Exploring Hardware Potential on ConvNeXt & ViTs

Michael Lu, Patrick Peng, Michael Youssef

***Abstract*—We aim to explore the performance of top image recognition models (ConvNext, ViT) along with a baseline (AlexNet, Resnet18) on an Eyeriss-like and simple output stationary accelerator with resource configurations ranging from budget to datacenter level. In doing so, we hope to analyze general trends and tradeoffs, through Timeloop simulation. In these simulations, we also experiment with reshaping the PE grid shape from vanilla Eyeriss while keeping PE count constant to optimize performance on some networks. We found that configuring the PE grid size to better account for convolutional layer sizes did lead to significant improvements in latency and utilization for ConvNeXt compared to the vanilla Eyeriss configuration. Furthermore, we found that modeling an Eyeriss-like accelerator with 3rd-generation Nvidia tensor core resource configurations across different tiers of accelerators highlighted the different behaviors and use cases of consumer vs. datacenter accelerators.**

I. Introduction

Today, Vision Transformers (ViT [3]) perform strongly on tasks classically handled by CNN-based architectures. However, these transformer models are more expensive to run, and may not have great opportunities for data reuse due to the usage of attention. At the same time, novel CNN architectures such as ConvNeXt [5], which offer much better opportunities for data reuse, have shown similar performance to ViT. In many cases, the deployment of some of these models, especially ViTs, may not be feasible due to hardware limitations, especially at the edge level. For example, while datacenters may have GPUs with high bandwidth and processing power, many potential users or even research clusters may only have RTX 3060-level GPUs or similar in their personal computer, making the analysis of the tradeoff between model performance, latency, energy cost, and utilization an important one to study.

Hence, in this paper we contribute the following: we construct simulated architectures for multiple deployment types and accelerator designs in Timeloop, we analyze the performance of various models upon these architectures, we explain the implications of these results, and we describe promising future avenues of exploration based on our results.

II. Models

In this section we briefly explain each model we evaluate our architectures on and the purpose for its inclusion.

A. AlexNet

AlexNet [4] serves as a baseline for our evaluations; although it is shallower than the others, it is a performant, foundational model for deep CNN networks in image classification, and influenced a large number of later works in general deep learning tasks around vision. Additionally, although it was computationally expensive at the time, it was made more feasible by using GPUs for its training, making it appropriate for this study.

B. ViT

A much more recent model, ViT is an application of the Transformer architecture [7] to the image classification problem, and is the best performing out of all of our evaluated models. However, due to a large model size and the usage of attention, it is computationally expensive to run even by today's standards, making it an interesting counterpoint to earlier models such as AlexNet.

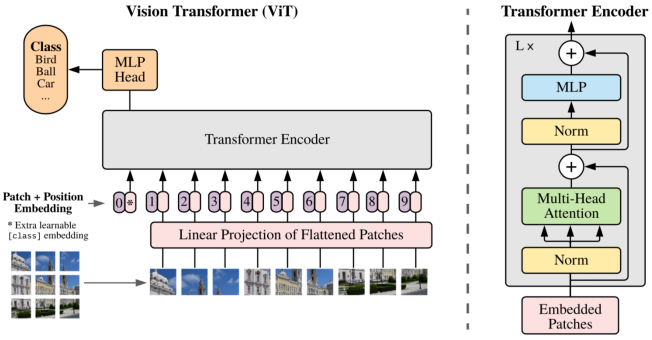

Fig. 1. ViT model architecture, visualized [3]

C. ConvNext

Developed in 2022, ConvNeXt was a response to the popularization of Transformer architectures, proving CNN networks could still remain competitive. The architecture features blocks that emulate the ideas of transformer blocks while using CNN layers; specifically ones with a 7 by 7 kernel.

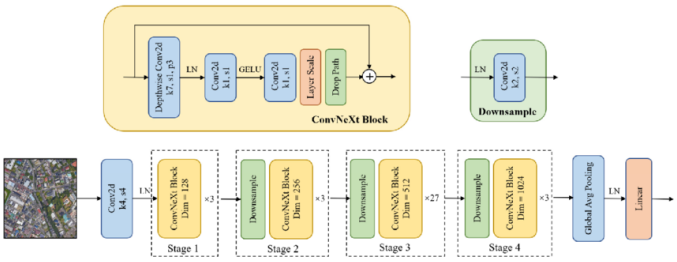

Fig. 2. ConvNext model architecture, visualized [1]

