Appendix: Tiled Squeeze-and-Excite

A. TSE Code

An implementation in PyTorch of the TSE block is given in Figure A1.

```
def TSE(x, kernel, se_ratio):
    # x: input feature map [N, C, H, W]
    # kernel: tile size (Kh, Kw)
    # se_ratio: SE channel reduction ratio

    N, C, H, W = x.size()

    # tiled squeeze
    sq = nn.AvgPool2d(kernel, stride=kernel, ceil_mode=True)
    # original se excitation
    ex = nn.Sequential(
        nn.Conv2d(C, C // se_ratio, 1),
        nn.ReLU(inplace=True),
        nn.Conv2d(C // se_ratio, C, 1),
        nn.Sigmoid()
    )
    y = ex(sq(x))
    # nearest neighbor interpolation
    y = torch.repeat_interleave(y, kernel[0], dim=-2)[:,:,:H,:]
    y = torch.repeat_interleave(y, kernel[1], dim=-1)[:,:,:,:W]
    return x * y
```

Figure A1: PyTorch code of our TSE block

B. Buffering and Latency in Dataflow Architectures

Dataflow-defined accelerators are highly optimized for power efficiency and throughput by leveraging the fixed flow of data in a DNN. This is usually achieved by having a network of processing elements (PE) passing data one to another using a fixed dataflow. An optimized dataflow minimizes access from energy consuming levels of the memory hierarchy (e.g. off-chip DRAM) in favour of locally-cached memory (see [29] for comprehensive review). Inter-layer pipelining [28] is a very effective method to reduce DRAM access and is commonly used by both industrial and academic solutions (see [5] for an overview). Inter-layer pipeling is important for our purposes so we dwell on this point. Let's assume that we have two consecutive 3×3 convolutional layers L_1 and L_2 (padding=1 and stride=1). For simplicity, assume each layer processes it's input row-wise (same setting as in most of our experiments) as is assigned to it's own PE. Once L_1 outputs it's first two output rows, L_2 can consume them and begin computing it's output. In this setting, the only required access to off-chip DRAM is to read/write the input/output to the pipeline while intermediate activations are stored locally (see [28] for a general derivation). Let us now compare the SE and TSE ops: Assume an input tensor of dimensions $H \times W \times C$. The GAP op in SE stops the pipeline (as all elements are required for a single output) which means we need to allocate $H \times W \times C$ of on-chip memory. In contrast, the TSE op operates on a tile of size $h \times w$ and once it is processed, the next element in the pipeline can start it's operation and the tile can be discarded. So for TSE we only need to allocate $h \times w \times C$ on-chip memory. Thus TSE requires only the fraction $(h \times w)/(H \times W)$ of the memory required by SE. The above illustrates the principles of why TSE is beneficial. Actual gains will depend on implementation details of the hardware and software. To that end, we measure latency improvement on the Hailo-8 [1], an AI-Accelerator which uses inter-layer pipelining. Results are given in Table 1. To get a clean measurement that is not dependent on the overall system configuration (e.g DRAM bandwidth) we only measure the latency of pipeline itself (i.e we assume memory movements is instantaneous). Since decreasing buffering means less movement between local and non-local memory, the numbers presented below are actually an underestimate of the overall

Network	Buffers		Latency Improvement
	SE	TSE	
RegNetY-200MF	0.38M	0.20M	$\times$1.25
RegNetY-400MF	0.76M	0.33M	$\times$1.53
RegNetY-600MF	0.88M	0.37M	$\times$1.68
RegNetY-800MF	1.07M	0.42M	$\times$1.79
RegNetY-1.6GF	2.07M	0.82M	$\times$1.87

Table B1: Measured latency on the Hailo-8. We see that latency is considerably improved when using TSE$_{7,W}$ instead of SE.

latency improvement. We see that even though the SE op adds only a small amount of compute, it has a big effect on the overall latency in pipelined architectures. Note that the results give relative improvement in latency as we can't disclose absolute latency numbers.

C. Different Design Selections

Here, we introduce different design selections for TSE that minimize the spatial context of the operation without losing accuracy. We will focus on two different modifications: (a) change the channel reduction ratio, and (b) change the 1x1 convolutions of the excitation step to different operations, *e.g.*, 3x3 convolution. We note that those alteration change the number of parameters in the model and thus make it incompatible with standard SE block. The target of this experiment is to examine whether a model with single row strip pooling or without any pooling can achieve the same accuracy as SE network with GAP.

