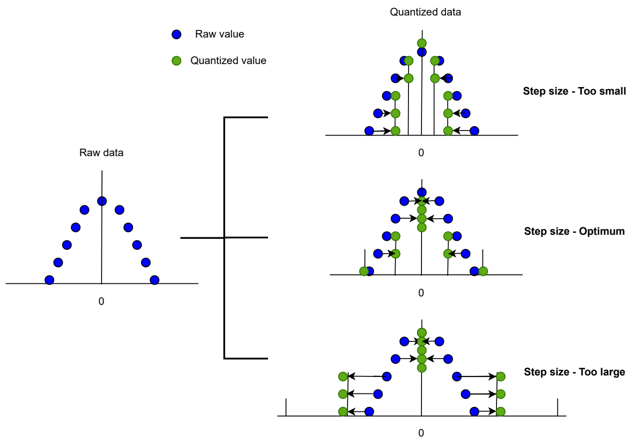


Figure 3.4 Importance of step size in quantization: on the right side, in all three cases, the data is quantized to five bins with different uniform step sizes. However, without optimum step size value, the quantization can detrimentally alter the range and resolution of the original data.

data being a poor representation of the raw data. (b) as the step size is a model parameter, it is also directly seeking to improve the metric of interest, i.e. accuracy.

Method:

Given: x - the parameter to be quantized, s - step size, Q_N and Q_P - number of negative and positive quantization levels respectively, and q(x;s) is the quantized representation with the same scale as x,

$$q(x;s) = \begin{cases} \lfloor \frac{x}{s} \rceil .s, & \text{if } -Q_N \leq \frac{x}{s} \leq Q_P \\ -Q_N.s, & \text{if } \frac{x}{s} \leq -Q_N \\ Q_P.s, & \text{if } \frac{x}{s} \geq Q_P \end{cases} \tag{3.9}$$

where $\lfloor a \rceil$ rounds the value to the nearest integer. Considering the number of bits, b, to which the data is to be quantized, Q_N = 0 for unsigned and Q_N = 2^{b-1} for signed data. Similarly, Q_P = 2^{b-1} for unsigned and $2^{b-1} - 1$ for signed data.

Modified LSQ:

In this work, the original LSQ method is slightly modified to remove the clipping function from the equations as (a) the bitwidth, b, required to calculate

Q_N and Q_P is not known. This is because the bitwidth is not pre-defined and is determined using the activation statistics of each layer while training which leads to a mixed precision model, which is more advantageous, and (b) clipping leads to accuracy drop as it alters the range of the activation. That is, if activations are clipped during training, there could be a significant difference between the real-valued activation value and the quantized activation value, which in turn affects the gradient calculations and, therefore the SGD optimization.

Thus, in temporal delta layer, the forward pass of the quantization includes only scaling, rounding and de-scaling and can be mathematically expressed as,

$$q(x; s) = \lfloor \frac{x}{s} \rceil . s \tag{3.10}$$

The gradient of the Eq. 3.10 for backpropagation is given by Eq. 3.11.

$$\nabla_s q(x; s) = \lfloor \frac{x}{s} \rceil - \frac{x}{s} \tag{3.11}$$

3.3.3 Sparsity Penalty

Quantized delta map, created using the above-mentioned methods, in itself has a fair number of absolute zeroes (or sparsity) available. However, like the biological brain, learning can help in increasing this sparsity further. The inspiration for this came from an elegant set of experiments performed by Y. Yu et al. [23]. The experiment showed a particular 30 seconds video to rodent specimens and tracked their activation density during each presentation. It was found that activation density decreased as the number of trials increased, i.e as the learning increased, the active neurons required for inference decreases.

Adapting the said concept to this work, a l_1 norm based constraint is introduced to the loss function. This is termed as the *sparsity penalty*. Therefore, the new cost function can be mathematically expressed as *cost function = task loss + sparsity penalty*, i.e,

$$Cost\ function = Task\ loss + \lambda\ (\frac{l_1\ norm\ of\ active\ neurons\ in\ delta\ map}{total\ number\ of\ neurons\ in\ delta\ map}) \tag{3.12}$$

where task loss minimizes the error between the true value and the predicted value and, sparsity penalty minimizes the overall temporal activation density. The λ mentioned in Eq. 3.12 refers to the penalty co-efficient of the cost function. If λ is too small, the sparsity penalty takes little effect and model accuracy is given more priority and if λ is too large, sparsity becomes the priority leading to very sparse models but with unacceptable accuracy. The key is to find the balance between task loss and sparsity penalty.