D. Model Comparisons

For some more concrete comparisons of these models, we present accuracies on common datasets for image classification, gathered from the papers themselves (for Imagenet-1k) and through aggregating various sources online (for MNIST):

	ImageNet-1k, trained on full dataset	MNIST
AlexNet	61.0%	∼99.0%
Resnet18	71.0%	∼99.1%
ConvNeXt	87.8%	∼99.5%
ViT	88.55%	∼99.7%

Fig. 3. Accuracy on Imagenet-1k, MNIST for AlexNet, Resnet18, ConvNeXt, ViT [4], [6], [5], [3]

Note that as AlexNet was released much earlier, the version of Imagenet it was evaluated on was slightly different from the version the later two were evaluated on, but the general trends are still the same.

III. Accelerator Designs

For this study, we focused on two accelerator designs: an Eyeriss-like accelerator, and a simple output stationary accelerator.

A. Output Stationary Accelerators

Output stationarity in an accelerator is a paradigm where the accelerator accumulates all the results for a single output cell before writing it back to DRAM. This creates speedups by significantly reducing the number of DRAM reads/writes of partial sums, as well as allowing 'hoists' of elements out of inner loop layers, reducing overall accesses. However, the speedups are all relative to certain layer sizes; generally, if the filters or the inputs are much smaller than the output dimensions, output stationarity does not provide as much of a speedup when compared to other paradigms such as input stationarity. However, in practice output stationarity is commonly used, as it provides performance improvements in many cases. As such, it was only fitting to include this design in our study.

B. Eyeriss-like Accelerators

Eyeriss is a well-known accelerator design introduced in [2], which adopts the paradigm of "row stationarity", a strategy designed specifically for CNN. We recommend reading the cited paper for the full details; but we aim to provide a brief summary of the paradigm here. In essence, "row stationarity" is described as running 1-dimensional CNN primitives in parallel, where each operates on one row of filter weights and one row of input filter map pixels to generate partial sums. These are then accumulated to generate output filter map pixels. By allocating these 1-dimensional primitives to individual PEs, each PE is able to reuse much more CNN data and accumulate psums more efficiently (the authors describe this as "sliding window processing"), reducing the drawbacks that may initially seem present in this strategy. Overall, Eyeriss was a foundational accelerator design, and provided significant improvements over industry GPUs at the time, motivating its inclusion in this study.

IV. Methods

Using Timeloop, we simulated per-layer performance of AlexNet, Resnet18, ConvNeXt, and ViT on an Eyeriss-like and a simple output stationary accelerator as described previously. Within each accelerator design, we explored different PE mesh grid sizes along with the vanilla Eyeriss configuration (14x12 PE mesh grid, 64x16 memory width) to see if changing the resource allocation without increasing the total number PEs or the memory width could yield performance increases. Specifically, we choose to simulate 8x21 and 21x8 grid sizes. Finally, we also adapted the resource configurations of the two accelerators to match a 1st and 3rd generation tensor core with memory bandwidths corresponding to a V100 (1st-gen), 3060 (3rd-gen), 3090 (3rd-gen) and A100 (3rd-gen) (Fig. I). For these GPUs, we abstract away SMs and other overhead on the card, as the most appropriate comparison for Eyeriss would be a tensor core. For our PE-grid sizes we doubled the FMA/s since each FMA requires two FLOPs which aligns with Timeloop's output stats. Additionally, we choose approximately square mesh configurations as we were not able to find information on the exact arrangement. For each simulation, we used a datawidth of 16 and generated associated roof line graphs and calculated total and averaged layer metrics.

	Fused-Multiply-Adds (FMA/s)	Memory Bus Width
V100	64	4096
3060	256	192
3090	256	384
A100	256	5120

TABLE I
1st and 3rd Generation Tensor Core Accelerator Configurations

V. Results & Discussion

A. Eyeriss vs Output Stationary

Alexnet, being a simpler and shallower model, was heavily compute-bound but had relatively high measured utilization. Resnet18 and ConvNeXt have similar utilization profiles with a group of low utilization dense layers closer to the roof line pivot and another group of higher utilization CNN layers. ConvNext has more dense layers due to the repeated transformer-like blocks. Finally, ViT has a greater variance in utilization across its layers with the lowest average utilization. This is most likely due to the large matrix multiplication operations found within the attention blocks which are not optimal under a CNN-focused architecture such as Eyeriss (Fig. 4).

