A. TPU/GPU-optimized Search Space Details

As described in Section 3, our search space is based on the factorized search space used in EfficientNet [57] with three major optimizations for TPUs and GPUs. These optimizations include accelerator-friendly space-to-depth/batch, fused convolution structures, and block-wise activation functions. In addition to these optimizations, we also added a few more dimensions to Ops in the baseline search space, such as more choices on SE ratios and convolution kernel sizes. Table 4 shows the details for the search space.

B. Ablation study on the DC accelerator optimized search space and the searched EfficientNet-X-B0 base model

As summarized in Section 5.1 and Section 6, all enhancements in the DC-accelerator-optimized search space (Section 3) contribute to improving accuracy-latency trade-offs in the searched base model — EfficientNet-X-B0. Table 5 shows the detailed ablation study on how these new model architecture components, including space-to-depth, fused convolution structures, and block-wise searchable activation functions, improve accuracy-latency-Pareto results over the baseline EfficientNet-B0. The Space-to-depth and fused convolution structures improve both the accuracy and speed on TPUv3 and GPUv100. The trends on the total FLOPs further confirms our analysis on new search space about activation functions as described in Section 3 and Section 5.1. Concretely, although activation functions have negligible impact on total model FLOPs on TPUs and GPUs, they have big impact on performance. On GPUv100, NAS selects ReLU activation for all layers/blocks for EfficientNet-X-B0 because of the performance degradation caused by non-fused swish. On TPU, NAS selects ReLU for blocks with depthwise convolutions and swish for blocks with vanilla convolutions to avoid overloading the vector units in TPUv3 as described in Section 5.1. As a result, the new activation function strategy improves speed but causes accuracy drop on both GPUv100 and TPUv3. However, thanks to the accuracy improvements from space-to-depth and fused convolutions, the final accuracy is comparable to the baseline EfficientNet-B0 on both TPUv3 and GPUv100 as shown in Table 5. The hybrid ReLU and swish activation functions on TPUv3 leads to the higher accuracy than the ReLU-only activation functions on GPUv100. Note that in Table 3, we report the lower accuracy from TPUv3 and GPUv100 as the final score.

On TPUv3, all new enhanced search space components contribute almost equally in inference speed, with the new activation function strategy offsetting some of the accuracy gains. On GPUv100, the new activation function strategy causes a more significant inference speedup than other new model architecture enhancements, but with a bigger accuracy drop than on TPUv3. This demonstrates the impact of the software stack. We believe a fused swish implementation for GPU software will make GPUv100 behave similar to TPUv3.

C. Ablation study on latency-aware compound scaling and the EfficientNet-X family

As summarized in Section 5.2 and Section 6, LACS achieves a better set of scaling factors than single-objective compound scaling that originally proposed in the EfficientNet [57] work. Clearly, searching for scaling coefficients at a lower target latency level (*e.g.*, EfficientNet-X-B1) and using them to create higher latency models (*e.g.*, Efficientnet-X-B7) is much more cost-effective than directly searching for coefficients at the higher latency model level (*e.g.*, Efficientnet-X-B7). However, searching first at low latency level models and scaling to high latency level models has the potential to deviate from the empirical optimum from direct searching at high latency level models, due to non-linear increases of accuracy and latency with larger depth, width, and resolution. In this ablation study, we first verify the efficacy of LACS in maintaining good scaling from small to large models, without deviation from the empirical optimum. We then provide more comparisons on results from LACS and single-objective compound scaling.

To verify the efficacy of LACS, we target the B7 model level of the EfficientNet-X family on GPU and compare the scaling factors yielded by LACS at X-B1 level and then applied at X-B7 level against direct accuracy-latency-Pareto-search at the X-B7 level to find the empirical optimum coefficients. As shown in Table 6, both the scaling coefficients and the resulting network dimensions are quite similar. Particularly, the network dimensions are within 6% of each other. This verifies that LACS can effectively scale up all the way to the high end models to form a model family, with negligible deviations from empirical optima.

With the verified efficacy of LACS, we present detailed comparisons on model dimensions of EfficientNet-X on TPUv3 and GPUv100 with the scaling factors obtained by LACS and by the original single-objective compound scaling as used in EfficientNet [57]. We first run single-objective compound scaling that uses accuracy as the sole objective as proposed in [57]. Even with the new EfficientNet-X-B0 as the base model, the single-objective compound scaling method finds the same compound scaling factors as with EfficientNet. On the other hand, LACS finds different compound scaling factors on TPUv3 and GPUv100. Table 2 shows these different scaling factors obtained from LACS and single-objective compound scaling. Note that since single-objective compound scaling only uses accuracy as the sole objective, unlike LACS, it does not generate different scaling factors for TPUv3 and GPUv100. Table 7 shows the detailed model dimensions generated by these different scaling factors. While LACS creates different families for TPUv3 and GPUv100, the most notable difference is

Op & layer	Searchable Architectures and Dimensions
Stage size	Total number of stages, with multiple layers per stage
Layers in each stage	Each layer can a simple Conv2D layer or a complete block such as MBConv
Convolution	**Types**: Conv2D, DepwiseConv2D, MBConv, fused MBConv, ResidueBottleneckBlock **Kernel size**: 3x3, 5x5, 7x7 **Stride**: 1, 2, 4 (Stride-2/4 are only allowed in the first layer of a stage, if chosen.) **Expansion Ratio** (for MBConv and FusedMBConv): 1, 3, 6
Reshape	Space to depth w/ stride-n Conv2D_NxN Space to batch w/ memory-intensive copy-reshape ops
Activation function	ReLU, swish, searchable at the block level instead of at each layer level
SE ratio	0 (i.e., no SE layer), 1.0, 0.5, 0.25. 0.125
Skip connections	none, identity with pool and/or Conv2D-1x1 are used when feature map tensors mismatch

Table 4: Complete Search Space Details

Model	Top-1 Accuracy (%).* (TPUv3 / GPUv100)	#Params (Million)	#FLOPs† (Billion)	I¶ (Ops/Byte)	E‡	Inference Latency§(ms) (TPUv3 / GPUv100)
EfficientNet-B0 [57]	77.3	5.3	0.39	19.7	52.4%	13.4 / 38.1
+SpaceToDepth	77.5	7.2	0.47	25.3	55.8%	11.9 / 35.6
+Fused Conv	77.8	7.6	0.91	62.5	56.1%	9.5 / 30.5
+Activation (EfficientNet-X-B0)	77.4 / 77.3	7.6	0.91	63.8	57.3%	8.7 / 22.5

Table 5: Contribution breakdowns of each enhanced model architectures to inference latency and accuracy on imagenet of the searched based model EfficientNet-X-B0 on TPUv3 and GPUv100. TPUv3 and GPUv100 results are separated by "/" when they differ, shown as "TPUv3 results / GPUv100 results". *Only with the different activation function selections, accuracies differ on TPUs and GPUs. †Following common practices, #FLOPs refer to #multiply-and-add operations. ¶I is the operational intensity measured on TPUv3. ‡E is the execution efficiency measured on TPUv3, w.r.t to roofline instead of peak hardware FLOPs/sec as shown in Equation 1. Only in the compute-bound region as shown in Figure 2, the roofline and peak hardware FLOPs/sec are the same. §The inference latency are measured for inferencing 128 images on TPUv3 and GPUv100, with mini batch size of 128. Models run in FP16 mode on GPUv100.

LACS search level	†Coefficients α, β, γ	X-B7 Dimensions
LACS at X-B1 level	(1.29, 1.16, 1.07)	(Depth: 75, Res: 368)
LACS at X-B7 level	(1.29, 1.14, 1.08)	(Depth: 79, Res: 350)

Table 6: Scaling coefficients α, β, γ and model dimensions yielded by LACS at low (X-B1, *i.e.*, EfficientNet-X-B1) level and directly at high (X-B7, *i.e.*, EfficientNet-X-B7) level on GPUv100. α, β, and γ are the base exponential terms to be used together with ϕ as described in Equation 5. Depth means total number of layers in the network. Res means the input resolution. Both scaling coefficients and model dimensions (depth, input resolution) produced by the methods are quite similar.

that both LACS versions prefer deeper and slimmer models as compared to original single-objective compound scaling, with the LACS results on GPU and TPU being 60% $\sim$ 70% deeper with $\sim$40% smaller input resolutions. The changes in scaling and the resulted model architectures are caused by the use of the accuracy-latency multi-objective that provides more visibility into the hardware architecture details. As a result, EfficientNet-X has much faster inference speed, with comparable accuracy to EfficientNet as shown in Table 3 and Figure 3.

EfficientNet-X Model	Single-obj scaling	LACS on GPU	LACS on TPU
X-B0	(16, 224)	(16, 224)	(16, 224)
X-B1	(17, 240)	(17, 229)	(17, 229)
X-B2	(19, 260)	(20, 241)	(20, 243)
X-B3	(22, 300)	(25, 258)	(26, 263)
X-B4	(28, 380)	(36, 289)	(38, 298)
X-B5	(35, 456)	(49, 317)	(52, 331)
X-B6	(41, 528)	(62, 343)	(68, 361)
X-B7	(49, 600)	(79, 368)	(87, 391)

Table 7: Comparison on depth (*i.e.*, layer count of the network) and input image resolution of EfficientNet-X model family with different compound scaling factors designed by LACS and single-objective compound scaling. Each result contains a pair of "(depth, input image resolution)". since single-objective compound scaling only uses accuracy as the sole objective, it does not produce different scaling factors for TPUv3 and GPUv100. The base model EfficientNet-X-B0 is also included, which is the same for all cases.

D. Searching and Scaling on Server-class CPUs

Since CPUs still play an important role in the datacenters for machine learning [24] despite the accelerators being the new driving force, we also perform NAS on Xeon Platinum 8180 CPUs, a representative server-class machine in datacenters. As shown in Figure 2, the Xeon Platinum 8180 Skylake CPUs processor still provide a significant amount of computation (FLOPS as FLOPs/Sec), although TPUs and GPUs have more than 10X more FLOPS. Computation of modern CPUs is mostly from the AVX512/AVX2 instruction sets [4] running the special vector units, which makes it possible for CPUs to share similar FLOPs-latency nonpropotionality on CPUs behavior to TPUs and GPUs. On the other hand, when the AVX512/AVX2 is disabled, CPUs become scalar machines. A reasonable expectation for such scalar machines is that they will demonstrate good FLOPs-latency propotionality, very different from DC accelerators such as TPUs and GPUs.

Therefore, we perform search and model scaling on the CPUs with AVX512/AVX2 on and off to thoroughly study the implications of different hardware architectures. We use the same search space as described in Section 3 and Appendix A. Our results find different model families on CPUs with AVX512/AVX2 on and off. For both with and without AVX512/AVX2, the searched model architectures on the Xeon 8180 CPUs use ReLU as the only choice for activation functions. The main reason for this behavior is believed to be that CPUs are more sensitive for the increased FLOPs of swish (swish has 4X more FLOPs than ReLU), because of their much lower FLOPS than accelerators.

Concretely, with AVX512/AVX2 turned on, the searched base model architecture is the same as EfficientNet-X-B0 on GPUs as shown in Table 1. When AVX512/AVX2 is turned off, the searched based model architectures is the same as the original EfficientNet-B0 except ReLU instead of Swish used for all activation functions. The seemingly surprising results are actually easy to understand. With AVX512/AVX2 disabled, the CPUs are essentially scalar machines and exhibit very good FLOP-latency proportionality, preferring low FLOPs models to achieve faster speed. Since EfficientNet is searched using FLOPs as the performance objective, NAS converges on EfficientNet on CPUs with AVX512/AVX2 disabled. With AVX512/AVX2 enabled, the CPUs start to behave more like accelerators, demonstrating FLOP-latency nonproportionality similar to TPUs and GPUs and thus preferring models with higher parallelism and efficiency despite higher FLOPs. For scaling to form the model family, LACS works slightly better on the CPUs with AVX512/AVX2; while original EfficientNet compound scaling works slightly better on the CPUs without AVX512/AVX2 because the CPUs demonstrate strong FLOP-latency proportionality when turning off AVX512/AVX2.

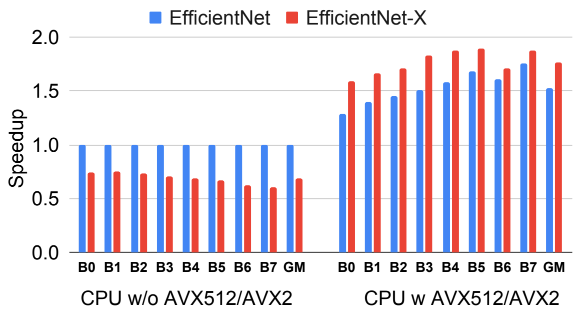

Figure 5: Speedup of EfficientNet-X over the baseline EfficientNet on CPUs. EfficientNet-X with ReLU (abbrev. EfficientNet-X) is the preferred model family obtained by search and scaling over Xeon 8180 CPU with AVX512/AVX2 enabled, while EfficientNet with ReLu (abbrev. EfficientNet) is the preferred model family obtained by search and scaling on Xeon 8180 CPU with AVX512/AVX2 disabled. Speedup is normalized again EfficientNet on Xeon 8180 CPU with AVX512/AVX2 disabled at individual model level respectively (thus the speedup of EfficientNet on Xeon 8180 CPU with AVX512/AVX2 disabled is always one). Models at the same level achieves similar accuracy. GM is geometric mean.

Figure 5 shows the detailed model inference speed from B0 to B7 level, with models at the same level achieving similar accuracy as shown in Table 3. EfficientNet-X with ReLU (abbrev. EfficientNet-X) is the searched model family on Xeon 8180 CPU with AVX512/AVX2 enabled, while EfficientNet with ReLu (abbrev. EfficientNet) is the searched model family on Xeon 8180 CPU with AVX512/AVX2 disabled. When AVX512/AVX2 is disabled, the Xeon 8180 CPU demonstrated strong FLOP-latency proportionality, with EfficientNet achieving 45% speedup on average (geometric mean) with about half FLOPs compared to EfficientNet-X. One the other hand, when AVX512/AVX2 is enabled, the Xeon 8180 CPU demonstrated obvious FLOPs-latency nonproportionality, with EfficientNet-X achieving 16% speedup average despite its 2X FLOPs compared to EfficientNet. Additionally, when enabling AVX512/AVX2 on the Xeon 8180 CPUs, EfficientNet-X and EfficientNet achieve 1.8X and 1.5X speedup, respectively.

These results, together with earlier results on TPUs and GPUs, indicate that the more advanced vector/matrix units a platform has, the stronger FLOPs-latency nonproportionality is for the platform. When the platform is a scalar machine, it has clear FLOPs-latency proportionality. Thus, a general rule of thumb for selecting performance objective can be derived as: Latency is a good performance objective for accelerators with strong matrix/vector unit, while FLOPs can still be useful as performance objective approximation for scalar platforms such as some mobile processors without accelerator units.