3.4 Experiments and Results

In this section, the proposed methodology explained in section 3.4 is analyzed to study how it helps achieve the desired temporal sparsity and accuracy.

3.4.1 Baseline

For baseline, the two-stream architecture [24] was used, with ResNet50 as the feature extractor on both spatial and temporal streams. The dataset used was UCF101, which is a widely used human action recognition dataset of 'in-the-wild' action videos, obtained from YouTube, having 101 action categories [25]. The spatial stream used single-frame RGB images of size (224, 224, 3) as the input, while the temporal stream used stacks of 10 RGB difference frames of size (224, 224, 10×3) as the input. Also, both these inputs were time distributed to apply the same layer to multiple frames simultaneously and produce output that has time as the fourth dimension. Both the streams were initialized with pre-trained ImageNet weights and fine-tuned with an SGD optimizer.

Under the above-mentioned setup, spatial and temporal streams achieved an accuracy of 75% and 70%, respectively. Then, both streams were *average fused* to achieve a final classification accuracy of 82%. Also, in this scenario, both streams were found to have an activation sparsity of $\sim$ 47%.

3.4.2 Experiments

Scenario 1: The setup consecutively places the fixed point based quantization layer and temporal delta layer after every activation layer in the network. The temporal delta layer here also includes a l_1 norm based penalty. The baseline weights were used as a starting point, and all the layers including the temporal delta layer is fine-tuned until acceptable convergence. The hyper-parameters specifically required for this setup were bitwidth (to which the activations were to be quantized) and penalty co-efficient to balance the tussle between task loss and sparsity penalty.

Scenario 2: The setup is similar to the previous scenario except for the activation quantization method used. The previous experiment used fixed precision quantization where all the activation layers in the network were quantized to the same bitwidth. However, this experiment uses learnable step-size quantization (LSQ), which performs channel-wise quantization depending on the activation distribution resulting in mixed-precision quantization of the activation maps.

The layer also introduces a hyperparameter during training (apart from the penalty coefficient mentioned earlier) for the step size initialization. Then, during training, the step size increases or decreases depending on the activation distribution in each channel.

3.4.3 Result Analysis

Table 3.1 and 3.1 show the baseline accuracy and activation sparsity compared against the two scenarios mentioned.

Firstly, when the temporal delta layers with fixed point quantized activations are included in the baseline model, it can be observed that the activation sparsity increases considerably with a slight loss in accuracy in both streams.

Table 3.1 Spatial stream - comparison of accuracy and activation sparsity obtained through the proposed scenarios against the baseline. In the case of fixed point quantization, the reported results are for a bitwidth of 6 bits.

Model setup (Spatial stream)	**Accuracy**	**Activation sparsity**
Baseline	75%	48%
Temporal delta layer with fixed point quantization	73%	74%
Temporal delta layer with learned step-size quantization	**69%**	**86%**

Table 3.2 Temporal stream - comparison of accuracy and activation sparsity obtained through the proposed scenarios against the benchmark. In the case of fixed point quantization, the reported results are for a bitwidth of 7 bits.

Model setup (Temporal stream)	**Accuracy**	**Activation sparsity**
Baseline	70%	47%
Temporal delta layer with fixed point quantization	68%	67%
Temporal delta layer with learned step-size quantization	**65%**	**89%**

This is because lowering the precision from 32 bits to 8 bits (or less) leads to temporal differences of activations going to absolute zero.

Additionally, the reason for close-to baseline accuracy in the method involving fixed point quantization can be attributed to fractional bit allocation flexibility. That is, as the bitwidth is fixed, the number of integer bits required is decided depending on the activation distribution within the layer, and the rest of the bits are assigned as fractional bits. This makes sure that the precision of the activation is compromised for range. Also, another contributing factor for accuracy sustenance is that the first and the last layers of the model are not quantized, similar to works like [26][27]. This is because the first and last layer has a lot of information density. Those are the layers where input pixels turn into features and features turn into output probabilities, respectively, which makes them more sensitive to quantization.

Although the activation sparsity gain in the case of the temporal delta layer with fixed point quantization is better than the baseline, it is still not sufficiently high as required. In this effort, the bitwidth of the activations are decreased in the expectation of increasing sparsity. However, as the bitwidth goes below a certain value (6 bits for spatial and 7 bits for temporal stream), sparsity increases, but accuracy starts to deteriorate beyond recovery, as shown in Table 3.3. This is because quantizing all layers of a network to the same bitwidth can mean that the inter-channel variations of the feature maps are not fully accounted for. Since the number of fractional bits is usually selected to cover the maximum activation value in a layer, the fixed bitwidth quantization tends to cause excessive information loss in channels with a smaller dynamic range. Therefore, it can be inferred that mixed-precision quantization of activations is a better approach to obtain good sparsity without compromising accuracy.