Moving down the plots, we can see that changing the mesh grid size to 8x21 kept or increased utilization of the dense and CNN layers across all four models with the larger improvement seen in ConvNeXt. We hypothesize that this is due to ConvNeXt primarily using 7x7 filter sizes, which may have a decent reuse benefit with a 8x21 grid compared to a 12x14 grid. That being said, all four models were compute-bound on the Eyeriss-like design with 168 total PEs (Fig. 4).

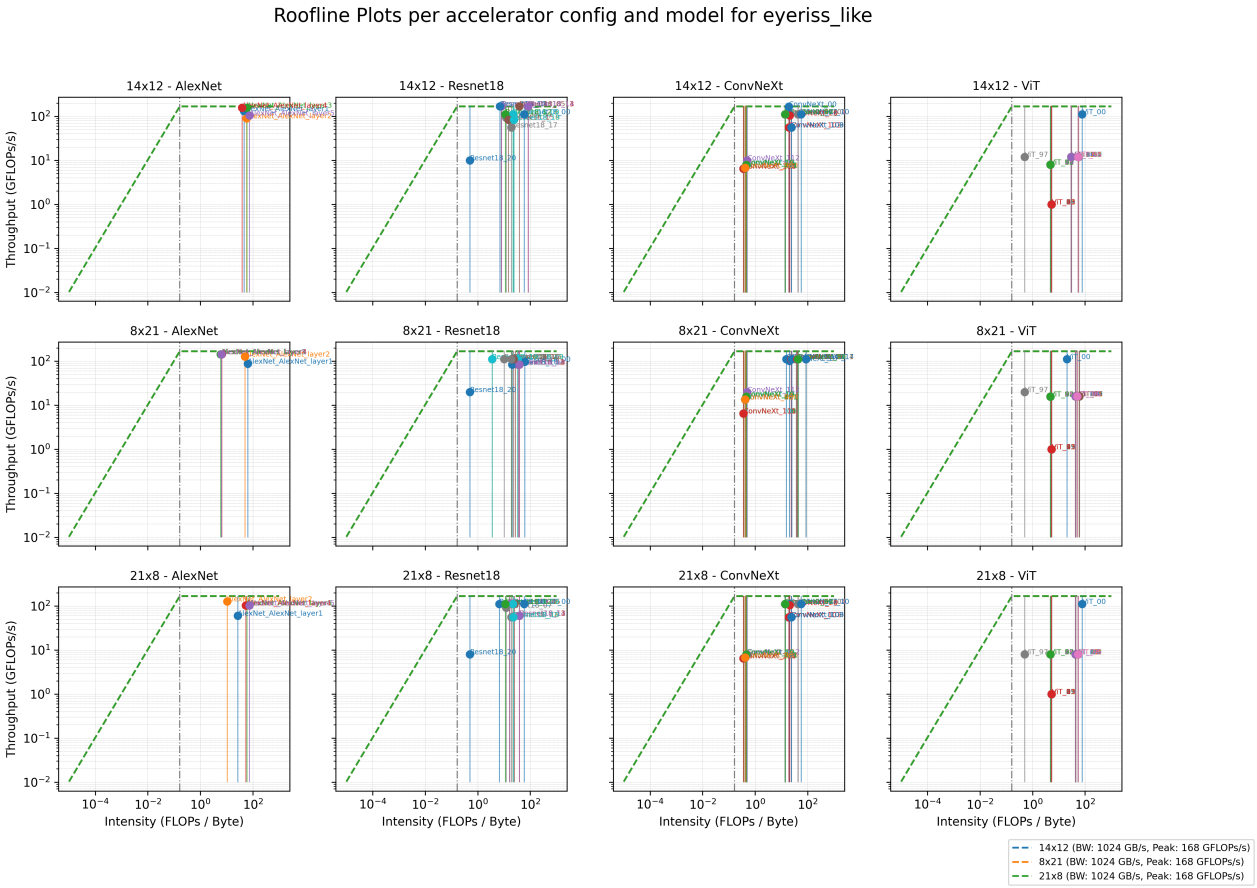

Fig. 4. Eyeriss-like Results for AlexNet, Resnet18, ConvNeXt, and ViT using 14x21, 8x21, and 21x8 PE Grid and 64x16 memory width. One vertical line per simulated layer with measured utilization marked with a dot. Grey dotted line provided to marke pivot point form memory- to compute-bound performance.

