Table 2: Microarchitecture search space. DWS: Depthwise Separable.

block name	type	kernel	dilation	nonlinearity
`conv2d_3x3`	Convolution	3	1	ReLU
`conv2d_5x5`	Convolution	5	1	ReLU
`dws_3x3`	DWS Conv.	3	1	ReLU
`dws_5x5`	DWS Conv.	5	1	ReLU
`dil_3x3`	Convolution	3	2	ReLU
`dil_5x5`	Convolution	5	2	ReLU
`identity`	-	-	-	-
`zero`	-	-	-	-

A Micro-architecture search

Table 2 presents the candidate operations in a cell. We include standard, dilated and depthwise separable (DWS) convolutions along with the identity and zero operations. For simplicity, we only consider ReLU activations.

B Utilization and Runtime details

In this section, we analyze the utilization and runtime of all the building blocks. We consider the operations of Table 2 as well as fully connected layers (for the classifier). Maxpooling layers, batch normalization and activation functions, i.e., ReLUs, are characterized by full utilization and zero runtime, since they need no matrix multiplications.

Let k_1 and k_2 be the kernel sizes, c and f the input and output channels, s_1 and s_2 the systolic array dimensions, h and w the height and width of the input, b the batch size. The number of operations is

$$\texttt{MACs} = hwbk_1k_2cf \tag{9}$$

The utilization of a specific layer is computed by dividing the number of MACs by the runtime.

Convolution The runtime and utilization of a convolution are computed in Section 3.2 of the main text:

$$\texttt{RUNTIME}_{\text{conv}} = \left\lceil \frac{k_1k_2c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil hwb \tag{10}$$

$$\texttt{UTIL}_{\text{conv}} = \frac{k_1k_2cf}{s_1s_2 \left\lceil \frac{k_1k_2c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil} \tag{11}$$

Table 3: Utilizations and runtimes for all building blocks. Symbols explained in text. † includes all other layer types: identity, zero, maxpooling, ReLUs.

Block Type	Runtime	Utilization
Convolution	$\left\lceil \frac{k_1 k_2 c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil hwb$	$\frac{k_1 k_2 cf}{s_1 s_2 \left\lceil \frac{k_1 k_2 c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil}$
Depthwise Convolution	$c \left\lceil \frac{k_1 k_2}{s_1} \right\rceil fhwb$	$\frac{k_1 k_2}{\left\lceil \frac{k_1 k_2}{s_1} \right\rceil f}$
Fully connected	$\left\lceil \frac{c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil b$	$\frac{cf}{s_1 s_2 \left\lceil \frac{c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil}$
†	0	1

Depthwise Convolution A single convolutional filter is applied to each input channel. In this case the number of input and output channels is the same $c = f$. There is no input reuse, meaning that only one column of the systolic array is used. In other words, the $\left\lceil \frac{f}{s_2} \right\rceil$ term in Eq. 10 is replaced by $\left\lceil \frac{1}{s_2} \right\rceil = 1$. Finally, the operation is repeated c times, yielding the following runtime:

$$\texttt{RUNTIME}_{\text{depthwise}} = c \left\lceil \frac{k_1 k_2}{s_1} \right\rceil hwb \tag{12}$$

$$\texttt{UTIL}_{\text{depthwise}} = \frac{k_1 k_2}{s_1 s_2 \left\lceil \frac{k_1 k_2}{s_1} \right\rceil} \tag{13}$$

The utilization is calculated by dividing the number of multiply-accumulates (MACs) by the runtime. Eq. 13 shows the ineffectiveness of the depthwise convolution, which is inversely proportional to the second dimension of the systolic array.

Depthwise Separable (DWS) Convolution The depthwise separable convolution is the sequence of a depthwise convolution and a (standard) convolution. Thus, the runtime and utilization are computed via addition of the respective terms.

Fully Connected layers The runtime and utilization can be derived from the convolution formulae by setting $k_1 = k_2 = 1$ and $h = w = 1$. Concretely, the kernel size can be considered to be 1×1, while the fully connected layer has c inputs and f outputs.

Table 4: Experimental results for `CIFAR10` over 3 random seeds.