Table 3.3 Result of decreasing activation bitwidth in fixed point quantization method. For spatial stream, decreasing below 6 bits caused the accuracy to drop considerably. For temporal stream, the same happened below 7 bits.

	Spatial stream		**Temporal stream**	
Activation bitwidth	**Accuracy (%)**	**Activation sparsity (%)**	**Accuracy (%)**	**Activation sparsity (%)**
32	75	50	70	47
8	75	68	70	65
7	75	71	**68**	**70**
6	**73**	**75**	61	73
5	65	80	-	-

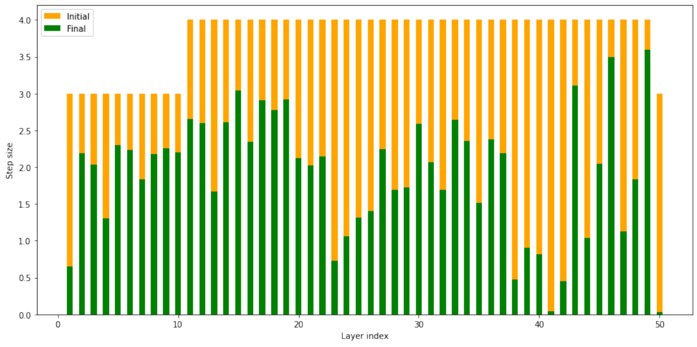


Figure 3.5 Evolution of quantization step size from initialization to convergence in LSQ. As step-size is a learnable parameter, it gets re-adjusted during training to cause minimum information loss in each layer.

Finally, using the temporal delta layer where incoming activations are quantized using learnable step-size quantization (LSQ) gives the best results for both spatial and temporal streams. As the step size is a learnable parameter, it gives the model enough flexibility to result in a mixed precision model, where each channel in a layer has a bitwidth that suits its activation distribution. This kind of channel-wise quantization minimizes the impact of low-precision rounding. It is also evident in Figure 3.5 that as the training nears convergence, the values of the step size differ according to the activation distribution and bitwidth required to represent each layer. Moreover, consistent with the expectation, the first and last layers during training opts for smaller step sizes implying they need more bitwidth for their representation.

Table 3.4 Final results from two-stream network after average fusing the spatial and temporal stream weights. With 5% accuracy loss, the proposed method almost doubles the activation sparsity available in comparison to the baseline.

	Baseline		**Proposed method**	
Model type	**Accuracy (%)**	**Activation sparsity (%)**	**Accuracy (%)**	**Activation sparsity (%)**
Spatial stream	75	50	69	86
Temporal stream	70	46	65	89
Two-stream (Average fused)	82	47	**77**	**88**

The weights generated using this method was then average fused to find the final two-stream network accuracy and activation sparsity (Table 3.4). Finally, the proposed method can achieve an overall 88% activation sparsity with 5% accuracy loss.

3.5 Conclusion

Intuitively, the proposed new temporal delta layer projects the temporal activation sparsity between two consecutive feature maps onto the spatial activation sparsity of their delta map. When executing sparse tensor multiplications in hardware, this spatial sparsity can be used to decrease the computations and memory accesses. As shown in Table 3.4, the proposed method resulted in 88% overall activation sparsity with a trade-off of 5% accuracy drop on UCF-101 dataset.

The collateral benefit of the obtained temporal sparsity is that the computations does not increase linearly with the increase in frame rate. In typical DNNs, doubling the frame rate would automatically necessitate doubling the computations. However, in the case of temporal delta layer based model, increasing the frame rate will not only improve the temporal precision of the network but also increase its temporal sparsity limiting the computations required [28].

The downside of using the temporal delta layer is that it requires keeping track of previous activations in order to perform delta operations. As a result, the overall memory footprint grows, putting more reliance on off-chip memory. However, the rising popularity of novel memory technologies (like resistive RAM [29], embedded Flash memory [30], etc.) may improve the cost calculations in the near future.

Disclaimer: This paper is a distillation of the research done by one of the authors as a part of her master thesis and is partially published in chapter 3 of [32]. The complete thesis, along with the results and analysis, is available online [31].

Acknowledgment

This work is partially funded by research and innovation projects TEMPO (ECSEL JU under grant agreement No 826655), ANDANTE (ECSEL JU under grant agreement No 876925) and DAIS (KDT JU under grant agreement No 101007273), SunRISE (EUREKA cluster PENTA2018e-17004-SunRISE) and Comp4Drones (ECSEL JU grant agreement No. 826610).