We also ran the same PE-grid configurations on a simple output stationary design (Fig. 5). Compared to the Eyeriss results, the dense layers performed similarly with lower utilization. However, the output stationary design lags in CNN layer performance due to the decreased reuse of the filters. As such, this design dramatically decreases overall utilization for CNN-based models while maintaining a similar profile for the ViT's layers.

B. 1st- and 3rd-generation-like Tensor Core Performance

As all of our models were entirely compute bound using the vanilla Eyeriss resource allocations, we also wanted to investigate what performance would look like using real world accelerator resource configurations (Fig. I).

Again, due to its relatively simple and shallower architecture, AlexNet was fully compute bound across all modeled accelerator configurations. However, compared to the vanilla Eyeriss resources (168 PEs, 64x16 memory width), we were able to get closer to the pivot point which indicates that an Eyeriss-like accelerator with 3060-like resource allocations may be better suited for these types of workflows. And, as we move across the grid to Resnet18 and ConvNeXt, we can see that our clusters of linear layers go from compute bound to memory bound while still maintaining near full utilization in our CNN-based layers. The 3090-like 3rd-gen tensor core configuration maintains this trend despite the increased memory bus width, moving the linear layers closer to the pivot point while still maintaining near full utilization (Fig. 6). This suggests that in an Eyeriss like accelerator, the 3090's resource allocation is nearly perfect for non-CNN-based workflows and still reaps the benefits of Eyeriss' CNN-focused design in terms of utilization. While not a direct comparison to the real world accelerators, this mirrors the 3090's success as a high-end accelerator option for non-datacenter workstations.

The V100, on the other hand, achieves higher relative utilization compared to the 3060 and 3090. However, the lower FMA/s performance paired with a higher memory bus width leads to all the models being fully compute bound once again (Fig. 6). We believe that this is likely due to the different use case of a datacenter accelerator like the V100 which has significantly more total VRAM (32GB) compared to the 3060 and 3090 (12GB and 24GB respectively). The layer sizes and problem configuration used for Timeloop assumed a batch size of 1 which is reflective of an edge deployment scenario such as on a personal workstation (3060 or 3090). However, in a datacenter, the accelerator would likely be tasked with