	Accuracy (%, ↑)				Runtime (μs, ↓)				HV (↓)
λ	0.1	0.5	1.0	5.0	0.1	0.5	1.0	5.0	
Blackbox	91.4±1.07	90.2±0.25	91.3±0.66	90.4±0.83	209±57	155±9	147±14	122±2	1.47
Roofline	91.7±0.68	89.2±0.85	88.7±0.91	87.6±4.58	214±43	175±33	137±62	252±53	1.86
FLOPS	90.0±0.88	88.4±1.91	84.0±6.39	87.0±0.99	235±26	320±37	251±55	159±33	2.68
U-Boost	90.9±0.88	91.4±0.90	91.3±0.24	89.5±1.14	73±8	51±10	39±9	30±0	0.386

$$\texttt{RUNTIME}_{\text{fc}} = \left\lceil \frac{c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil b \tag{14}$$

$$\texttt{UTIL}_{\text{fc}} = \frac{cf}{s_1 s_2 \left\lceil \frac{c}{s_1} \right\rceil \left\lceil \frac{f}{s_2} \right\rceil} \tag{15}$$

C Additional experimental results

In this Section, we present additional experiments on `CIFAR10` and `ImageNet100` datasets.

C.1 `CIFAR10` dataset

Fig. 8 shows the cells found during the micro-architecture search stage for *all* methods. The methods opt for different configurations. Specifically, the FLOPS model selects mainly depthwise separable convolutions, since they correspond to fewer operations. However, such convolutions result in very increased runtimes and severe mitigation in utilization, as Eq. 12 and Eq. 13 show. The Roofline model operates on the compute-bound region and behaves identically as the FLOPS model. The Blackbox model tries to compensate (in terms of utilization) by omitting convolutions, including depthwise separable convolutions. This suggests that it is able to understand that DWS are antithetical to the utilization objective and opts for operations with no utilization overhead, such as the identity and zero gates.

Table 4 presents the experimental results for `CIFAR10` in more detail. The proposed method achieves significantly lower runtimes for all λ values outperforming the baselines in a range of $\sim 2.8 - 5\times$. It is also worth mentioning that the FLOPS and Roofline models do not exhibit decreasing runtimes as λ increases. They are also characterized by high variance in the runtime measurements, indicating an unsophisticated search. This drawback can be attributed to the loss function for the utilization term which does not take into account the number of channels. The blackbox model and our proposed method have lower standard deviations and a monotonically decreasing runtime. Finally, our proposed method has better quality of exploration for the tradeoff of accuracy and runtime, as the Hypervolume metric indicates.