The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Spain, Portugal, Belgium, Germany, Slovenia, Czech Republic, Netherlands, Denmark, Norway and Turkey.

References

[1] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[2] K. Chen and W. Tao, "Once for all: a two-flow convolutional neural network for visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 12, pp. 3377–3386, 2017.

[3] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, *et al.*, "T-cnn: Tubelets with convolutional neural networks for object detection from videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, 2017.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[6] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *arXiv preprint arXiv:1608.03665*, 2016.

[7] M. Mahowald, "The silicon retina," in *An Analog VLSI System for Stereoscopic Vision*, pp. 4–65, Springer, 1994.

[8] J. W. Mink, R. J. Blumenschine, and D. B. Adams, "Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis," *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, vol. 241, no. 3, pp. R203–R212, 1981.

[9] A. Yousefzadeh, M. A. Khoei, S. Hosseini, P. Holanda, S. Leroux, O. Moreira, J. Tapson, B. Dhoedt, P. Simoens, T. Serrano-Gotarredona, *et al.*, "Asynchronous spiking neurons, the natural key to exploit temporal sparsity," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 668–678, 2019.

[10] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, “Deltarnn: A power-efficient recurrent neural network accelerator,” in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 21–30, 2018.

[11] O. Moreira, A. Yousefzadeh, F. Chersi, G. Cinserin, R.-J. Zwartenkot, A. Kapoor, P. Qiao, P. Kievits, M. Khoei, L. Rouillard, *et al.*, “Neuronflow: a neuromorphic processor architecture for live ai applications,” in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 840–845, IEEE, 2020.

[12] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.

[13] H. Yang, W. Wen, and H. Li, “Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures,” *arXiv preprint arXiv:1908.09979*, 2019.

[14] S. Seto, M. T. Wells, and W. Zhang, “Halo: Learning to prune neural networks with shrinkage,” in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 558–566, SIAM, 2021.

[15] M. Mahmoud, K. Siu, and A. Moshovos, “Diffy: A déjà vu-free differential deep neural network accelerator,” in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 134–147, IEEE, 2018.

[16] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, “Compressed video action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6026–6035, 2018.

[17] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, “Eva^2: Exploiting temporal redundancy in live computer vision,” in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 533–546, IEEE, 2018.

[18] L. Cavigelli, P. Degen, and L. Benini, “Cbinfer: Change-based inference for convolutional neural networks on video data,” in *Proceedings of the 11th International Conference on Distributed Smart Cameras*, pp. 1–8, 2017.

[19] P. O’Connor and M. Welling, “Sigma delta quantized networks,” *arXiv preprint arXiv:1611.02024*, 2016.

[20] P.-E. Novac, G. B. Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, “Quantization and deployment of deep neural networks on microcontrollers,” *Sensors*, vol. 21, no. 9, p. 2984, 2021.

[21] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[22] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[23] Y. Yu, R. Hira, J. N. Stirman, W. Yu, I. T. Smith, and S. L. Smith, "Mice use robust and common strategies to discriminate natural scenes," *Scientific reports*, vol. 8, no. 1, pp. 1–13, 2018.

[24] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *arXiv preprint arXiv:1406.2199*, 2014.

[25] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[26] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[27] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[28] M. A. Khoei, A. Yousefzadeh, A. Pourtaherian, O. Moreira, and J. Tapson, "Sparnet: Sparse asynchronous neural network execution for energy efficient inference," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 256–260, IEEE, 2020.

[29] S. Huang, A. Ankit, P. Silveira, R. Antunes, S. R. Chalamalasetti, I. El Hajj, D. E. Kim, G. Aguiar, P. Bruel, S. Serebryakov, *et al.*, "Mixed precision quantization for reram-based dnn inference accelerators," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 372–377, IEEE, 2021.

[30] M. Kang, H. Kim, H. Shin, J. Sim, K. Kim, and L.-S. Kim, "S-flash: A nand flash-based deep neural network accelerator exploiting bit-level sparsity," *IEEE Transactions on Computers*, 2021.

[31] P. Vijayan, "Temporal Delta Layer." http://resolver.tudelft.nl/uuid:0806241d-9037-4094-a197-6e65d6482f2b.

[32] O. Vermesan and M. Diaz Nava (Eds), Intelligent Edge-Embedded Technologies for Digitising Industry ISBN: 9788770226103, River Publishers, Gistrup, Denmark, 2022.