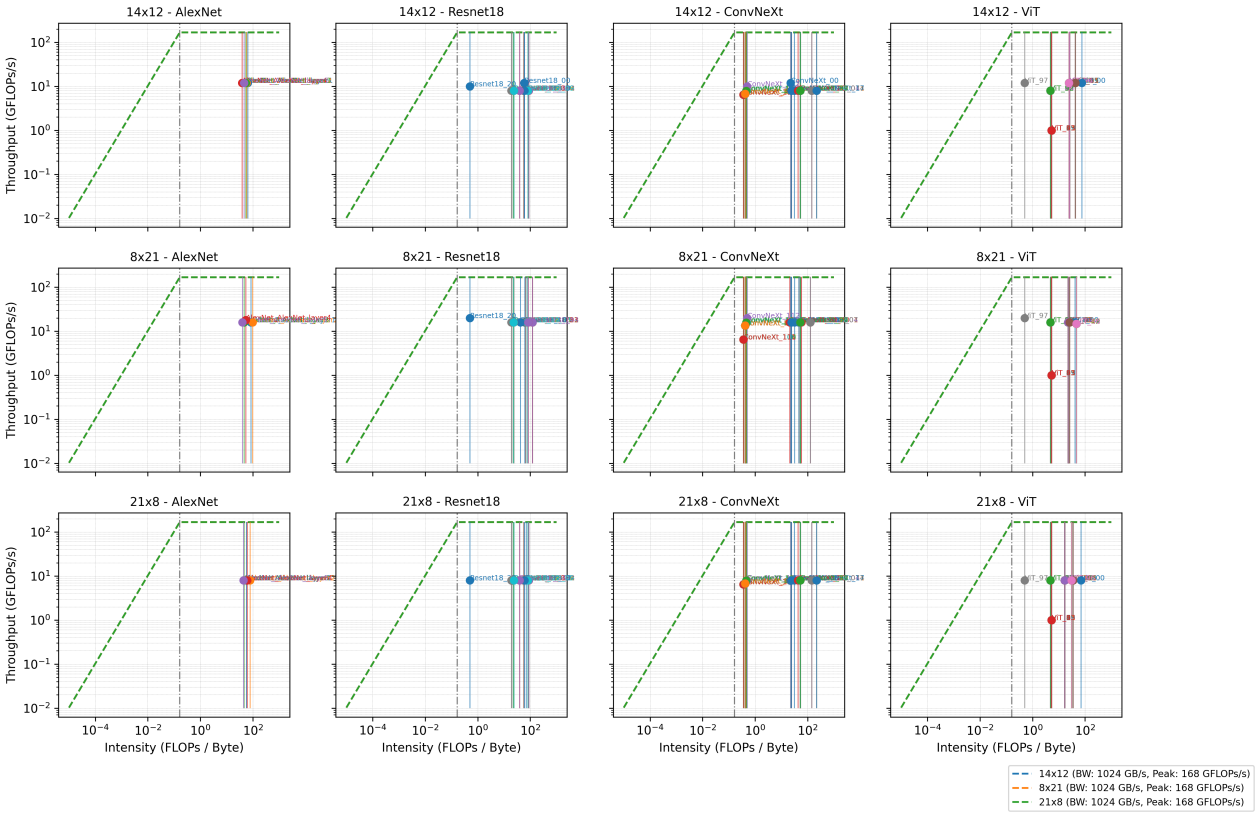

Fig. 5. Simple Ouptut Stationary Results for AlexNet, Resnet18, ConvNeXt, and ViT using 14x21, 8x21, and 21x8 PE Grid and 64x16 memory width

handling large batches data which would take better advantage of its larger memory bus width and potentially move the simulated performance closer to or even into the memory-bound region. The A100, despite its use of 3rd-generation tensor cores, also demonstrates this trend with the simulated models all being entirely compute bound once again (Fig. 6).

TABLE II
PERFORMANCE METRICS FOR ALEXNET ACROSS ACCELERATORS

Accelerator	Cycles	Energy	Cycles/layer	Util./layer	Energy/layer
3060	3.52M	11649.48	879617.8	59.71%	2670.52
3090	3.52M	11649.48	879617.8	59.71%	2670.52
V100	11.71M	11708.38	2.8M	71.85%	2670.52
A100	3.52M	11649.71	879617.8	59.71%	2670.52
14x12	9.61M	9601.19	3.17M	66.69%	2729.77
8x21	8.33M	14973.00	2.19M	76.92%	3455.66
21x8	10.29M	11988.26	2.34M	62.29%	3105.00

C. Layer Performance Statistics

Along with generation roof line visualizations of the different models and their respective layers, we also calculated summary statistics for each simulation. Specifically, we include total cycles, total energy, averaged cycles per layer, averaged utilization per layer, and averaged energy per layer. Averaging was done with a weighted average by the number of cycles each layer ran for.

TABLE III
PERFORMANCE METRICS FOR RESNET18 ACROSS ACCELERATORS

Accelerator	Cycles	Energy	Cycles/layer	Util./layer	Energy/layer
3060	5.83M	26273.9	370054.16	60.79%	1420.99
3090	5.83M	26273.9	370054.16	60.79%	1420.99
V100	18.96M	23793.6	1.17M	74.76%	1380.99
A100	5.83M	26273.9	370054.16	60.79%	1420.99
14x12	14.76M	21814.73	950767.18	73.14%	1263.62
8x21	18.95M	23793.60	1.17M	56.99%	1381.43
21x8	22.59M	20642.01	1.50M	47.80%	1202.00