We consider two operations to swap the 1x1 convolution with: $Conv2D_{3\times3}$ and $Conv2D_{3\times1}$ and each model is named according to the following scheme: TSE$_{h\times w}C_{k_x\times k_y}R_c$ where $C_{k_x\times k_y}$ and R_c are the dimensions of the convolution kernel and the channel reduction ratio, respectively. For instance, TSE$_{1\times W}C_{3\times1}R_2$ is a model with strip pooling of a single row, channel reduction ratio of 2 and two $Conv2D_{3\times1}$ operations. The baseline in our experiment is RegNetY-800MF model [25] which has a native channel reduction ratio of 4. The results are shown in Table C2.

TSE Without Pooling. In the first experiment we use TSE without pooling, e.g., TSE$_{1\times1}$ (bottom part of Table C2). The baseline top-1 accuracy of 75.54% obtained with the original excitation step of SENet and without any pooling. First, we

Method	Params	GFLOPs	Buffer	Top-1
SE	6.2M	0.79	1.07M	76.30
TSE$_{1\times W}C_{1\times1}R_4$	6.2M	0.80	0.06M	75.79
TSE$_{1\times W}C_{1\times1}R_2$	7.08M	0.81	0.06M	75.90
TSE$_{1\times W}C_{3\times1}R_4$	7.8M	0.82	0.06M	76.40
TSE$_{1\times W}C_{3\times1}R_2$	10.3M	0.84	0.06M	76.43
TSE$_{1\times1}C_{1\times1}R_4$	6.2M	0.93	-	75.54
TSE$_{1\times1}C_{1\times1}R_2$	7.08M	2.84	-	75.90
TSE$_{1\times1}C_{3\times3}R_2$	20.17M	2.84	-	77.14

Table C2: Different design selections for TSE. Top-1 accuracy comparison of different RegNetY-800MF models on ImageNet-1K. The naming convention is: TSE$_{h\times w}C_{k_x\times k_y}R_c$ where $C_{k_x\times k_y}$ and R_c are the dimensions of the convolution kernel and the channel reduction ratio, respectively.

Model	Input	SE			TSE$_{1\times W}$		
		mAP	Buffer	GFLOPs	mAP	Buffer	GFLOPs
EfficientDet-D0	512$\times$512	33.8	13.8M	2.5	34.4	0.28M	2.7
EfficientDet-D1	640$\times$640	39.0	33.6M	6.1	39.5	0.52M	6.7
EfficientDet-D2	768$\times$768	42.3	50.8M	11	42.4	0.68M	11.8

Table C3: Comparison of mAP accuracy results on MS COCO-2017 validation set for different EfficientDet models [32] with TSE$_{1\times w}C_{3\times1}R_4$.

show that decreasing the channel reduction ratio can increase the accuracy of the network by 0.36%. However, even with reduction ratio of 2 the network has a large accuracy margin compare to the SE model. Second, we replace the $Conv2D_{1\times1}$ operation with $Conv2D_{3\times3}$. This replacement increases the number of parameters and compute but also improves network accuracy above the SE baseline. The purpose of this experiment is to show that GAP (or any global spatial context) is not mandatory for channel attention and variations in the excitation step can 'overcome' the lack of spatial context and even give improvement over more basic excitation schemes with global pooling. It also shows the trade-off between the number of parameters and compute to pipeline buffering which optimally can be optimized to a specific hardware.

Single Row Strip-tiling. Here, we use TSE with a single row strip-tiling, *e.g.*, TSE$_{1\times W}$. Unlike the first experiment, here, we squeeze the rows which induces less computation. We also note that changing the channel reduction ratio has limited accuracy gains. To match SE accuracy in this case, we swap the $Conv2D_{1\times1}$ convolution to $Conv2D_{3\times1}$ which increases the number of parameters (even with the same channel reduction ratio). This variant of TSE shows how minimum amount of spatial context can be enough to generate meaningful attention factors. We further verified that TSE$_{1\times W}C_{3\times1}R_4$ can be used with high resolution networks as well. For this experiment, we employ the EfficientDet [32] models. The results are shown in Table C3 and shows that the same accuracy can be achieved with less pipeline buffering with the cost of adding a small amount of additional computations.

In summary, both experiments show that there is a rather large design space for SE-like operations. Spatial squeezing has a dual role of decreasing computation and aggregating spatial context, however, the more spatial information is squeezed, the greater the buffering required. At the extreme, spatial squeezing can be avoided altogether with the cost of increased computations and reduced accuracy. Adding a spatial component to the excite operation improves the performance of channel attention while significantly increasing the number of parameters. In fact, using $Conv2D_{3\times3}$ has superior performance than the original SE block. The original SE block uses GAP to reduce the compute to minimum and a simple excitation to reduce the parameter count to a minimum, however, is also maximizes the required pipeline buffering. Based on the above observations, TSE is designed to bring all three of compute, parameters and buffering to a minimum while maintaining accuracy.