



Exponent Sharing for Model Compression

Prachi Kashikar and Sharad Sinha

Email: prachi.183311004@iitgoa.ac.in and sharad@iitgoa.ac.in

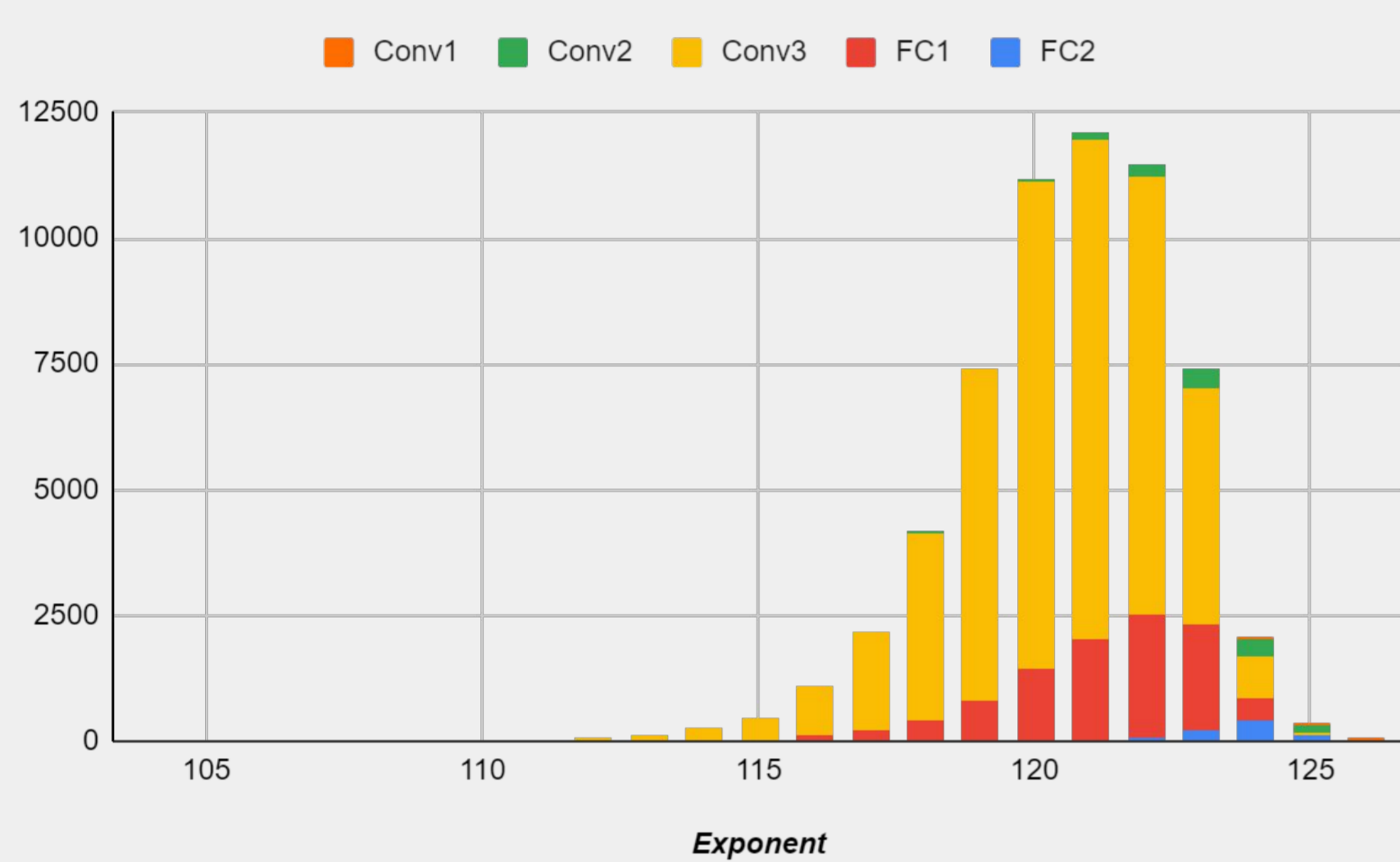
School of Mathematics and Computer Science, Indian Institute of Technology Goa, India

Abstract

AI on the edge has become a significant focus of research in the past decade. This area involves deploying various applications in computer vision and natural language processing on small devices with limited on-chip memory and battery power. However, neural networks require large storage space. It is crucial to compress the models to make them fit into these small devices.

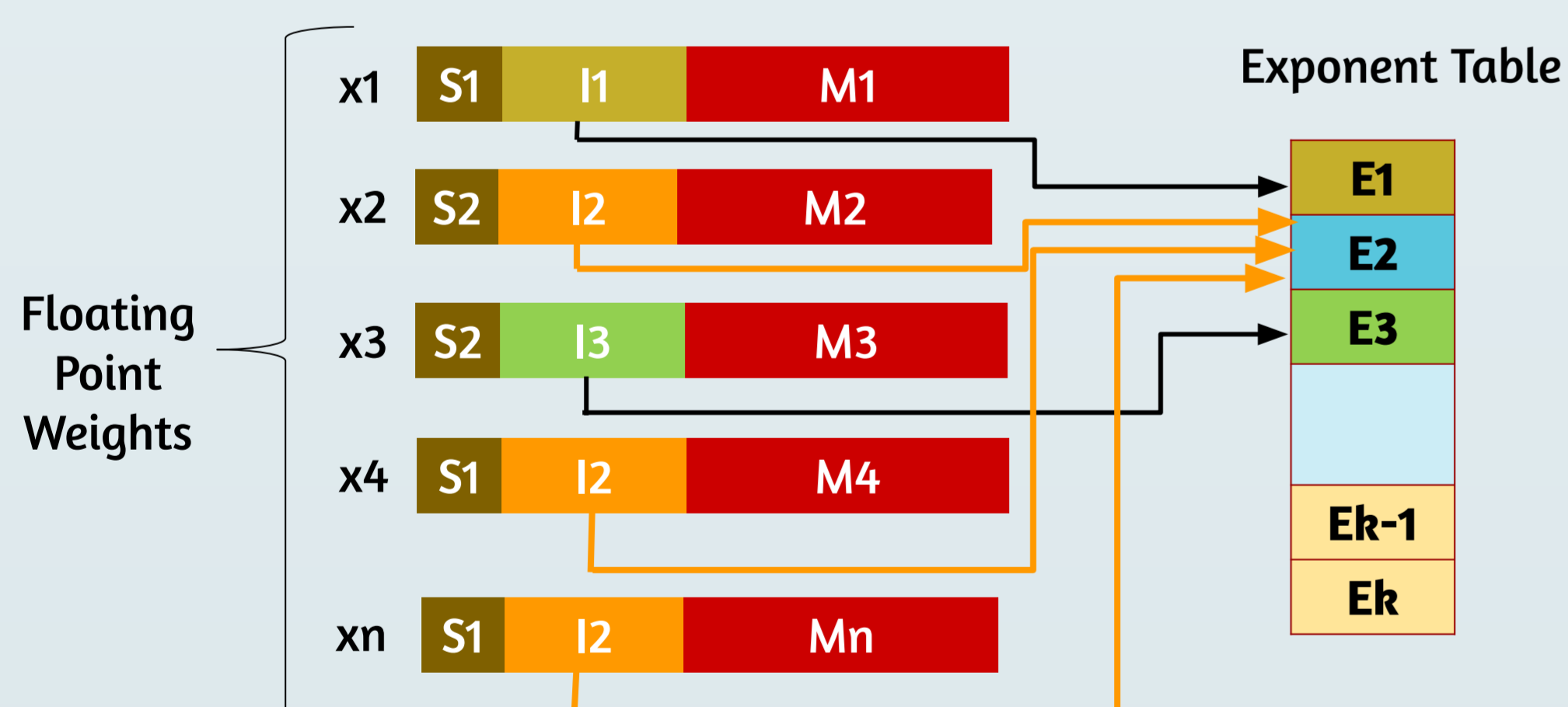
The existing model compression techniques, such as quantization, knowledge distillation, pruning, etc. reduce the size of the model by sacrificing some accuracy. We propose a new lossless model compression approach that makes use of the exponent distribution in weights for a trained model. We demonstrate how exponent sharing in weights can help reduce memory requirements. It is important to note that our proposed "exponent sharing in weights" method is different from the existing "sharing of weights" approach. Additionally, one can apply exponent sharing on top of other existing compression methods as well. We also discuss where our method stands in comparison with other model compression methods.

Exponent Frequency Distribution in LeNet Trained on MNIST (Float32)



Methodology

- There is a lot of repetition of exponents in floating point weights in neural network
- Storing exponents only once and referring from its all occurrences with the help of indexing saves memory



- The model compression is more if the degree of distinctness of exponents is more
- The model compression is more if the frequency of distinct exponents is more

Exponent Sharing in LeNet ((Float32))

Layer		Bits Required		Memory Saved	
		Original	Exponent Sharing	Bits	%
Local	Conu1	4800	4106	694	9.48
	Conu2	48000	42104	5896	
	Conu3	1536000	1392160	143840	
	FC1	322560	292456	30104	
	FC2	26880	23584	3296	
Global	Complete Model	1938240	1756698	181542	9.37

Note: Experiments are performed in Pytorch using Tesla T4 GPU

Expression

- For a model, if number of weights in a layer is n , then memory required for Float32 format is,

$$M_{orig} = n \times 32$$

- The memory required after exponent sharing is,

$$M_{comp} = n \times (s + i + m) + e \times k$$

- s = Sign bit
- i = Index bits ($\lceil \log_2 k \rceil$)
- m = Mantissa bits
- e = Exponent bits
- k = Number of distinct exponents

Mantissa Approximation during training

- Mantissa approximation can reduce the range of real values and in turn the range of exponents
- Mantissa approximation during training i.e Exponent Share Aware Training (in our case retraining, ESART) reduces the length of mantissa of floating point weights without any accuracy loss

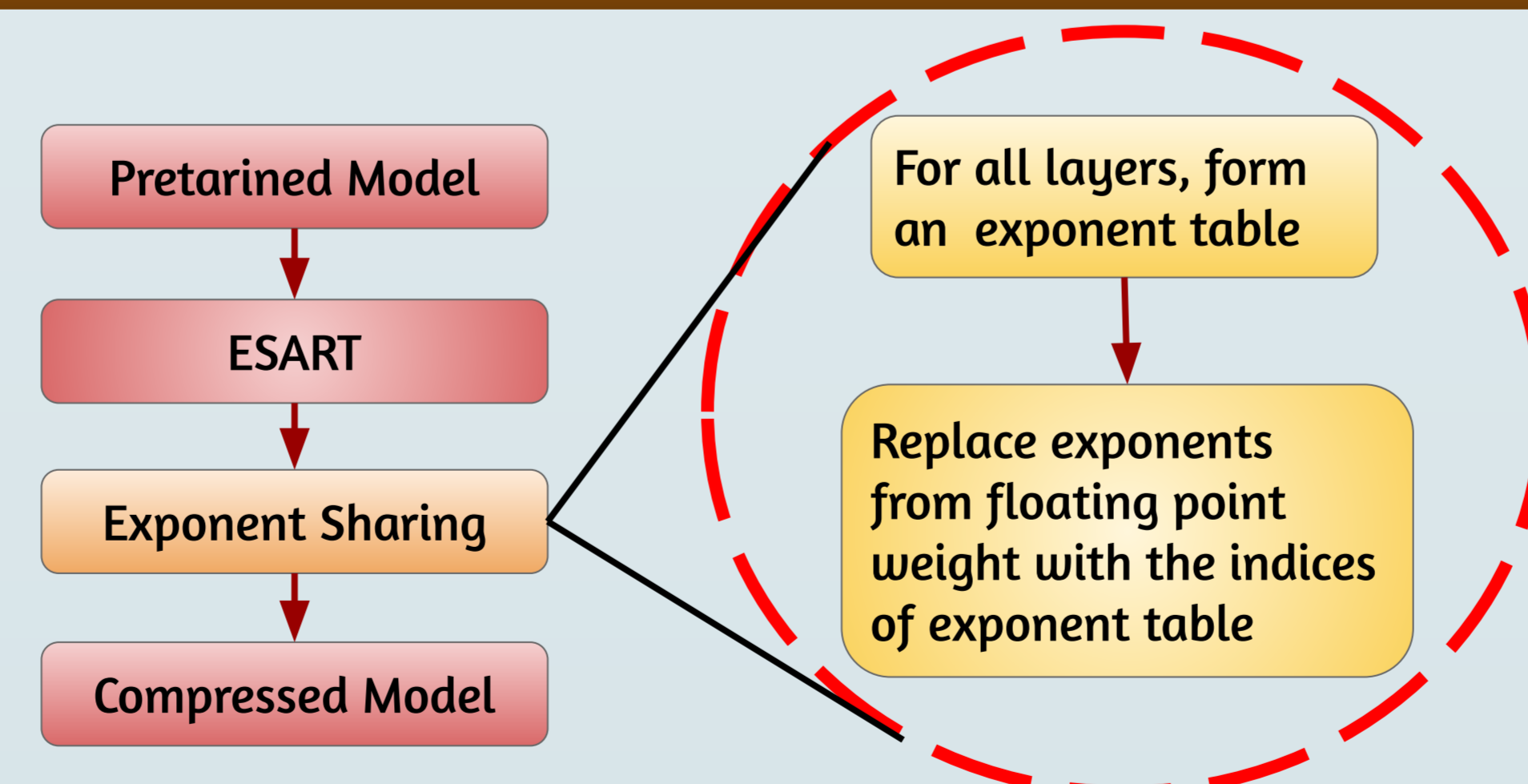
GeMMs with Exponent Sharing

- Consider a Generalised Matrix Multiplications (GeMM) of two matrices, Weights ($M \times N$) and Input ($N \times O$),
- if C_{orig} is the number of clock cycles before exponent sharing and $C_{expShare}$ is the number of clock cycles after exponent sharing then,

$$\text{On sequential architectures } C_{expShare} = C_{orig} + M \times N \times O$$

$$\text{On parallel architectures } C_{expShare} = C_{orig} + M \times O$$

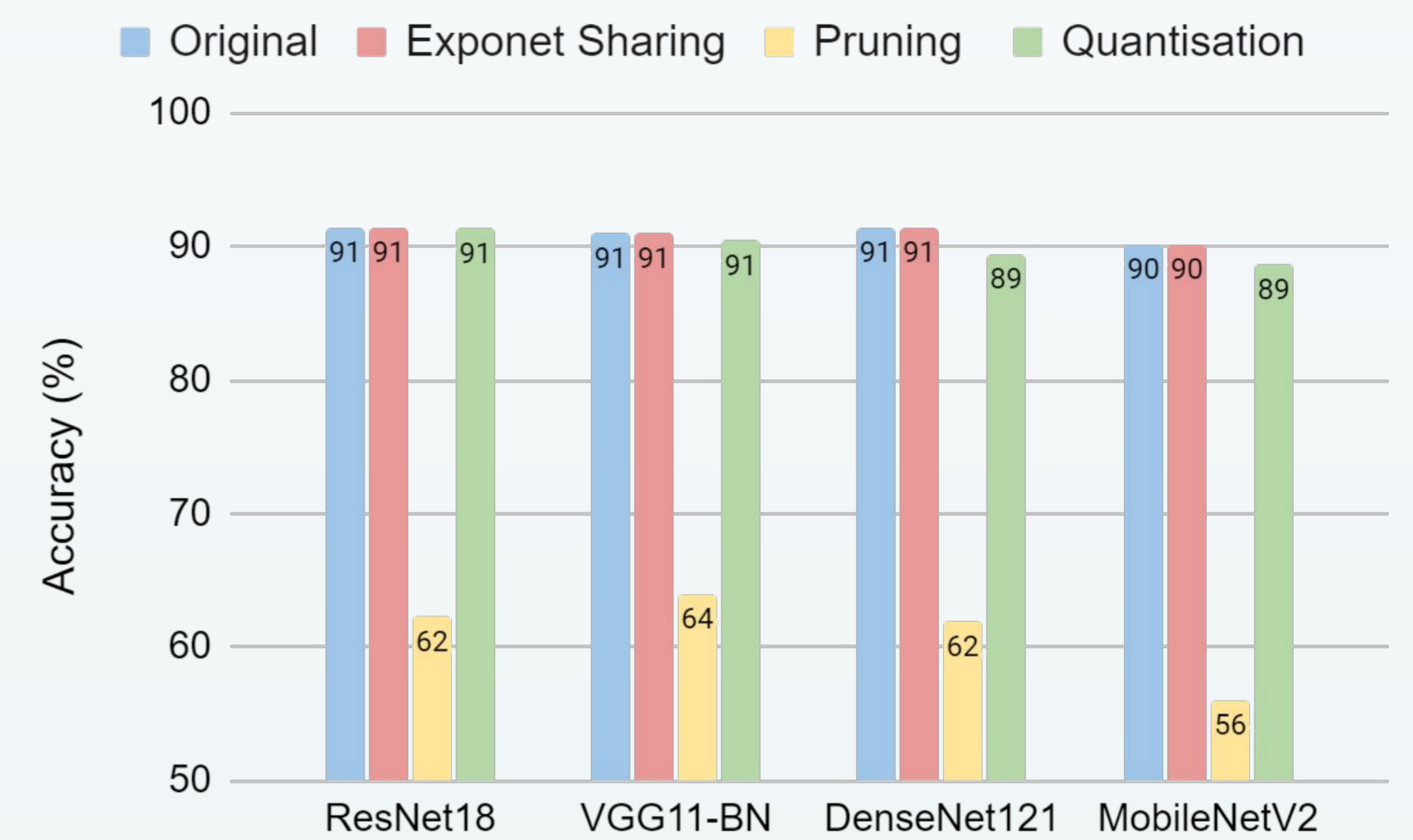
Flow



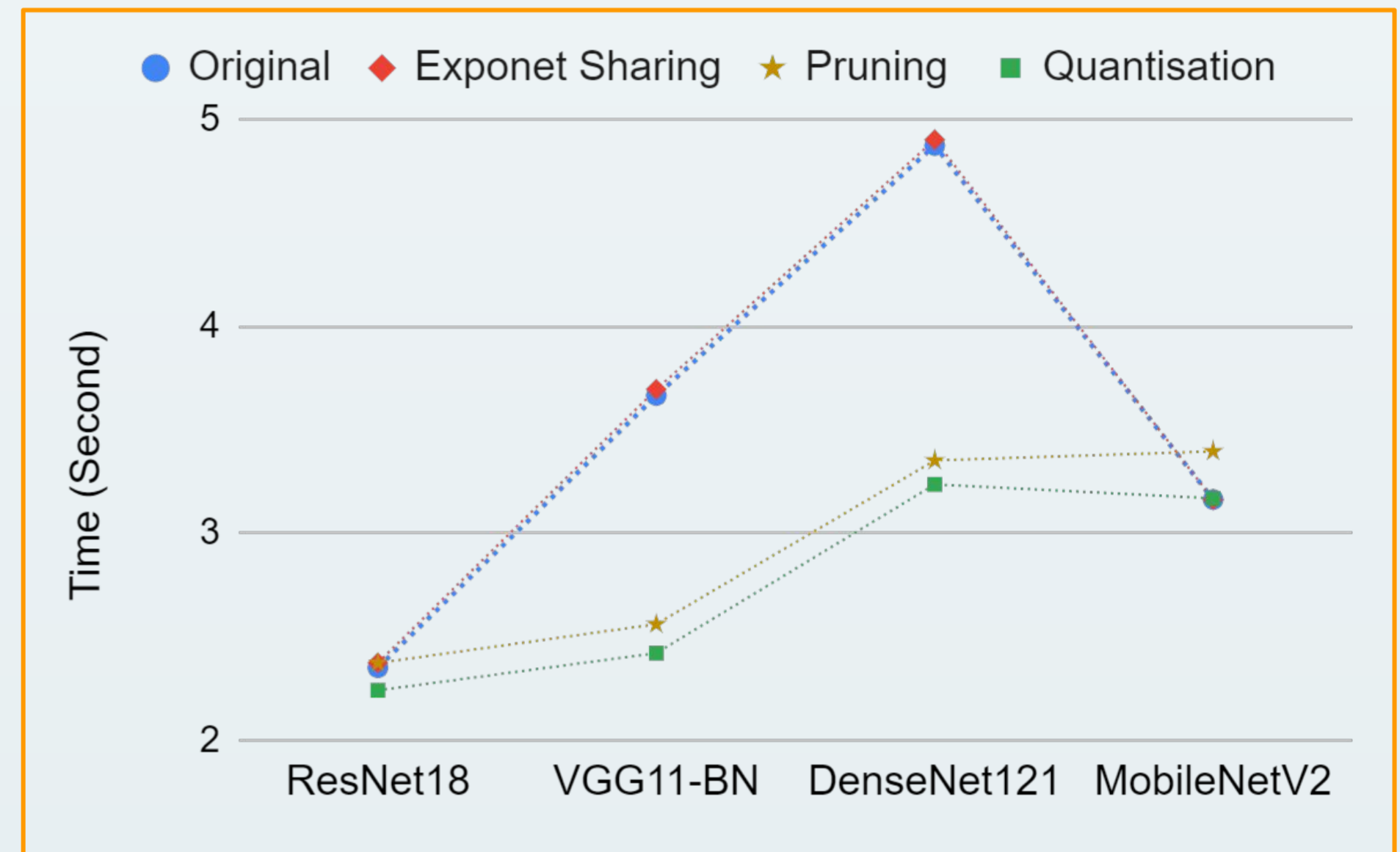
Exponent Sharing in STOA Models

Model	Model Size in BFloat16 in MB	
	Original	Exponent Sharing
ResNet18	21.30	14.22
VGG11-BN	53.66	34.89
DenseNet121	13.19	9.07
MobileNetV2	4.23	2.60

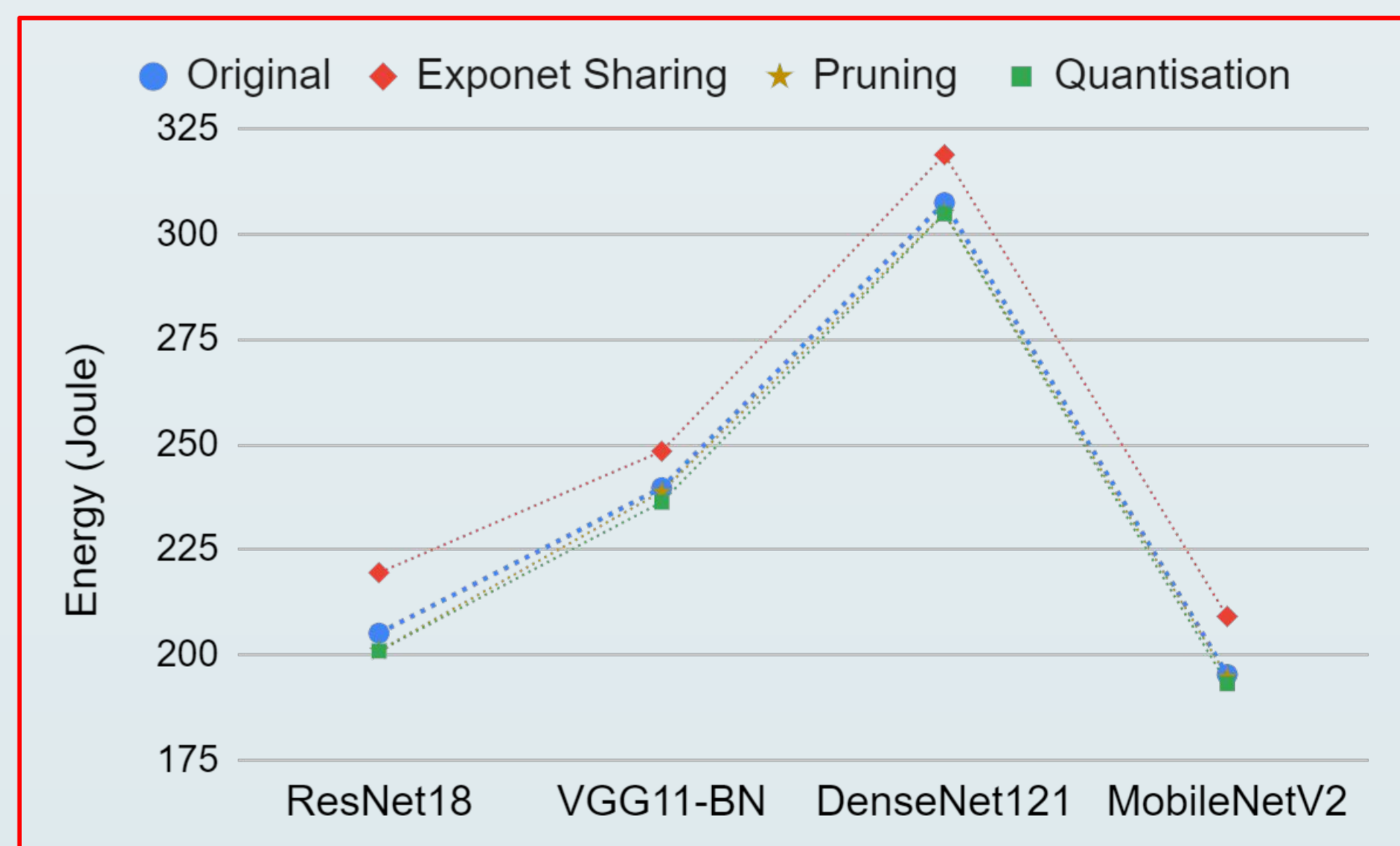
Accuracy after Exponent Sharing



Execution Time after Exponent Sharing



Energy after Exponent Sharing



Conclusion

- Exponent sharing compresses a model without any accuracy loss
- Layerwise exponent sharing yields better memory savings than sharing exponents in the all weights of a model
- The overhead in execution time can be reduced by parallel reading hardware design
- Exponent-Share-Aware- Training can improve memory savings up to 43% in BFloat16 floating point format
- There is slight overhead in execution time (1%) and energy requirement(5%), but it is comparable with other model compression methods due to no loss in accuracy

Acknowledgements

We thank Prof. Olivier Sentieys, Inria, Rennes, France for his valuable inputs for this work. This work is supported by DST-INRIA-CNRS Project IFC/4131/DST- Inria/2018-2019/1 through CEFIPRA, India.