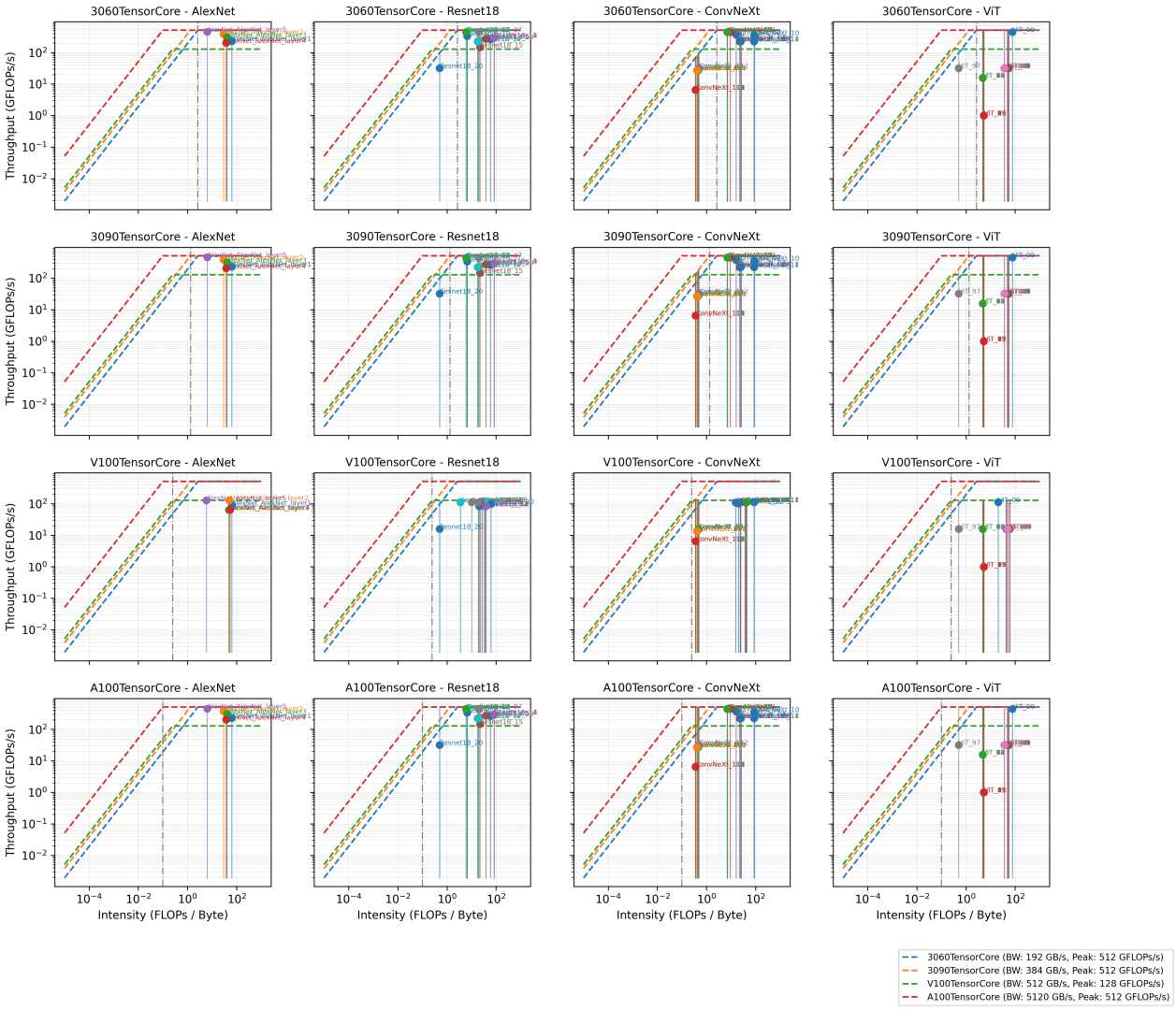

Fig. 6. Eyeriss-like Results for AlexNet, Resnet18, ConvNeXt, and ViT using 3060, 3090, V100, and A100 Tensor Core Resource Configurations

TABLE IV
PERFORMANCE METRICS FOR CONVNEXT ACROSS ACCELERATORS

Accelerator	Cycles	Energy	Cycles/layer	Util./layer	Energy/layer
3060	250M	1496811.9	7.61M	71.14%	38100.5
3090	250M	1496811.9	7.61M	71.14%	38100.5
V100	812.3M	825326.7	22.40M	87.48%	22743.2
A100	250M	1496811.9	7.61M	71.14%	38100.5
14x12	880.6M	1132939.1	25.83M	61.49%	30814.28
8x21	812.3M	825321.4	22.40M	66.66%	22743.54
21x8	880.9M	1132939.5	25.82M	61.47%	30805.42

TABLE V
PERFORMANCE METRICS FOR VIT ACROSS ACCELERATORS

Accelerator	Cycles	Energy	Cycles/layer	Util./layer	Energy/layer
3060	528.9M	149842.6	10.78M	6.23%	3083.73
3090	528.9M	149842.6	10.78M	6.23%	3083.73
V100	1052.9M	218749.0	21.64M	12.51%	3211.27
A100	528.9M	149842.6	10.78M	6.23%	3083.73
14x12	1401.3M	154887.9	28.91M	7.16%	3118.66
8x21	1052.5M	156057.3	21.65M	9.53%	3211.30
21x8	2098.5M	147805.0	43.43M	4.78%	3029.26