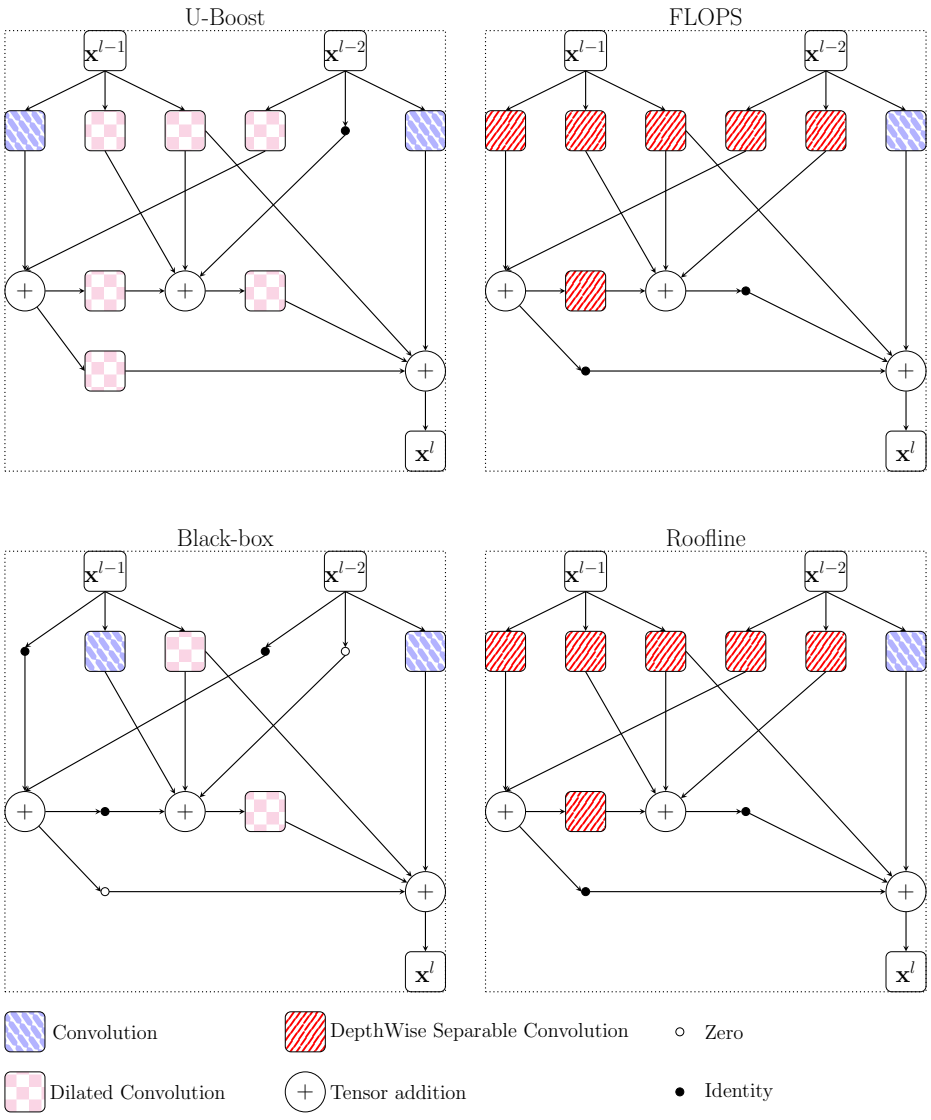

Fig. 8: Cell architectures found for $\lambda = 0.1$ on the `CIFAR10` dataset.

C.2 `ImageNet100` dataset

Table 5 presents additional experimental results on `ImageNet100`. The FLOPS and Roofline baselines exhibit significant drops in performance as more emphasis is placed on runtime. U-Boost outperforms the other methods in terms of runtime by a notable margin of $\sim 2.1 - 3.8\times$.

Table 5: imagenet100

	Accuracy (%, ↑)			Runtime (ms, ↓)			HV (↓)
	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 5.0$	$\lambda = 0.1$	$\lambda = 1.0$	$\lambda = 5.0$	(across λ)
Blackbox	87.5	87.8	87.9	4.8	4.05	3.8	45.98
Roofline	86.5	84.0	74.2	4.7	3.5	2.9	100.62
FLOPS	87.2	78.4	80.2	6.1	3.45	3.42	102.02
U-Boost	87.8	87.9	86.3	2.2	1.05	0.77	13.94

D Hyperparameters

The complete list of hyperparameters is presented in Table 6.

Table 6: Experiment Hyperparameters. – indicates that the `ImageNet100` experiment uses the same settings as the `CIFAR10` experiment. †: the architecture for `ImageNet100` is produced by search on `CIFAR10`. MS: micro-architecture search, CS: channel search, FT: final training.

	`CIFAR10`	`ImageNet100`
`ms_no_epoch`	10	†
`cs_no_epoch`	30	†
`ft_no_epoch`	100	70
`array_size`	[128, 128]	–
`start_arch_train`	0	–
`weight_vs_arch`	0.8	–
`search_sgd_init_lr`	0.05	–
`search_sgd_momentum`	0.9	–
`search_sgd_weight_decay`	3e-4	–
`search_weight_grad_clip`	0.5	–
`adam_init_lr`	0.1	–
`adam_weight_decay`	0	–
`init_tau`	1.0	–
`tau_anneal_rate`	0.95	–
`min_tau`	0.001	–
`search_batch_size`	64	–
`train_batch_size`	256	–
`train_sgd_init_lr`	0.1	–
`train_sgd_momentum`	0.9	–
`train_sgd_weight_decay`	5e-4	–
`train_weight_grad_clip`	0.5	–