As shown in the tables, the simulated stats remain constant for all the 3rd generation tensor cores regardless of memory bus width as we never reach the performance roof line even for the 3060. This result aligns with our roof line graphs which show that in all situations, our accelerators are not

fully utilized. As such, the more interesting results come when looking at the change in performance for a given model when we vary the PE grid sizes of the Eyeriss-like configurations (14x12, 8x21, and 21x8) and when we compare performance across models given their accuracy or 'model quality' as context. Specifically, while the 8x21 grid size has the lowest latency, lowest energy, and highest average utilization for ConvNeXt and ViT, the vanilla 14x12 grid size performs best on Resnet18 and Alexnet. This mirrors the trend we saw in our roof line plots which suggest that the 8x21 grid size yields a significant utilization improvement for the linear layers while also have an outsized performance boost for the CNN layers in ConvNeXt. Again, this seems to be due to ConvNeXt's use of 7x7 filters which fit the 8x21 grid size better and may yield better reuse. On the otherhand, Alexnet and Resnet18 primarily use 3x3 filters which may lead to increased parallelization bottlenecks when reducing the X-dimension of the grid size. Specifically, upper right cluster of CNN layers in Fig. 4 shows that the 8x21 grid yields higher utilization of the ConvNeXt CNN layers than the Resnet18 or AlexNet CNN layers while providing a similar boost in utilization across all linear layers. As models like ViT and ConvNeXt represent newer, higher quality models, this suggests that re-evaluating existing accelerator designs might yield quick model efficiency improvements without the need for more PEs or improvements in memory efficiency.

The results also highlight Eyeriss' CNN-focused design with ConvNeXt, a fully CNN-based transformer-like model, having less than half the latency, over 10x the average utilization per layer, and around the same total energy as ViT. As both of these models have very similar performance in terms of accuracy, running ConvNeXt may dramatically reduce latency while also better utilizing available hardware resources. This is especially true if the deployment leverages hardware architectures that are optimized for CNN-based computations.

VI. Future Work

Considering the results we have previously presented, we acknowledge that there are many possible avenues for future work. We list some of these here.

A. Analysis of GELU and Sigmoid Activations on Efficiency

ViT models feature attention computations, which we found difficult to model properly in Timeloop. Although the attention matrix computations were accounted for, the usage of sigmoid activations in that process was not. Similarly, ConvNeXt features GELU activations, which were also not modeled in our analysis. Although we ignored these, the inking their impact would be negligible, this would be an important consideration in a more thorough analysis.

B. Sparse Optimizations

Sparse optimizations such as gating and compression would also be valuable areas to explore when determining the difference in efficiencies between ConvNeXt and ViT, as they are commonly used in architectures today. For example, ViT attention maps may be sparse, and sparse optimizations could save energy and cycles when used on these layers. On the other hand, ConvNeXt uses GELU instead of ReLU, making outputs less sparse and reducing the possible improvement from sparse optimizations. We tried implementing gating in Timeloop for the Eyeriss-like accelerator. However, we were not able to get this component running for our project report.

C. Heterogeneous Architectures

Throughout this study, we have mostly simulated models running on only one accelerator or only the other, where one accelerator was designed for CNNs (Eyeriss) and one was designed more generally (output stationary). With a heterogeneous architecture (possibly composed of two sub-accelerators similar to the two accelerators we analyze here) we may be able to optimize model inference efficiency even more. However, this would require mapping workloads to the appropriate subaccelerator, which is a new problem by itself.

D. Quantization

Another aspect of optimization that could be explored is quantization. While this can have impacts on the accuracy of models, that is a quality end users may be willing to exchange for higher efficiency in energy and latency. Studying this would require evaluating the accuracy of our models under quantization, which becomes a more complex problem when some approximations (such as the GELU approximation) become trickier due to numerical instability.

Acknowledgements

Thank you to Professor Pellaur and our TA Tianxi Lu for their advising and guidance on this project and throughout the semester!

References

[1] S. Chen, Y. Ogawa, and Y. Sekimoto, "Large-scale individual building extraction from open-source satellite imagery via super-resolution-based instance segmentation approach," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 195, pp. 129–152, 01 2023.

[2] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, 2016, pp. 262–263.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[5] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022. [Online]. Available: https://arxiv.org/abs/2201.03545

[6] PyTorch Team, "ResNet-18 — Torchvision Main Documentation," 2024, accessed: 2025-05-11. [Online]. Available: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf