Generalized Depthwise-Separable Convolutions for Adversarially Robust and Efficient Neural Networks

Hassan Dbouk & Naresh Shanbhag

Department of Electrical and Computer Engineering

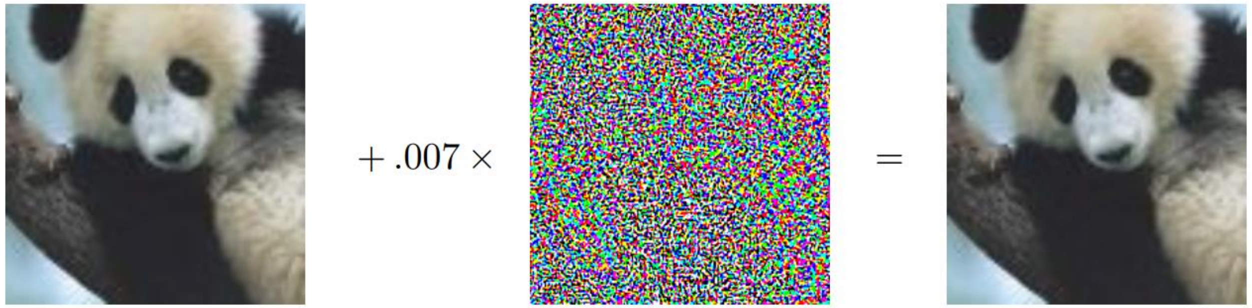
University of Illinois at Urbana-Champaign

Motivation: Robust and Efficient Inference

deep nets are vulnerable

original sample

adversarial sample



decision: ‘panda’

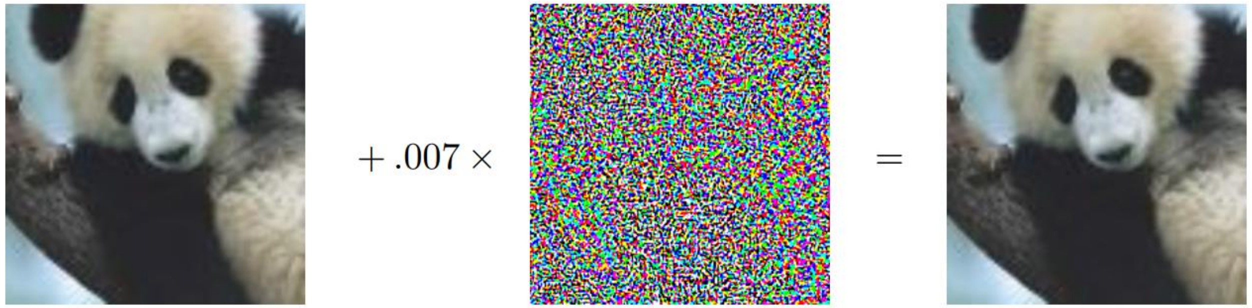
decision: ‘gibbon’

Motivation: Robust and Efficient Inference

deep nets are vulnerable

original sample

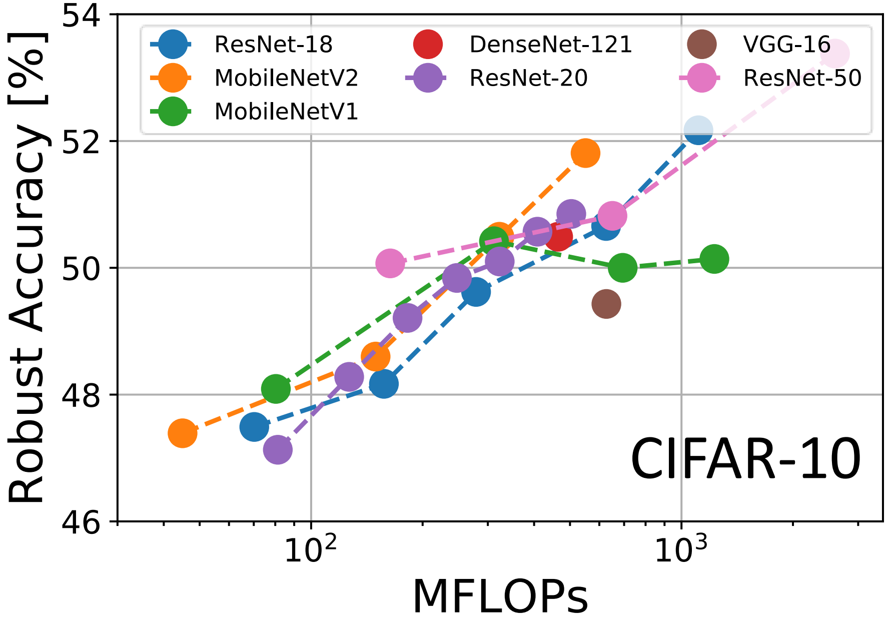
adversarial sample



decision: 'panda'

decision: 'gibbon'

deep nets are expensive

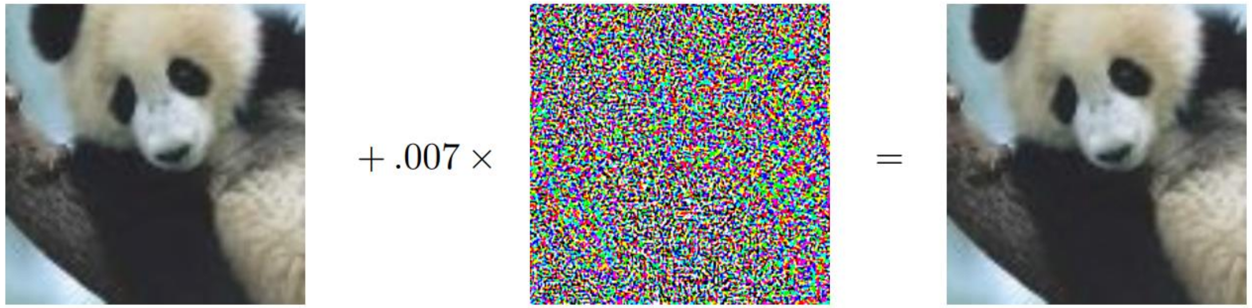


Motivation: Robust and Efficient Inference

deep nets are vulnerable

original sample

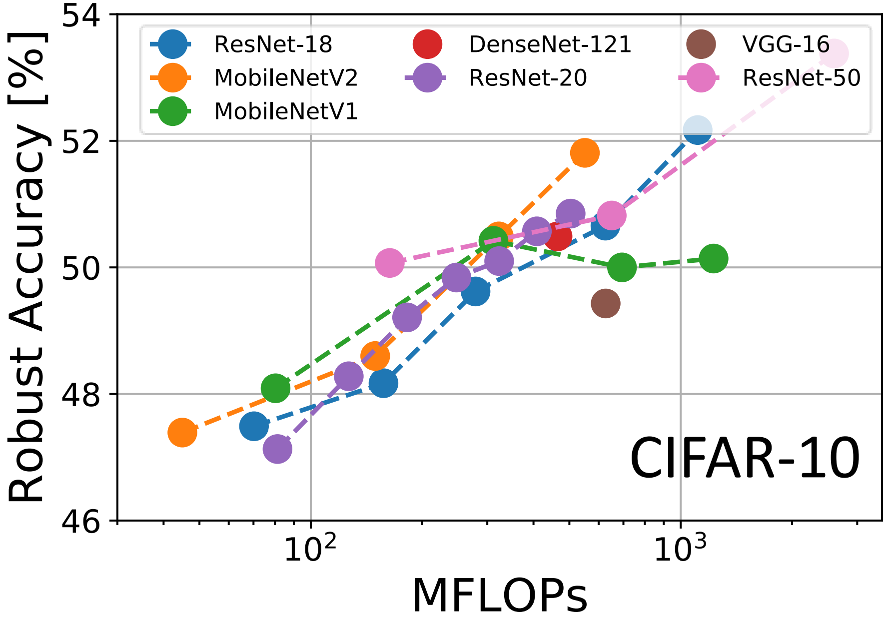
adversarial sample



decision: ‘panda’

decision: ‘gibbon’

deep nets are expensive



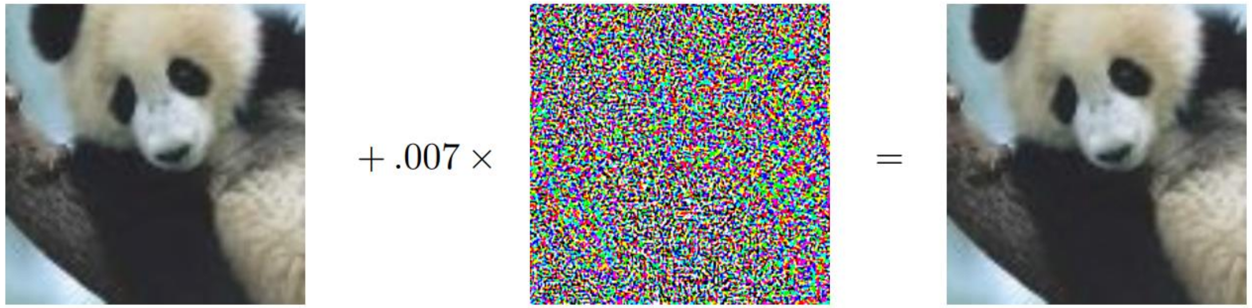
design **robust** and **accurate** deep nets that achieve **high FPS** when mapped onto edge hardware

Motivation: Robust and Efficient Inference

deep nets are vulnerable

original sample

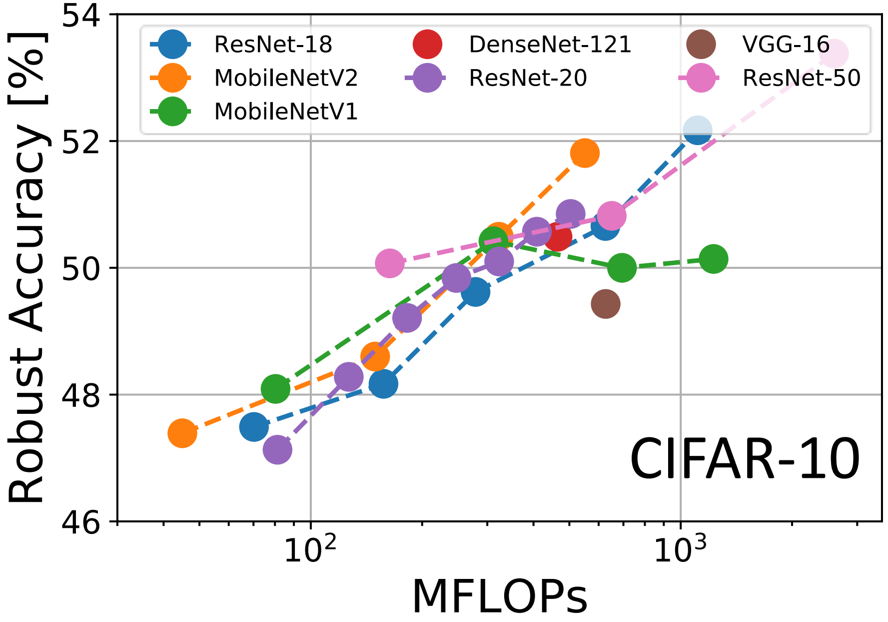
adversarial sample



decision: ‘panda’

decision: ‘gibbon’

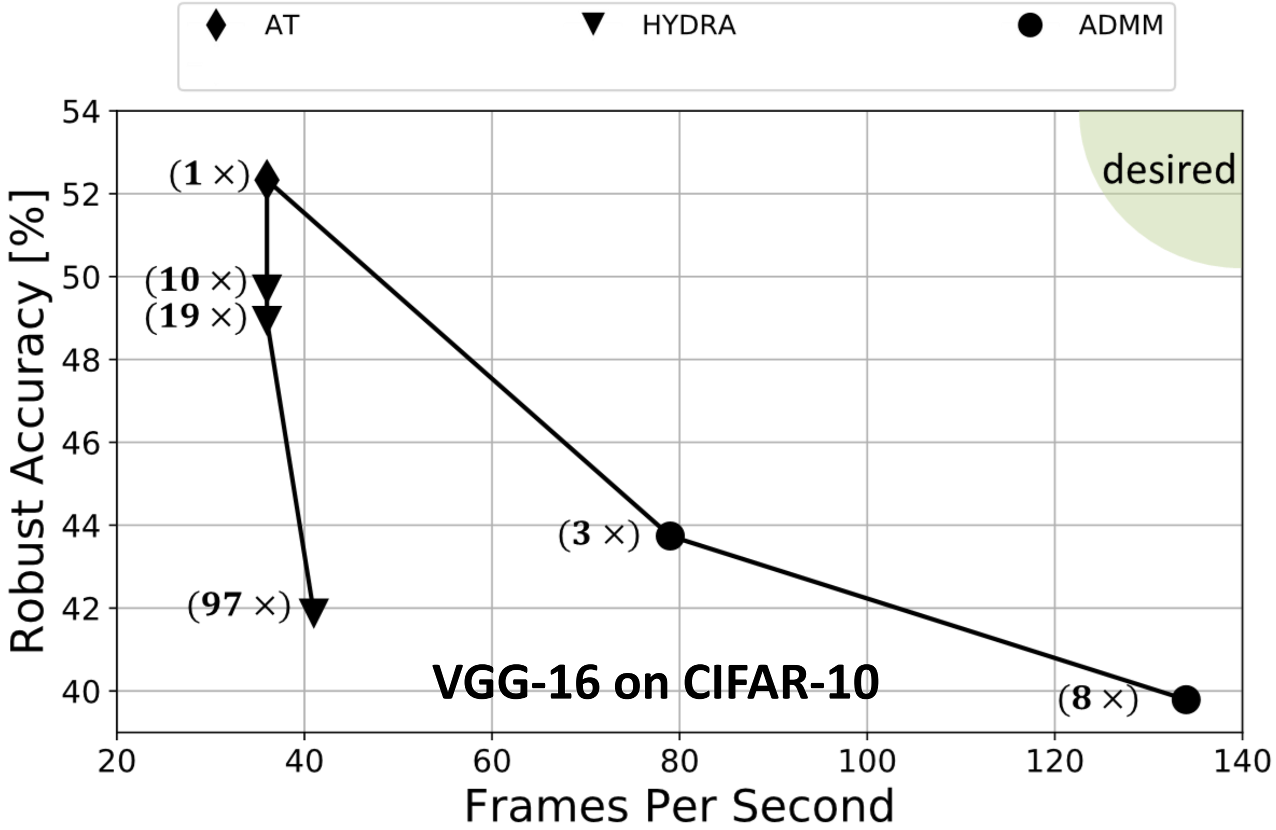
deep nets are expensive



design **robust** and **accurate** deep nets that achieve **high FPS** when mapped onto edge hardware

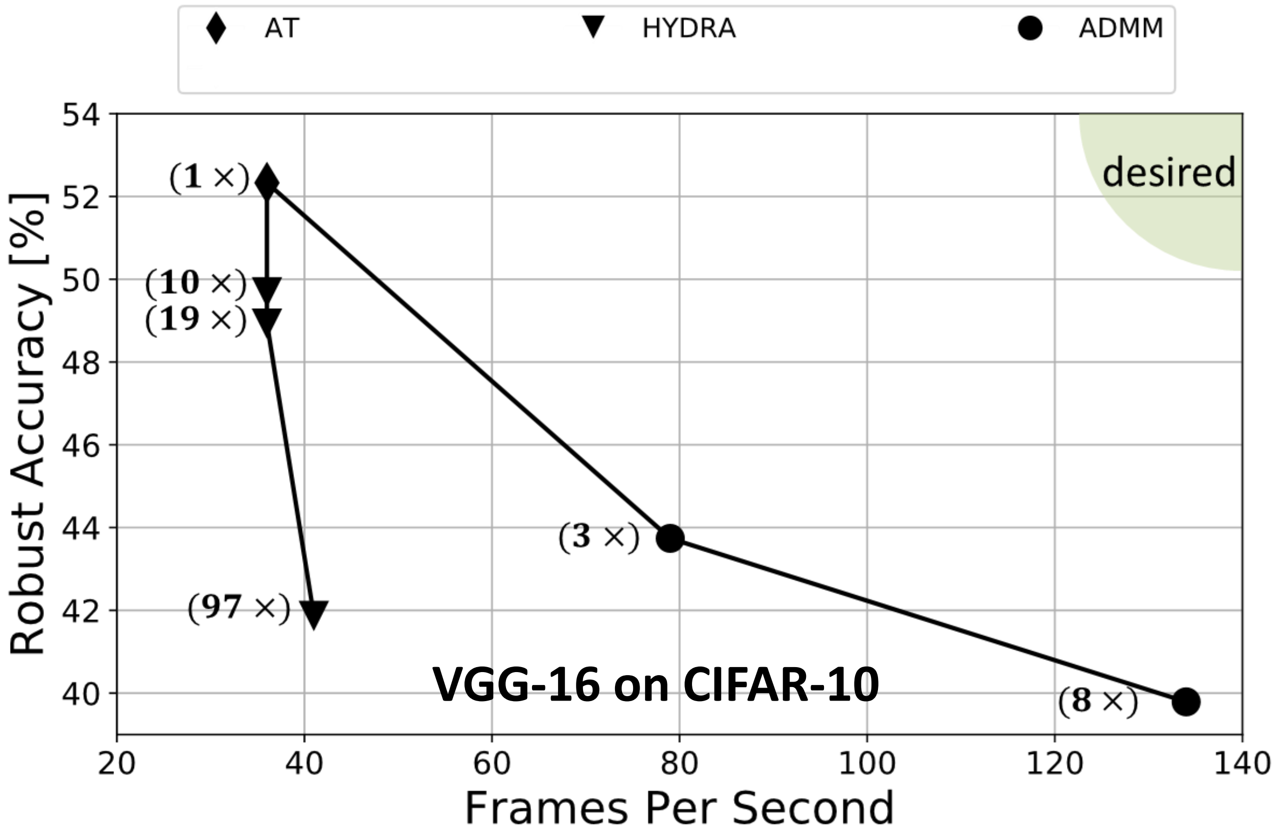
NVIDIA Jetson Xavier

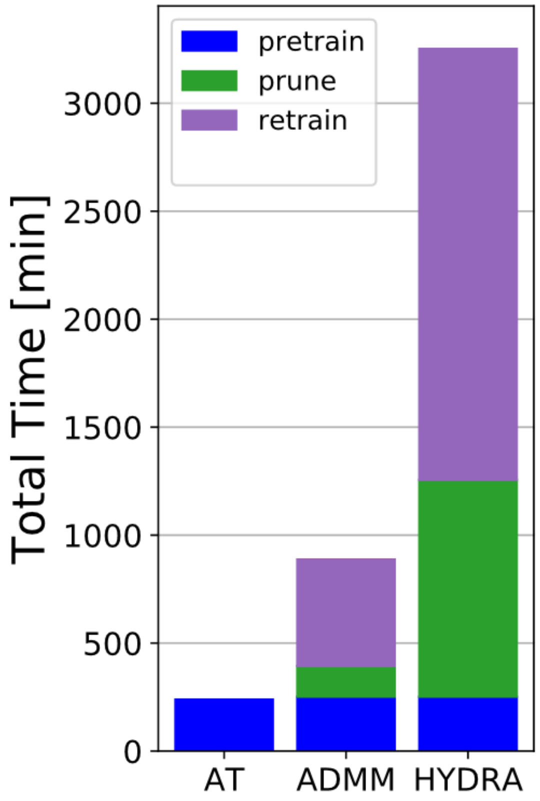
Limitations of Existing Techniques



- reductions often **<u>don’t</u>** translate to hardware

Limitations of Existing Techniques

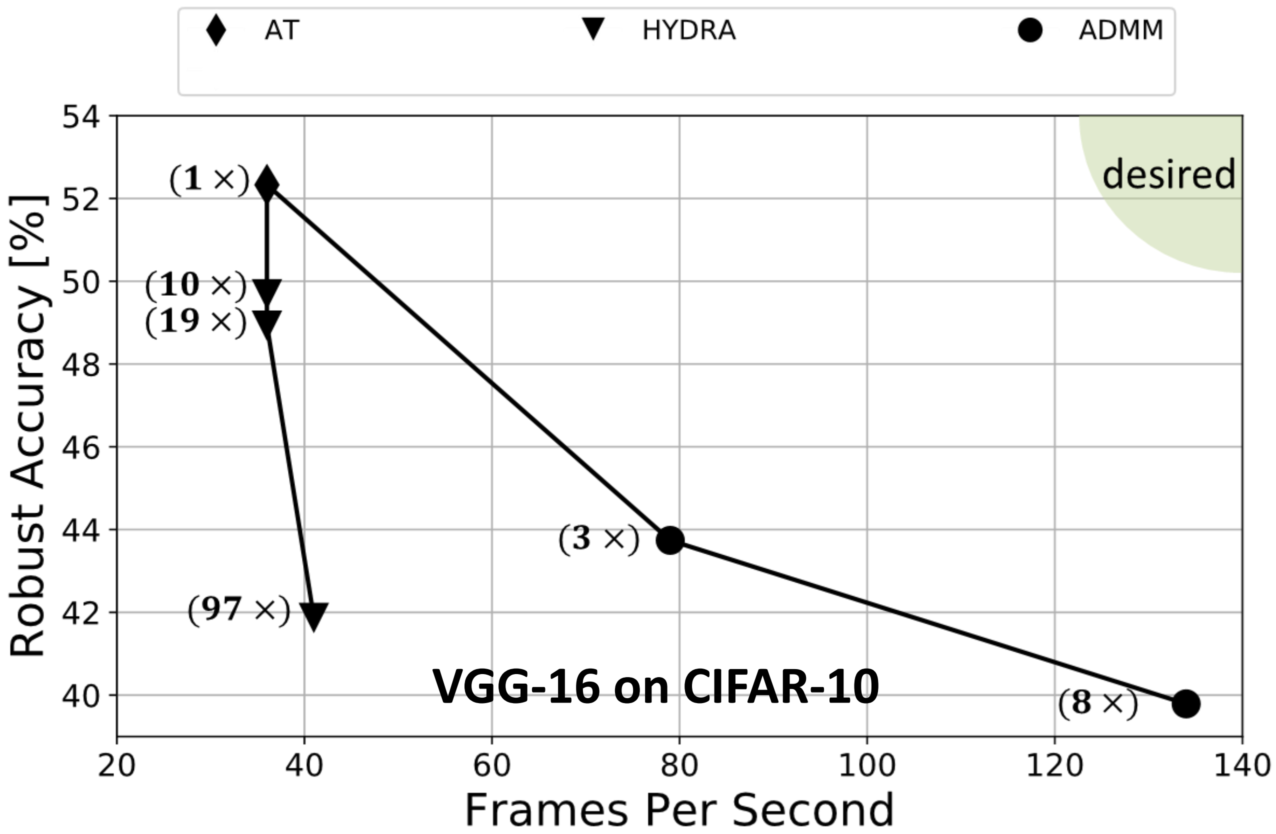


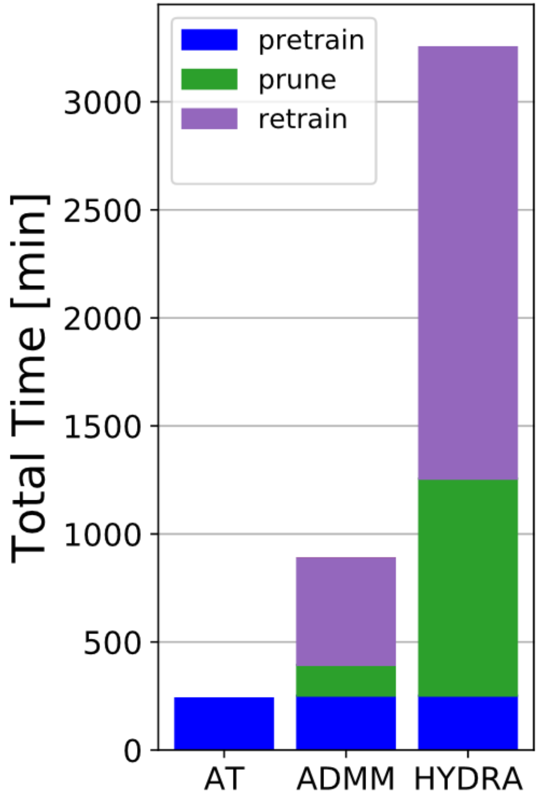


FPS measured on NVIDIA Jetson
time measured on NVIDIA 1080 Ti

- reductions often **<u>don’t</u>** translate to hardware
- make AT **<u>more expensive</u>**

Limitations of Existing Techniques

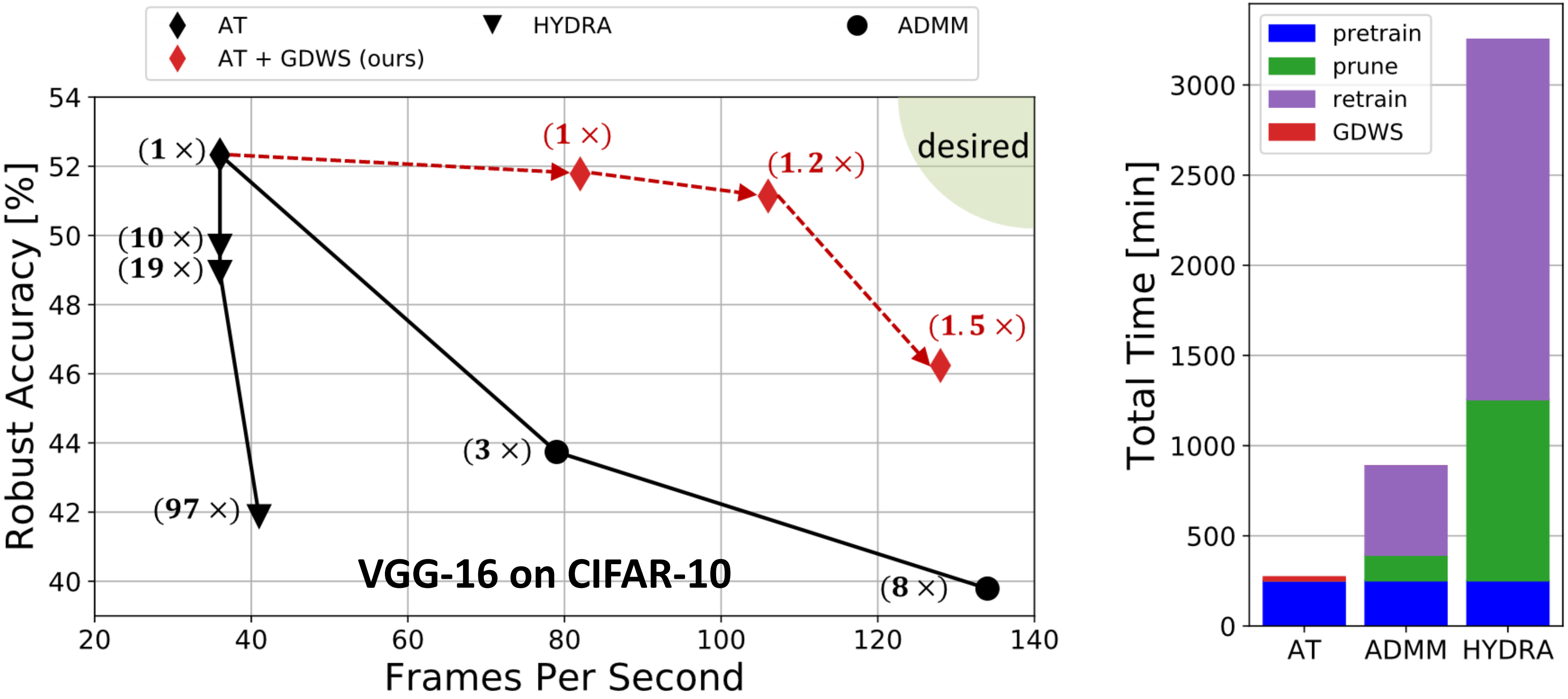




FPS measured on NVIDIA Jetson
time measured on NVIDIA 1080 Ti

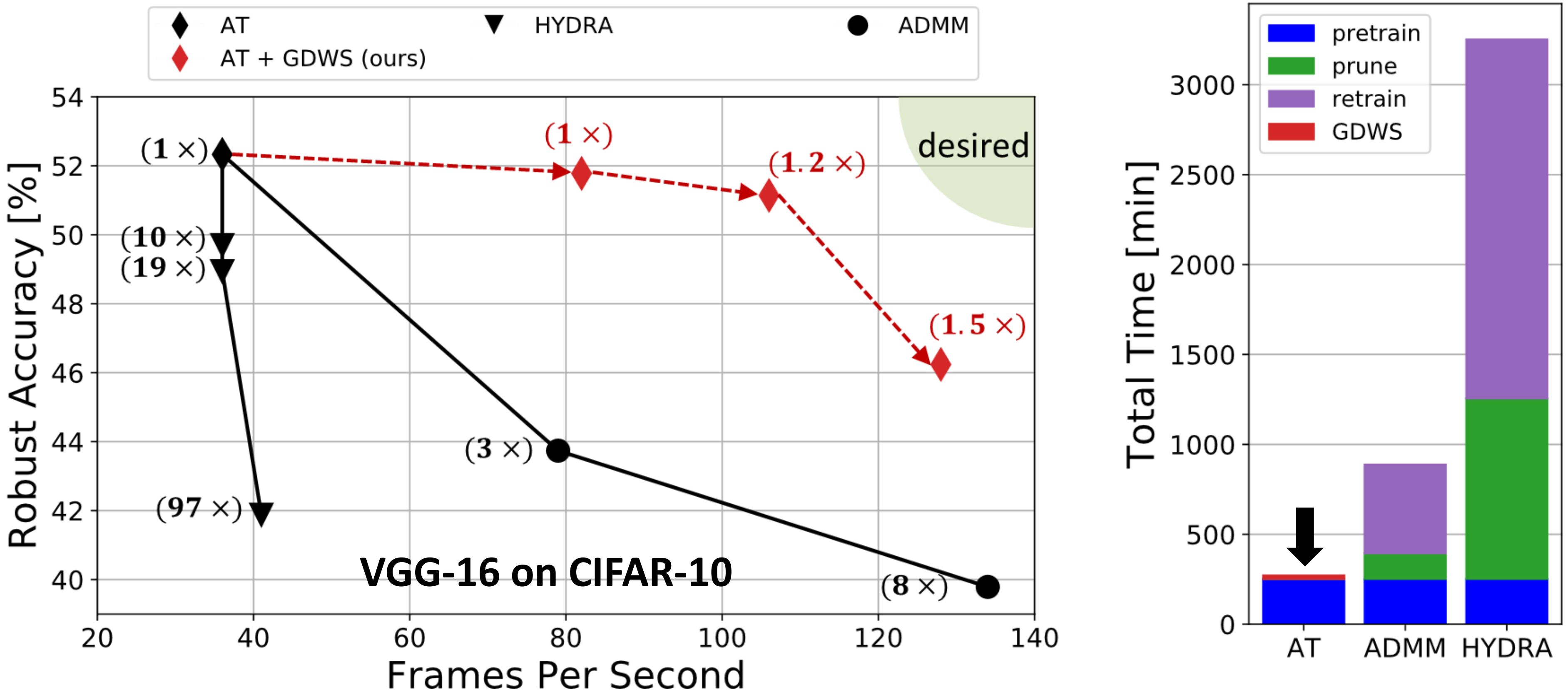
- reductions often **<u>don't</u>** translate to hardware
- make AT **<u>more expensive</u>**
- ad hoc in nature, **<u>no theoretical</u>** basis behind them

Propose: Generalized Depth-wise Separable Convolutions



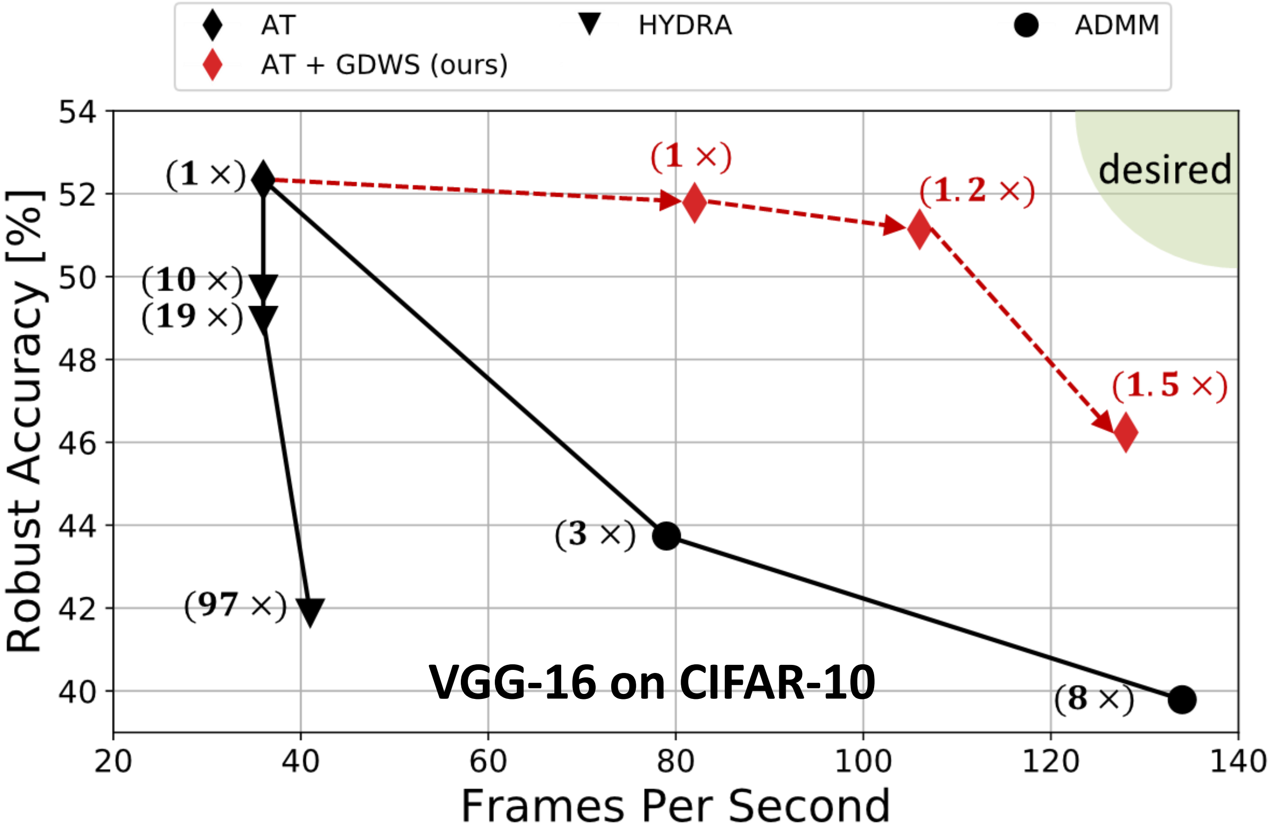
- dramatically improve **FPS** while preserving **robust accuracy**

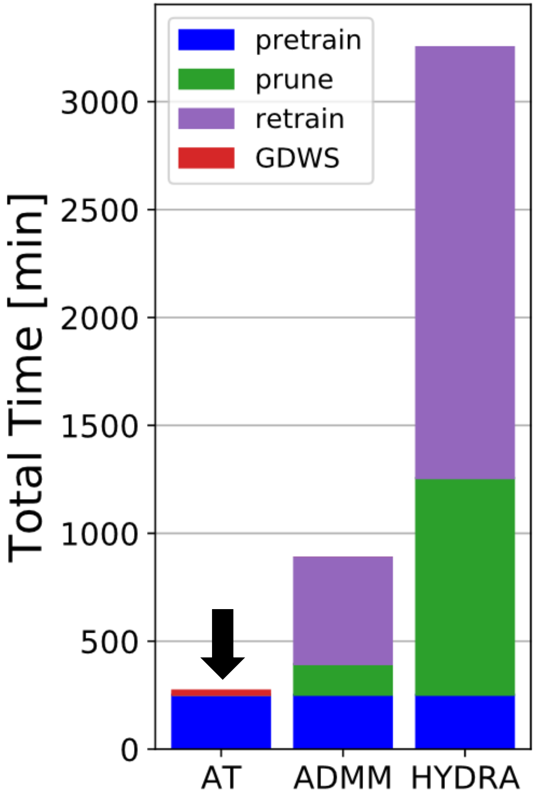
Propose: Generalized Depth-wise Separable Convolutions



- dramatically improve **FPS** while preserving **robust accuracy**
- operate on pre-trained models → *no additional training*

Propose: Generalized Depth-wise Separable Convolutions

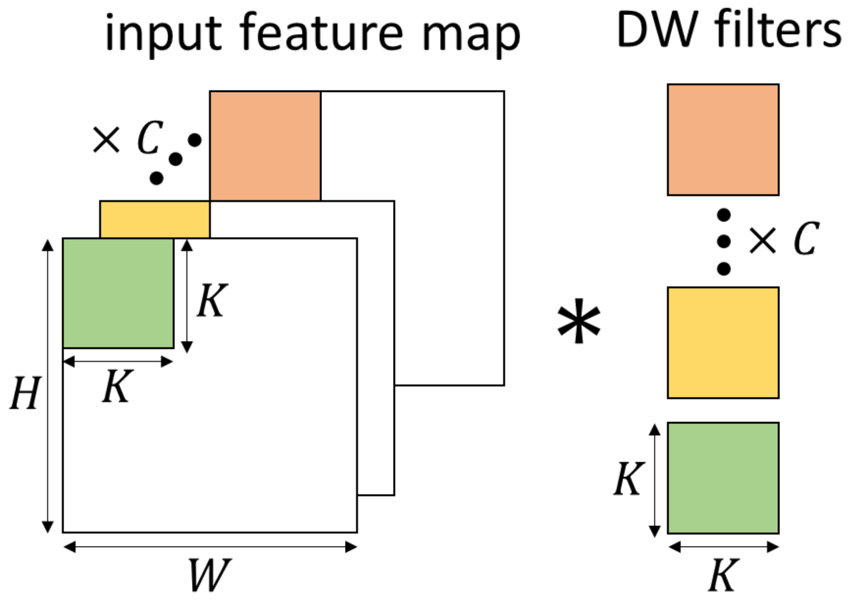




- dramatically improve **FPS** while preserving **robust accuracy**
- operate on pre-trained models → *no additional training*
- optimal and efficient approximation algorithms developed

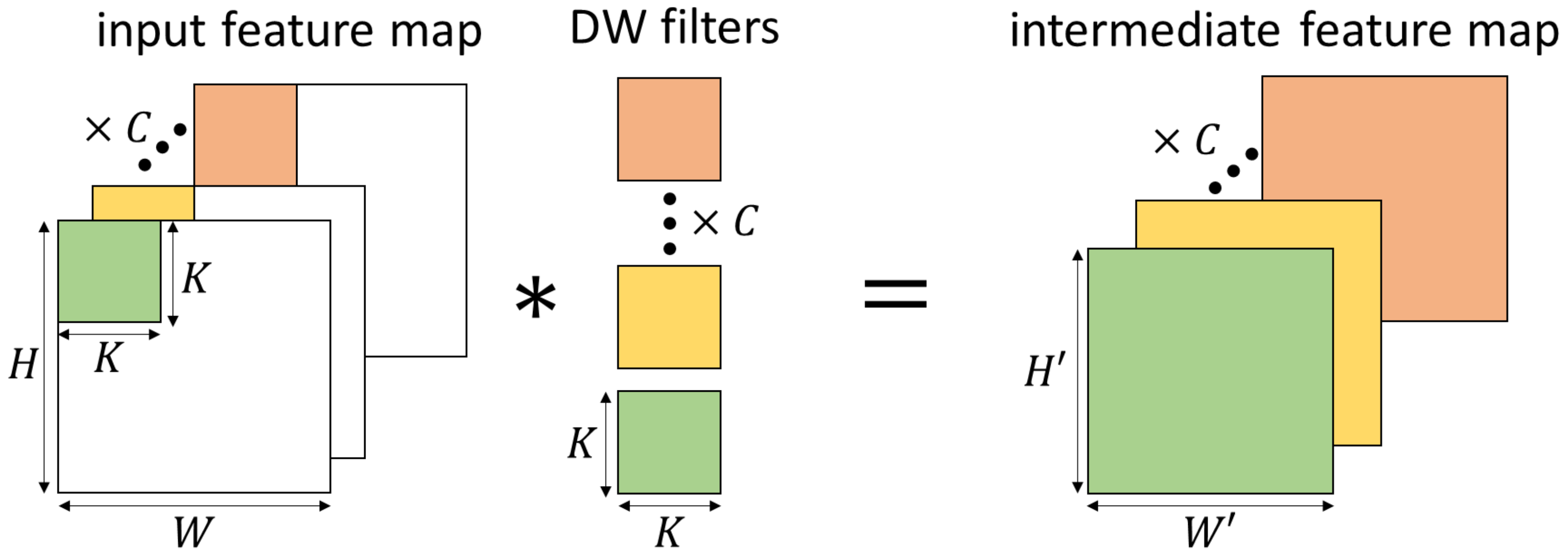
Depth-wise Separable Convolutions

- popularized by MobileNets [arXiv’17, CVPR’18]



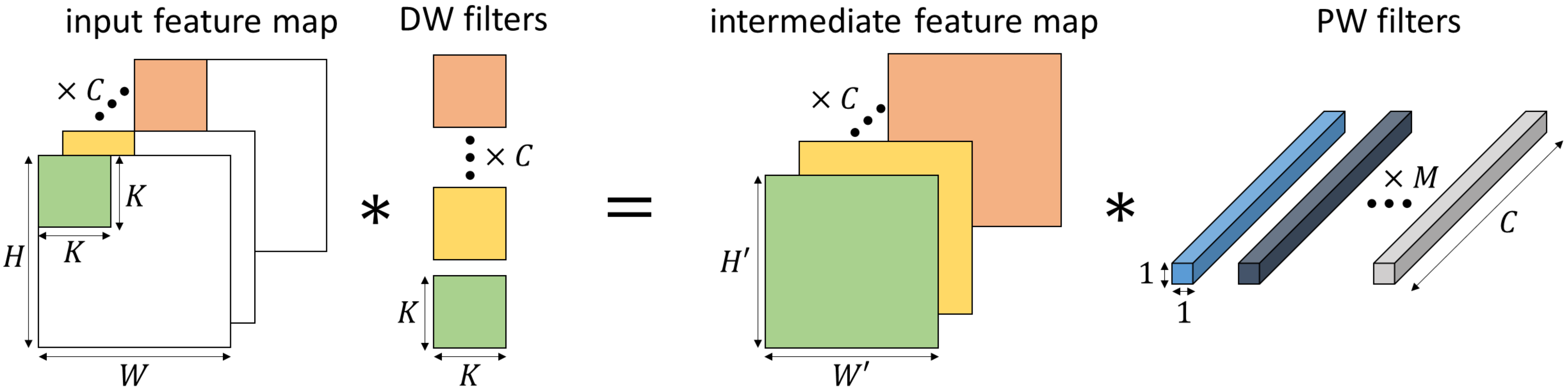
Depth-wise Separable Convolutions

- popularized by MobileNets [arXiv’17, CVPR’18]



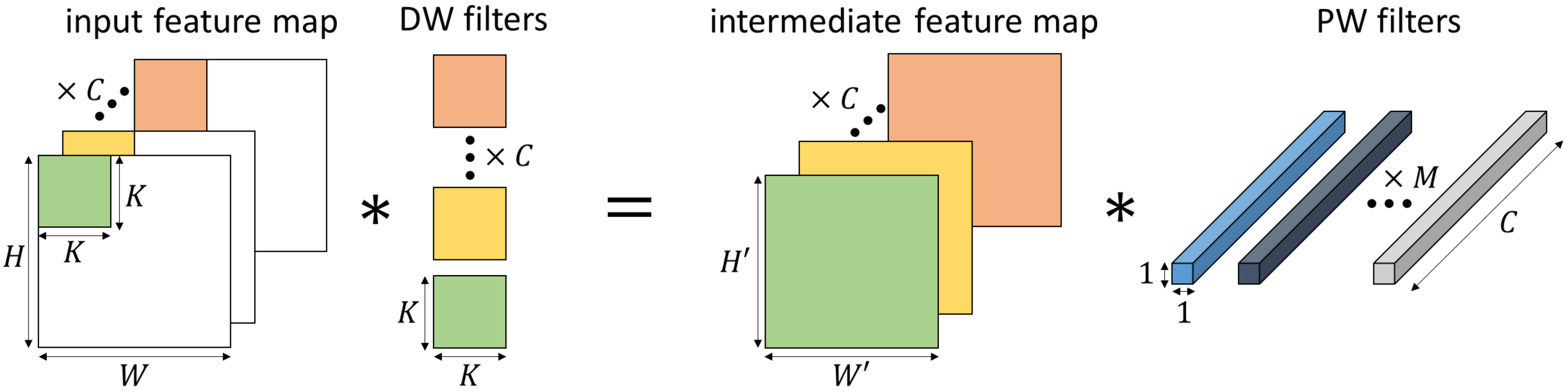
Depth-wise Separable Convolutions

- popularized by MobileNets [arXiv'17, CVPR'18]



Depth-wise Separable Convolutions

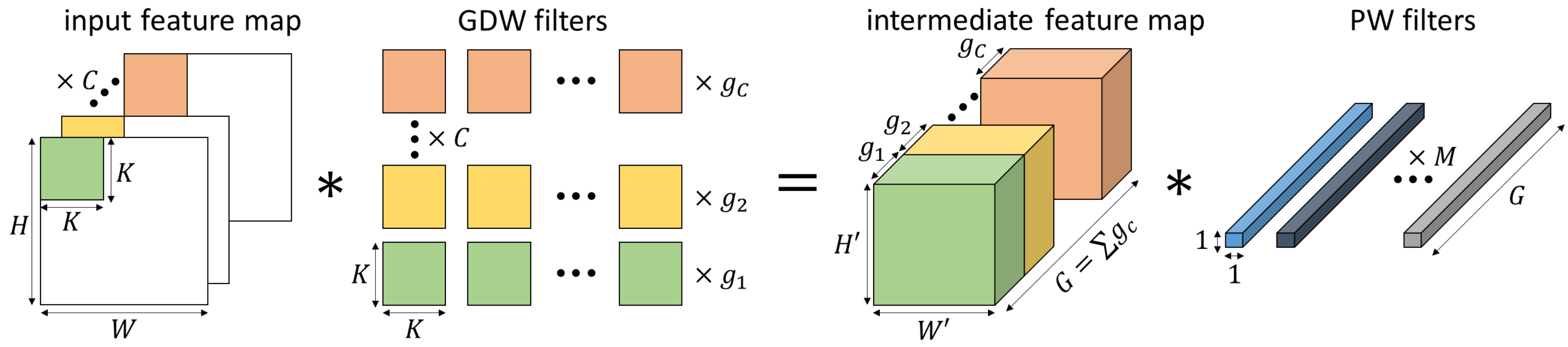
- popularized by MobileNets [arXiv’17, CVPR’18]



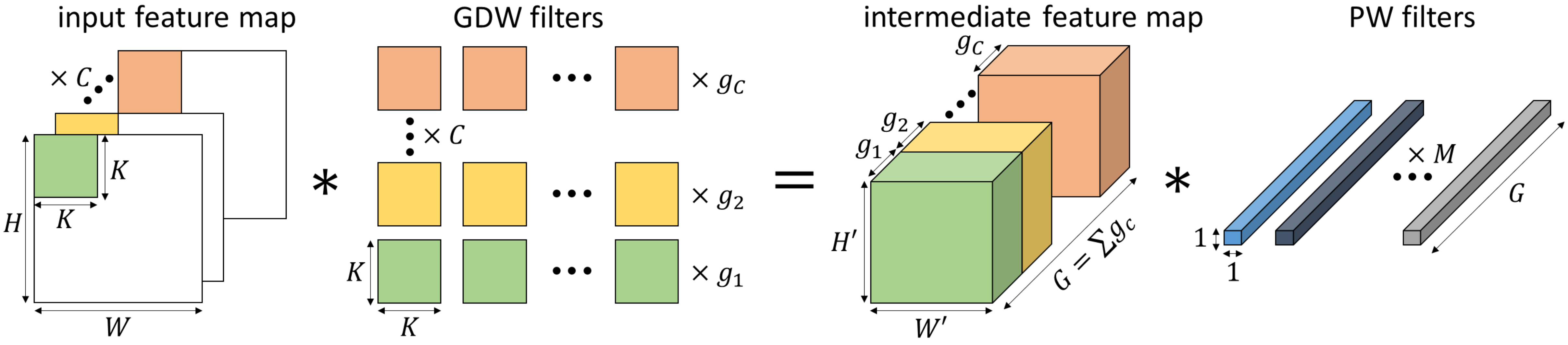
- number of FLOPs required per forward pass:

$$H'W'C(K^2 + M)$$

Generalized Depth-wise Separable Convolutions



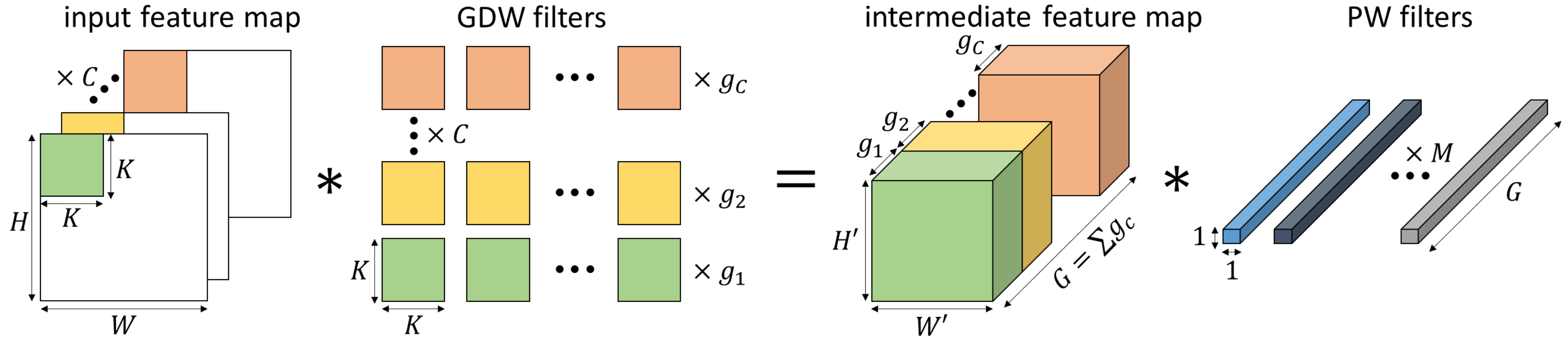
Generalized Depth-wise Separable Convolutions



- number of FLOPs required per forward pass:

$$H'W'\left(\sum_{c=1}^{C} g_c(K^2+M)\right) = H'W'G(K^2+M) = \gamma(\mathbf{g})$$

Generalized Depth-wise Separable Convolutions

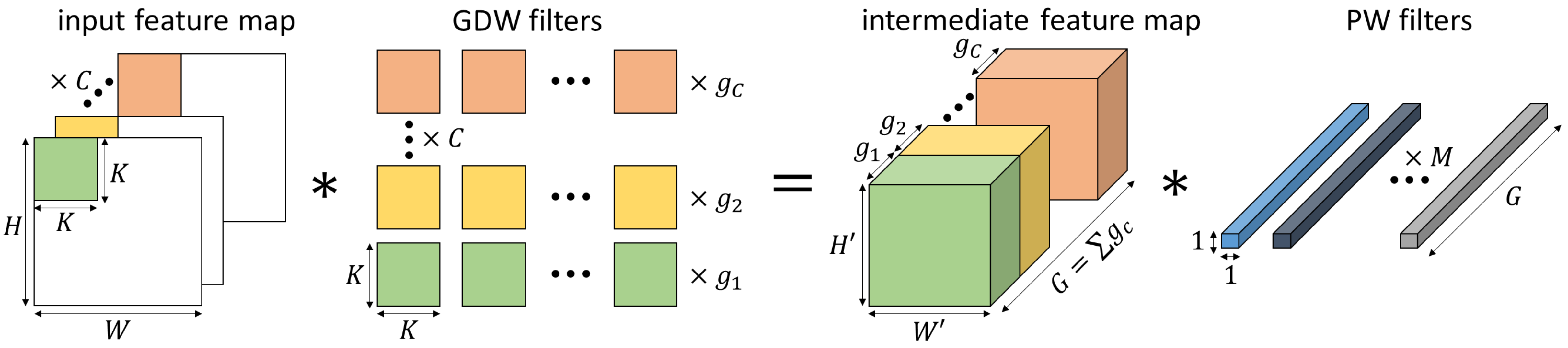


- number of FLOPs required per forward pass:

$$H'W'\left(\sum_{c=1}^{C} g_c(K^2 + M)\right) = H'W'G(K^2 + M) = \gamma(\mathbf{g})$$

- how to choose the g_c's?

Generalized Depth-wise Separable Convolutions

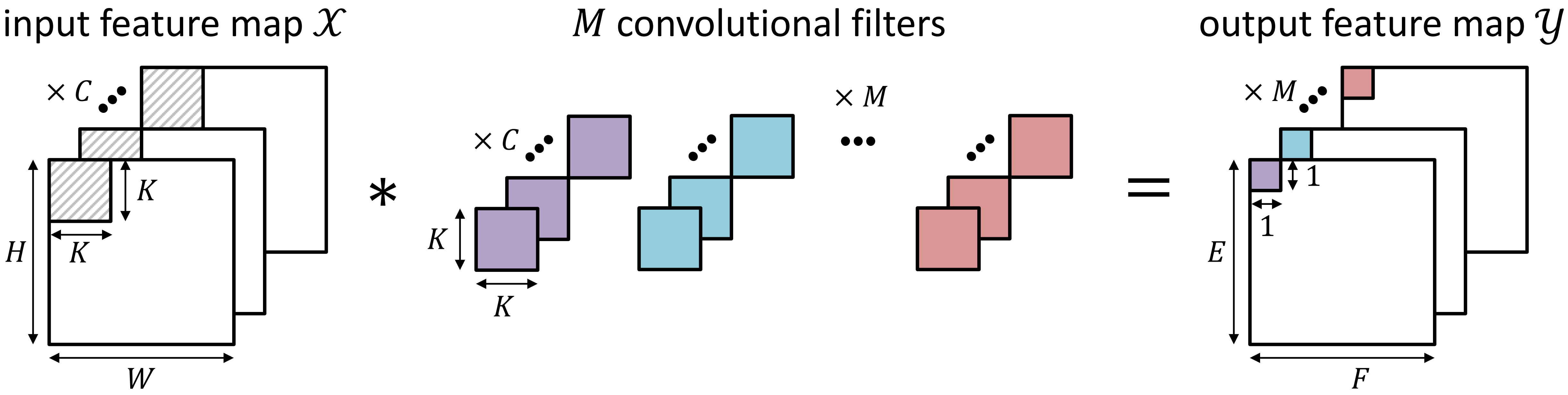


- number of FLOPs required per forward pass:

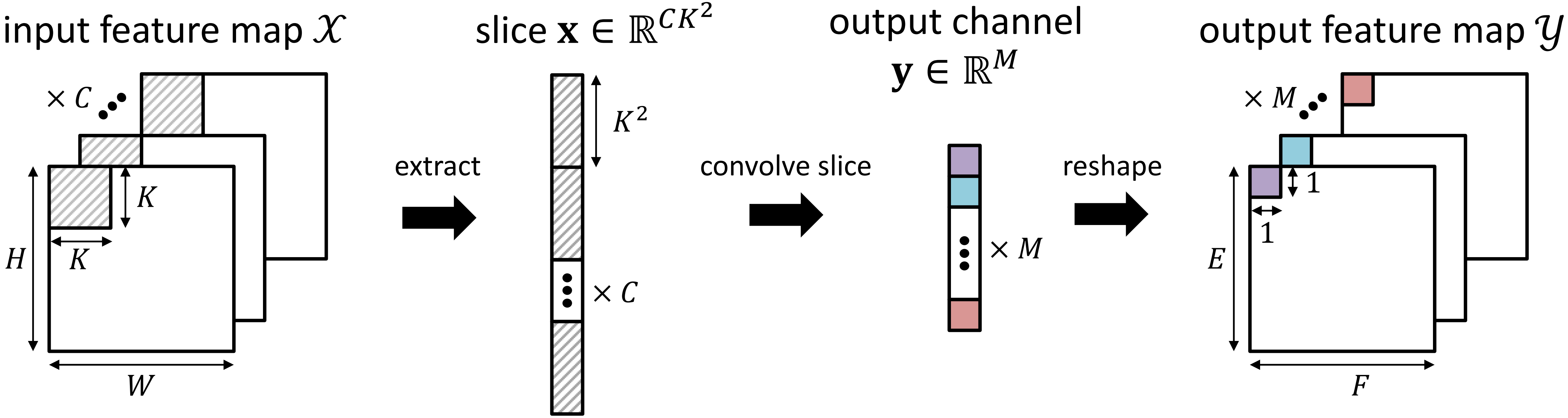
$$H'W'\left(\sum_{c=1}^{C} g_c(K^2+M)\right) = H'W'G(K^2+M) = \gamma(\mathbf{g})$$

- how to choose the g_c's? → optimal approximation algorithm

Standard 2D Convolution as a Matrix Multiplication



Standard 2D Convolution as a Matrix Multiplication



input feature map $\mathcal{X}$

extract

slice $\mathbf{x} \in \mathbb{R}^{CK^2}$

convolve slice

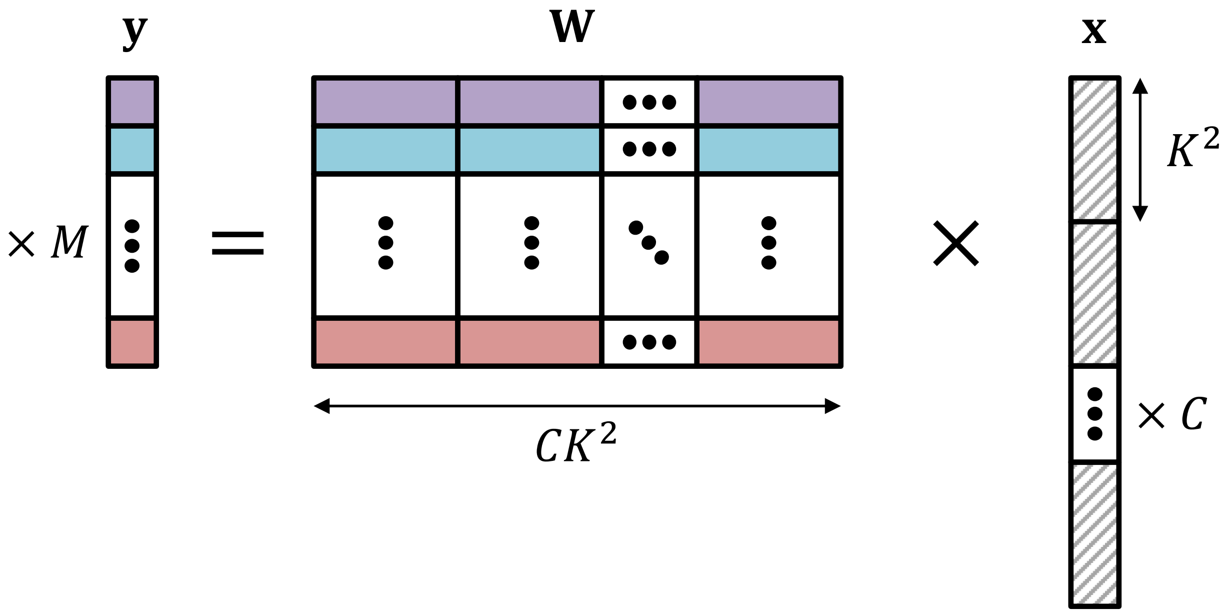
output channel $\mathbf{y} \in \mathbb{R}^M$

reshape

output feature map $\mathcal{Y}$

- vectorizing inputs and outputs

Standard 2D Convolution as a Matrix Multiplication

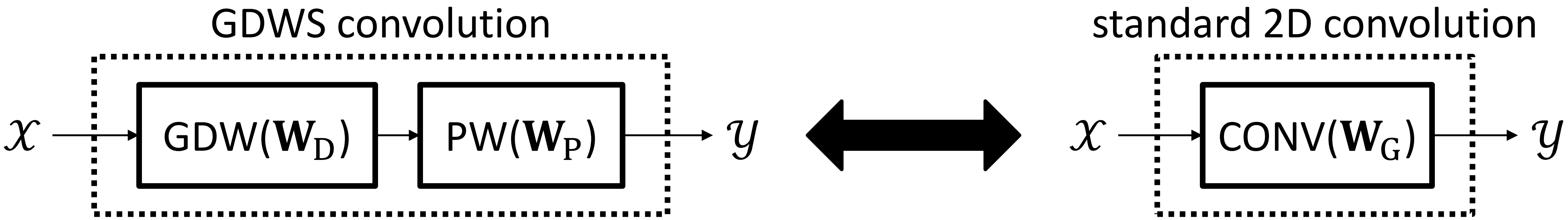


- matrix vector multiplication for one output channel vector
- complete convolution via matrix multiplication

Property 1: Equivalent Standard Convolution

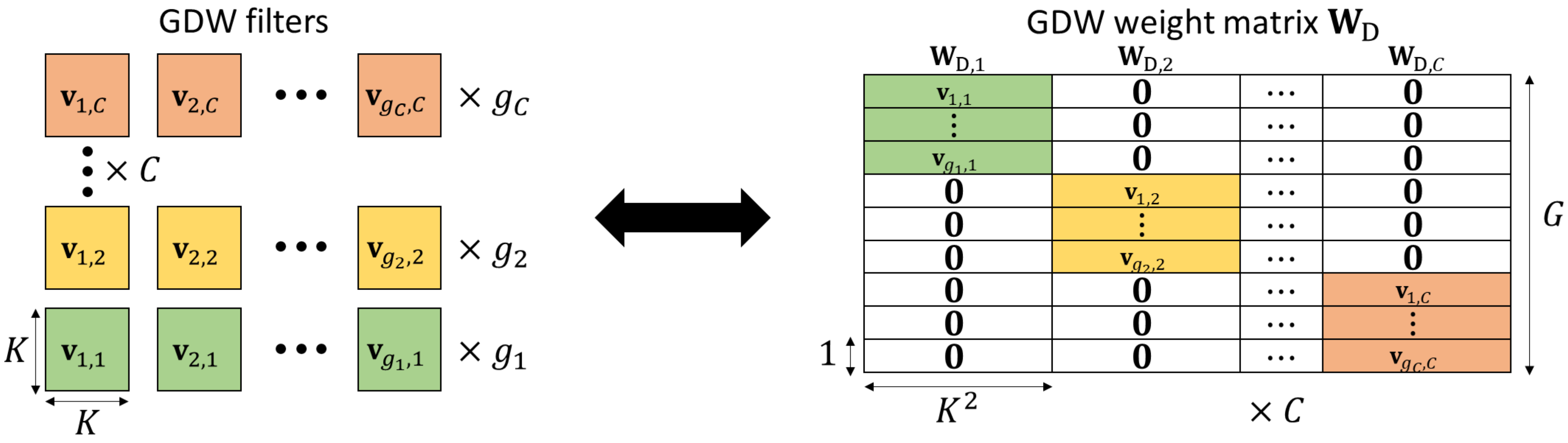
*Every **GDWS** convolution has an equivalent **standard** 2D convolution with weight matrix*:

$$\mathbf{W}_{\mathrm{G}} = \mathbf{W}_{\mathrm{P}} \times \mathbf{W}_{\mathrm{D}} \in \mathbb{R}^{M \times CK^2}$$



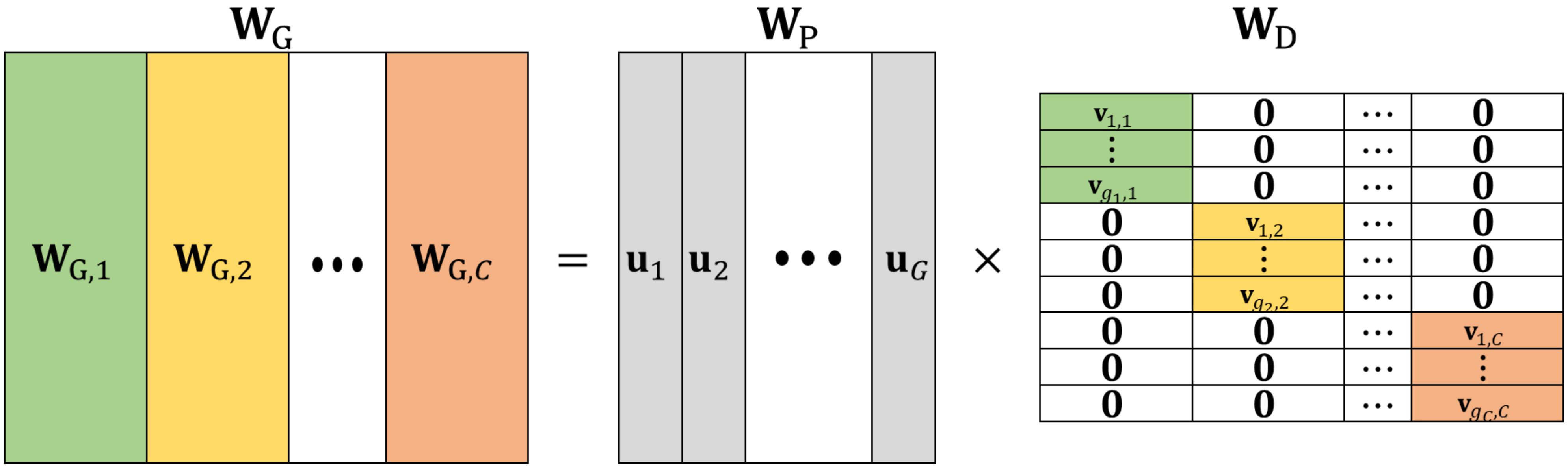
Property 2: GDW Convolution Matrix

The weight matrix of a GDW convolution has a block-diagonal structure:



Structure of GDWS-equivalent Standard Convolution

Lemma. *The* ***GDWS****-equivalent* ***standard*** *2D convolution weight matrix* $\mathbf{W}_\mathrm{G}$ *can be expressed as*:



$$\mathbf{W}_{\mathrm{G},c} \in \mathbb{R}^{M\times K^2} \qquad \& \qquad \mathrm{rank}\left(\mathbf{W}_{\mathrm{G},c}\right) \le \min(g_c, K^2) \;\; \forall c \in [C]$$

Convolution Approximation Error

$$e(\mathbf{W}, \mathbf{Q}, \boldsymbol{\alpha}) = \sqrt{\sum_{c=1}^{C} \alpha_c \|\mathbf{W}_c - \mathbf{Q}_c\|_{\mathrm{F}}^2}$$

where:

- $\mathbf{W} = [\mathbf{W}_1 | \ldots | \mathbf{W}_C]$, $\mathbf{Q} = [\mathbf{Q}_1 | \ldots | \mathbf{Q}_C]$, and $\mathbf{W}_c, \mathbf{Q}_c \in \mathbb{R}^{M \times K^2}$ $\forall c \in [C]$
- $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix
- $\boldsymbol{\alpha} \in \mathbb{R}_+^C$ is the weight error vector

Convolution Approximation Error

$$e(\mathbf{W}, \mathbf{Q}, \boldsymbol{\alpha}) = \sqrt{\sum_{c=1}^{C} \alpha_c \|\mathbf{W}_c - \mathbf{Q}_c\|_{\mathrm{F}}^2}$$

where:

- $\mathbf{W} = [\mathbf{W}_1 | \dots | \mathbf{W}_C]$, $\mathbf{Q} = [\mathbf{Q}_1 | \dots | \mathbf{Q}_C]$, and $\mathbf{W}_c, \mathbf{Q}_c \in \mathbb{R}^{M \times K^2}$ $\forall c \in [C]$
- $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix
- $\boldsymbol{\alpha} \in \mathbb{R}_+^C$ is the weight error vector

- Note that $\alpha_c = 1 \; \forall c \in [C]$ simplifies $e(\mathbf{W}, \mathbf{Q}, \boldsymbol{\alpha})$ to $\|\mathbf{W} - \mathbf{Q}\|_{\mathrm{F}}$

<u>Main Result</u>: Error-constrained Optimal Approximation

Theorem. *Given a* (C, K, M) *standard 2D convolution with weight matrix* $\mathbf{W}$*, the* $(C, K, \mathbf{g}, M)$ *GDWS approximation with weight matrix* $\widehat{\mathbf{W}}$ *that minimizes the complexity* $\gamma(\mathbf{g})$ *subject to* $e\left(\mathbf{W}, \widehat{\mathbf{W}}, \boldsymbol{\alpha}\right) \leq \beta$ *(for some* $\beta \geq 0$*), can be constructed in polynomial time via the* LEGO *Algorithm.*

Main Result: Error-constrained Optimal Approximation

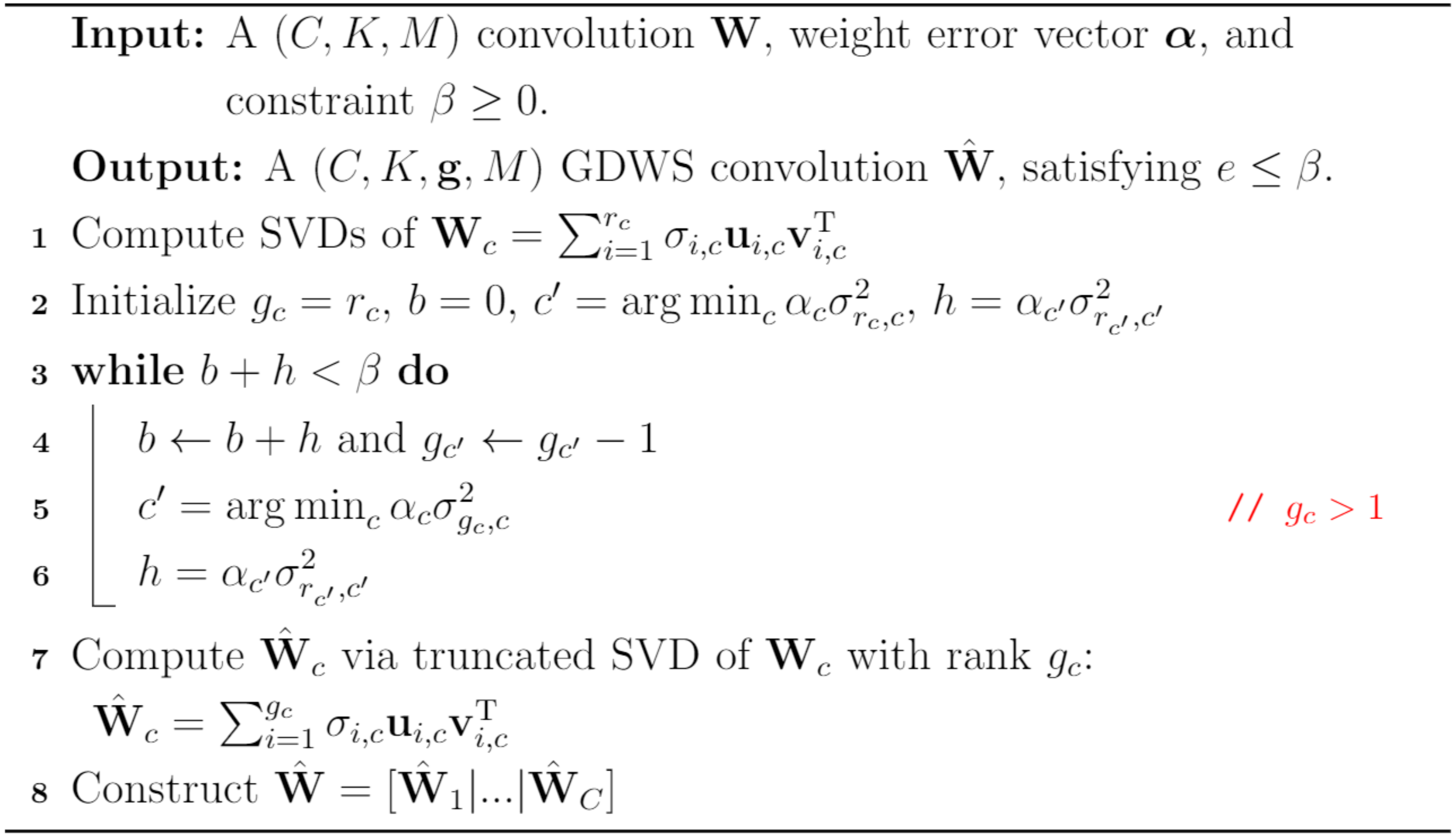
Theorem. *Given a (C, K, M) standard 2D convolution with weight matrix $\mathbf{W}$, the $(C, K, \mathbf{g}, M)$ GDWS approximation with weight matrix $\widehat{\mathbf{W}}$ that minimizes the complexity $\gamma(\mathbf{g})$ subject to $e\left(\mathbf{W}, \widehat{\mathbf{W}}, \boldsymbol{\alpha}\right) \leq \beta$ (for some $\beta \geq 0$), can be constructed in polynomial time via the* LEGO *Algorithm.*

That is:

$$\widehat{\mathbf{W}} = \underset{\mathbf{Q}:\, e(\mathbf{W},\mathbf{Q},\boldsymbol{\alpha}) \leq \beta}{\operatorname{argmin}} \gamma(\mathbf{g}) = \underset{\mathbf{Q}:\, e(\mathbf{W},\mathbf{Q},\boldsymbol{\alpha}) \leq \beta}{\operatorname{argmin}} \sum_{c=1}^{C} g_c$$

can be solved $\forall \boldsymbol{\alpha} \in \mathbb{R}_{+}^{C}$ optimally and efficiently

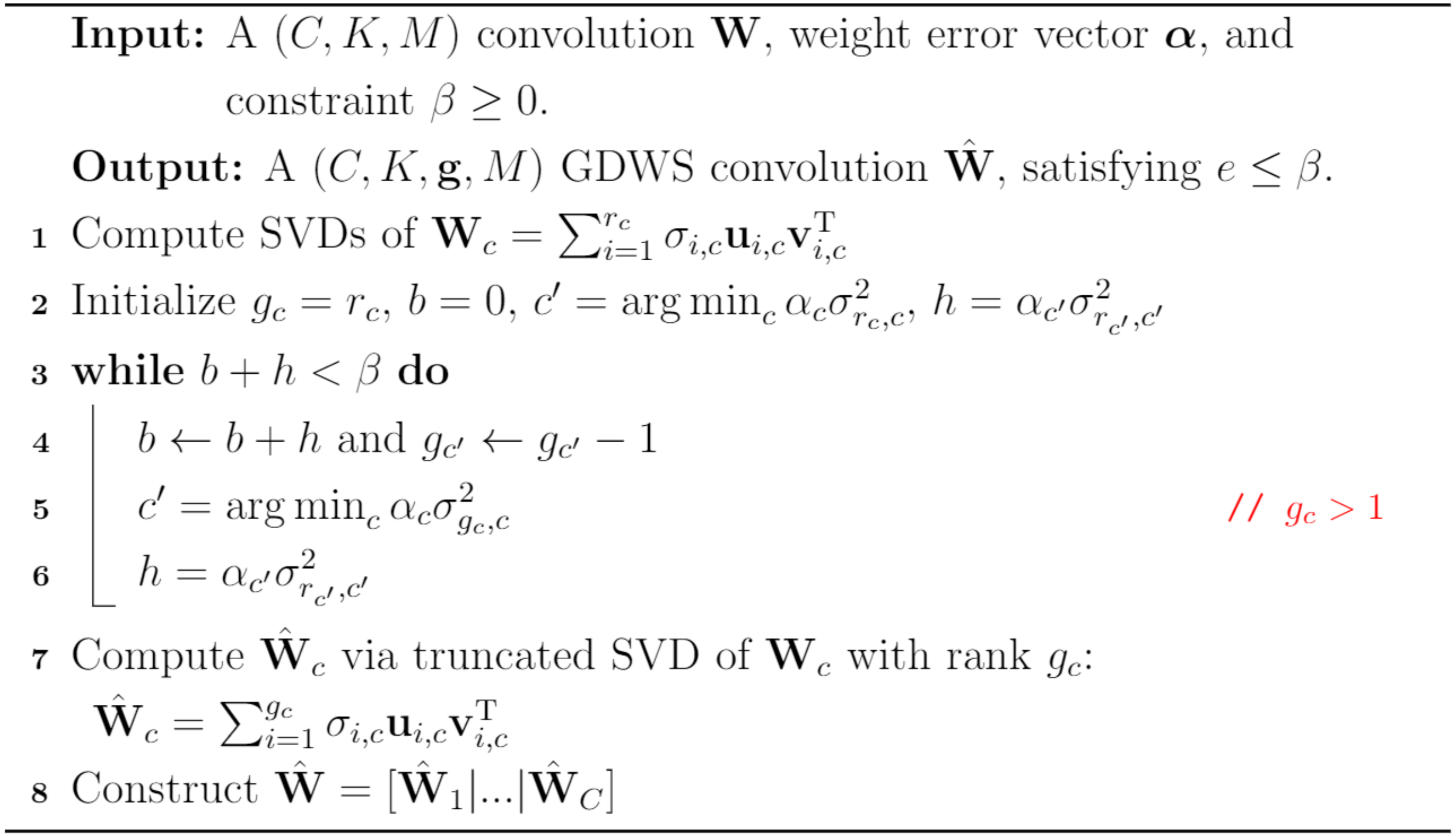
LEGO: Least Complex Error-constrained GDWS Optimal Approximation



```
Input: A $(C, K, M)$ convolution $\mathbf{W}$, weight error vector $\boldsymbol{\alpha}$, and
       constraint $\beta \geq 0$.
Output: A $(C, K, \mathbf{g}, M)$ GDWS convolution $\hat{\mathbf{W}}$, satisfying $e \leq \beta$.
Compute SVDs of $\mathbf{W}_c = \sum_{i=1}^{r_c} \sigma_{i,c}\mathbf{u}_{i,c}\mathbf{v}_{i,c}^{\mathrm{T}}$
Initialize $g_c = r_c$, $b = 0$, $c' = \arg\min_c \alpha_c\sigma_{r_c,c}^2$, $h = \alpha_{c'}\sigma_{r_{c'},c'}^2$
while $b + h < \beta$ do
    $b \leftarrow b + h$ and $g_{c'} \leftarrow g_{c'} - 1$
    $c' = \arg\min_c \alpha_c\sigma_{g_c,c}^2$                         // $g_c > 1$
    $h = \alpha_{c'}\sigma_{r_{c'},c'}^2$
Compute $\hat{\mathbf{W}}_c$ via truncated SVD of $\mathbf{W}_c$ with rank $g_c$:
    $\hat{\mathbf{W}}_c = \sum_{i=1}^{g_c} \sigma_{i,c}\mathbf{u}_{i,c}\mathbf{v}_{i,c}^{\mathrm{T}}$
Construct $\hat{\mathbf{W}} = [\hat{\mathbf{W}}_1|...|\hat{\mathbf{W}}_C]$
```

- greedy construction algorithm

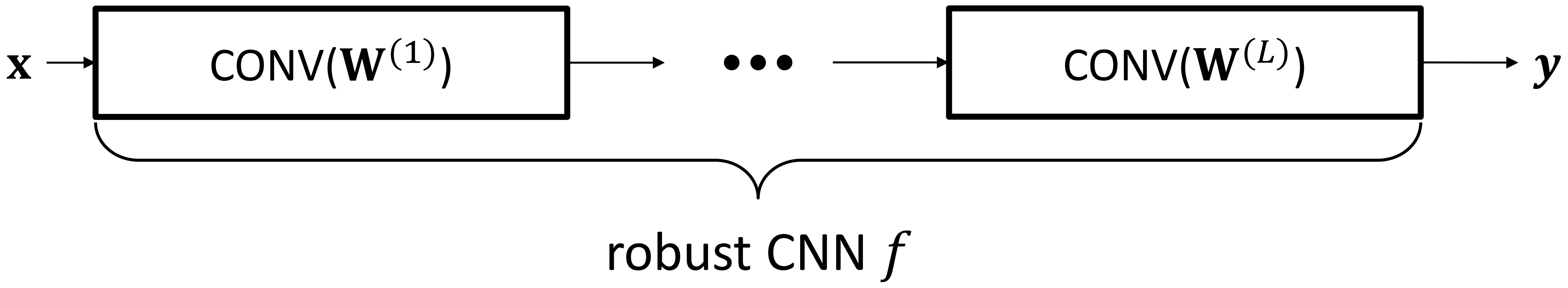
LEGO: Least Complex Error-constrained GDWS Optimal Approximation



```
Input: A $(C, K, M)$ convolution $\mathbf{W}$, weight error vector $\boldsymbol{\alpha}$, and
       constraint $\beta \geq 0$.
Output: A $(C, K, \mathbf{g}, M)$ GDWS convolution $\hat{\mathbf{W}}$, satisfying $e \leq \beta$.
Compute SVDs of $\mathbf{W}_c = \sum_{i=1}^{r_c} \sigma_{i,c}\mathbf{u}_{i,c}\mathbf{v}_{i,c}^{\mathrm{T}}$
Initialize $g_c = r_c$, $b = 0$, $c' = \arg\min_c \alpha_c \sigma_{r_c,c}^2$, $h = \alpha_{c'}\sigma_{r_{c'},c'}^2$
while $b + h < \beta$ do
    $b \leftarrow b + h$ and $g_{c'} \leftarrow g_{c'} - 1$
    $c' = \arg\min_c \alpha_c \sigma_{g_c,c}^2$                              // $g_c > 1$
    $h = \alpha_{c'}\sigma_{r_{c'},c'}^2$
Compute $\hat{\mathbf{W}}_c$ via truncated SVD of $\mathbf{W}_c$ with rank $g_c$:
    $\hat{\mathbf{W}}_c = \sum_{i=1}^{g_c} \sigma_{i,c}\mathbf{u}_{i,c}\mathbf{v}_{i,c}^{\mathrm{T}}$
Construct $\hat{\mathbf{W}} = [\hat{\mathbf{W}}_1|...|\hat{\mathbf{W}}_C]$
```

- optimality due to (1) Eckart-Young [Psych., 1936] & (2) GDWS Lemma

Constructing GDWS Networks



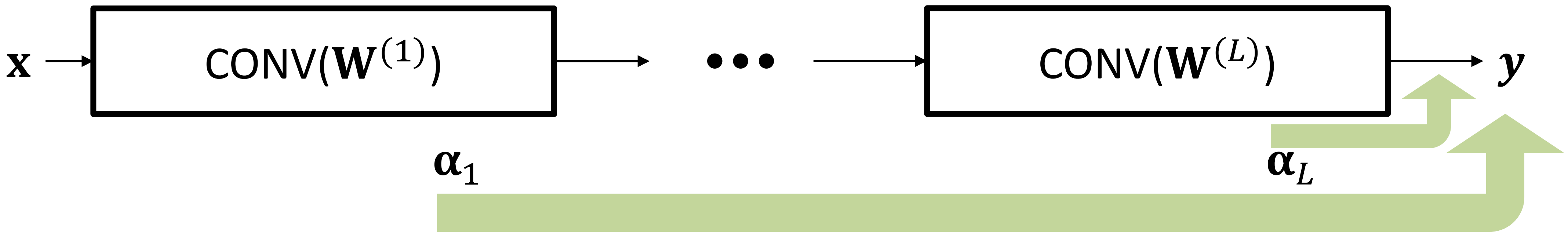
$\mathbf{x}$ → CONV($\mathbf{W}^{(1)}$) → ••• → CONV($\mathbf{W}^{(L)}$) → $\boldsymbol{y}$

$\underbrace{\hphantom{xxxxxxxxxxxx}}$

robust CNN f

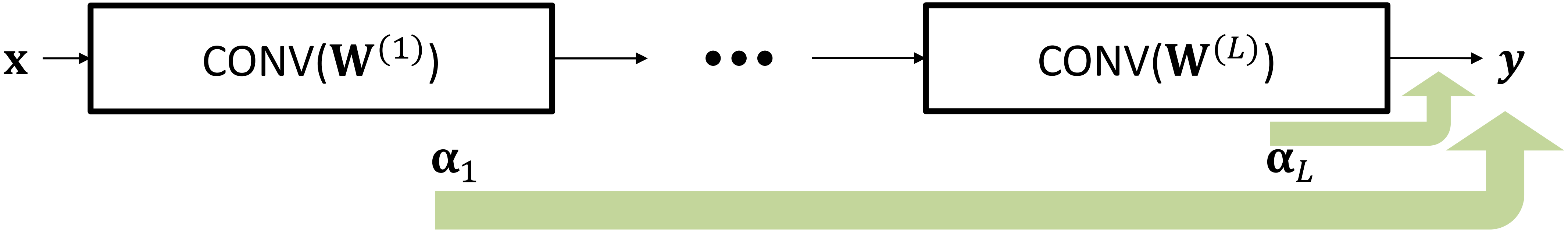
① start with a pre-trained robust CNN f

Constructing GDWS Networks



② compute the per-layer sensitivity based $\boldsymbol{\alpha}_l$

Constructing GDWS Networks



$\mathbf{x}$ → CONV($\mathbf{W}^{(1)}$) → ••• → CONV($\mathbf{W}^{(L)}$) → $\boldsymbol{y}$

$\boldsymbol{\alpha}_1$

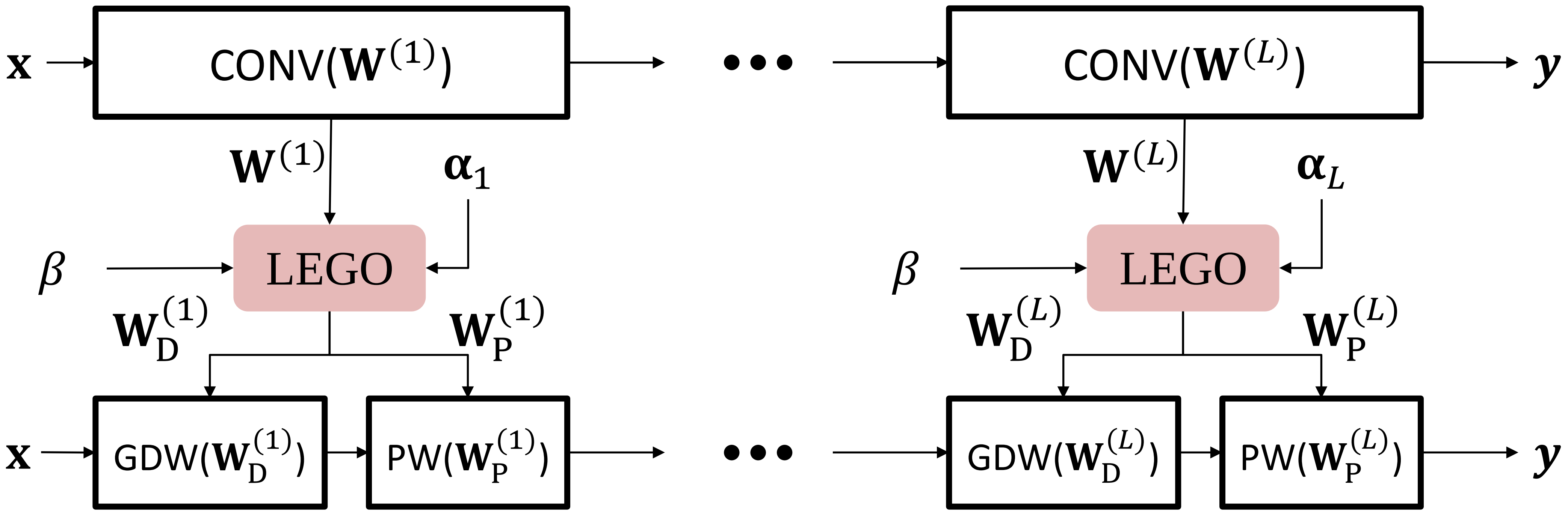
$\boldsymbol{\alpha}_L$

inspired by [Sakr et al., ICML'17]:

$$\alpha_{c,l} = \mathbb{E}_x\left[\sum_{\substack{j=1\\ j\neq n_x}}^{N} \frac{\left\|\mathbf{D}_{x,j}^{(c,l)}\right\|_{\mathrm{F}}^{2}}{2\delta_{x,j}^{2}}\right] \quad \forall l \in [L], \forall c \in [C_l]$$

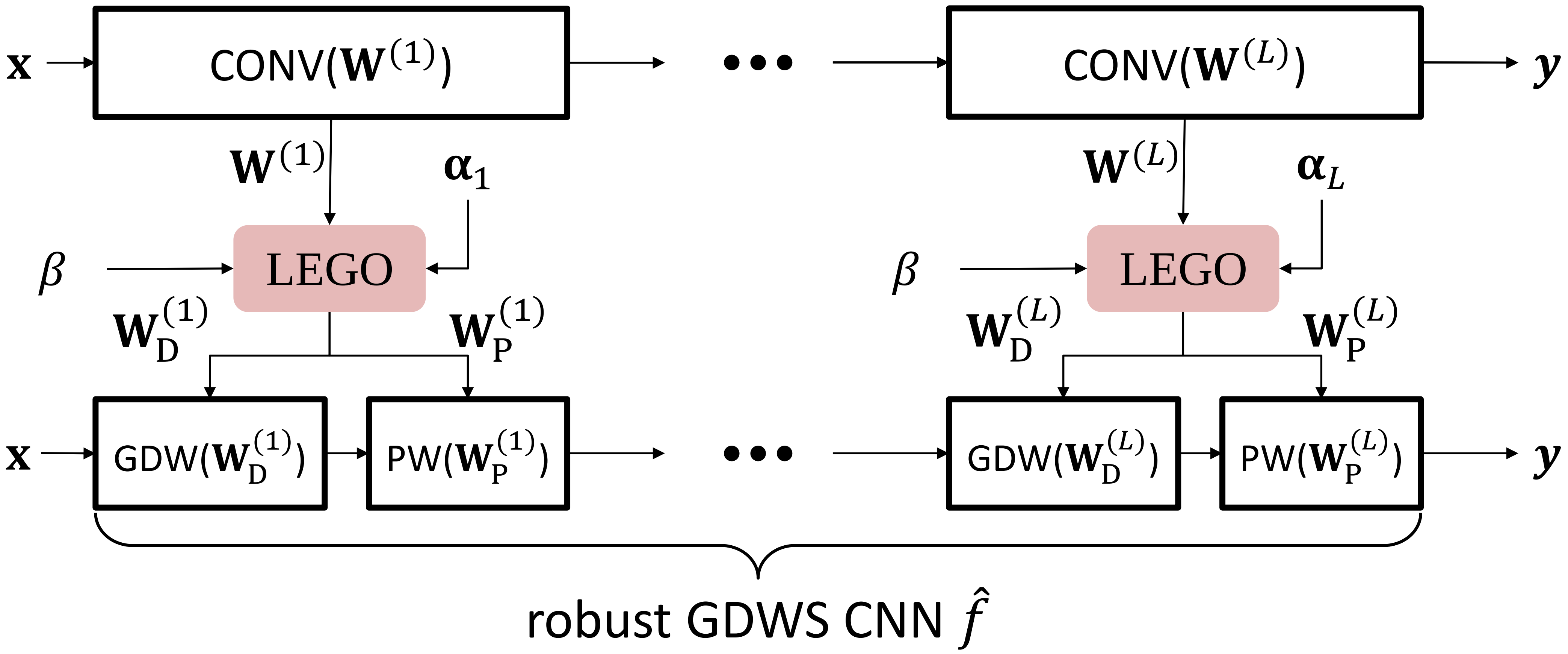
② compute the per-layer sensitivity based $\boldsymbol{\alpha}_l$

Constructing GDWS Networks



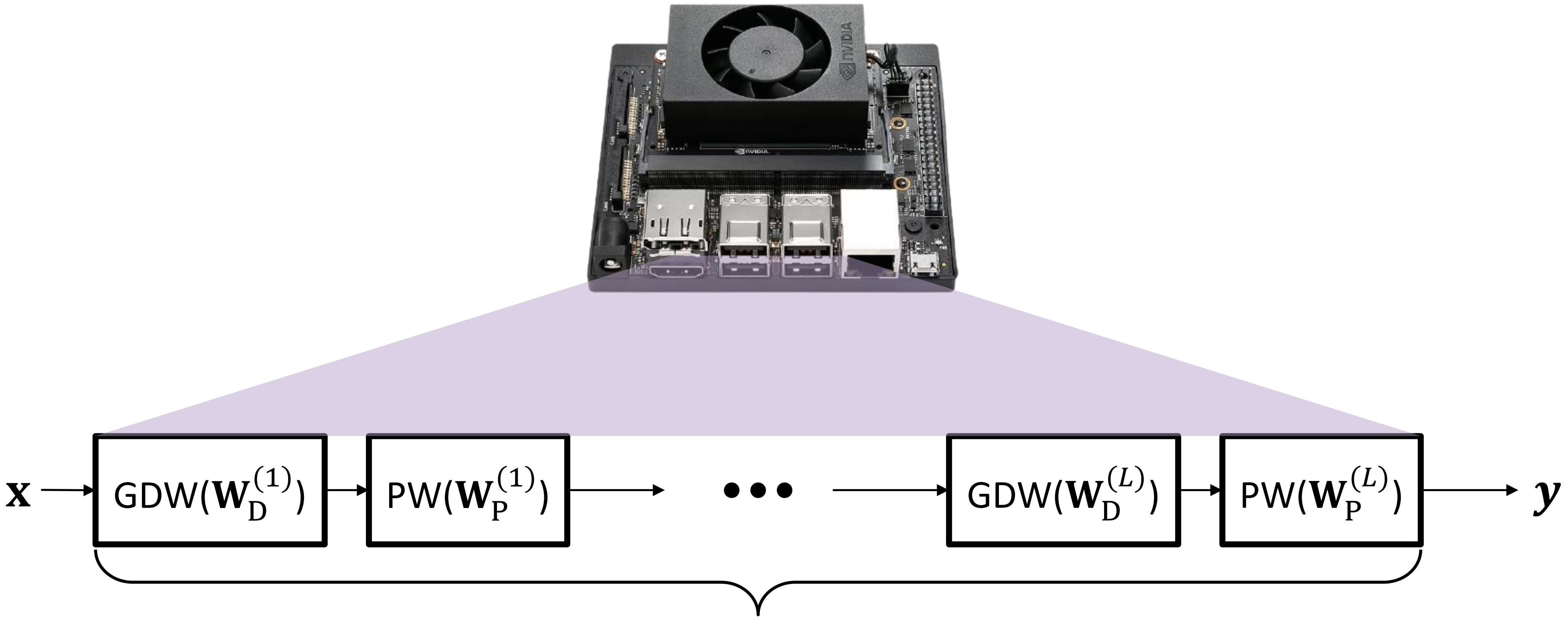
③ construct per-layer optimal GDWS approximation with constraint β

Constructing GDWS Networks



③ construct per-layer optimal GDWS approximation with constraint β

Constructing GDWS Networks



$$\mathbf{x} \rightarrow \underbrace{\boxed{\text{GDW}(\mathbf{W}_{\text{D}}^{(1)})} \rightarrow \boxed{\text{PW}(\mathbf{W}_{\text{P}}^{(1)})} \rightarrow \bullet\bullet\bullet \rightarrow \boxed{\text{GDW}(\mathbf{W}_{\text{D}}^{(L)})} \rightarrow \boxed{\text{PW}(\mathbf{W}_{\text{P}}^{(L)})}}_{\text{robust GDWS CNN } \hat{f}} \rightarrow \boldsymbol{y}$$

④ deploy $\hat{f}$ on the Jetson

Experimental Results & Comparisons

Pre-adversarially Trained Networks– CIFAR-10

Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
ResNet-50	84.21	53.05	89.7	16
+ GDWS ($\beta = 0.001$)	**83.72**	**52.94**	81.9	**37**
WRN-28-4	84.00	51.80	22.3	17
+ GDWS ($\beta = 1 \times 10^{-5}$)	**83.27**	**51.70**	18.9	**65**
ResNet-18	82.41	51.55	42.6	28
+ GDWS ($\beta = 0.005$)	**81.17**	**50.98**	29.1	**104**
VGG-16	77.49	48.92	56.2	36
+ GDWS ($\beta = 0.25$)	**77.17**	**49.56**	28.7	**129**

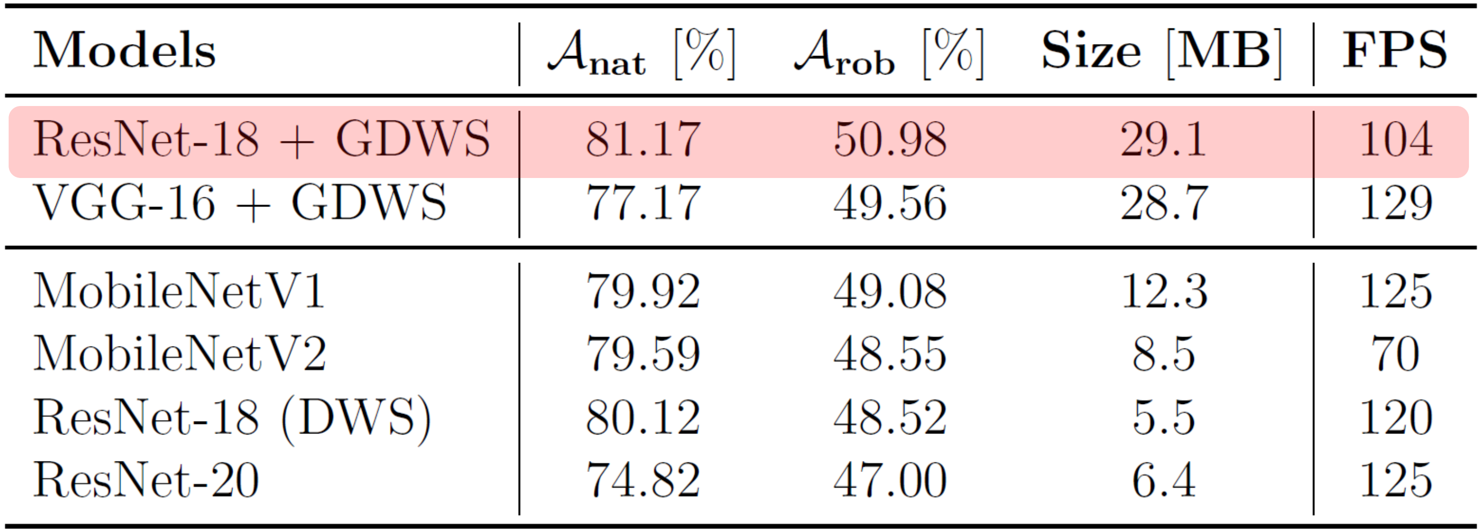
- preserves both $\mathcal{A}_{\text{rob}}$ and $\mathcal{A}_{\text{nat}}$ of original baselines
- dramatically improves the **FPS** in spite of modest reductions in **model size**

Comparison with Lightweight Networks – CIFAR-10

- natural question: *why not train lightweight networks from scratch, instead of approximating pre-trained complex networks with GDWS?*

Comparison with Lightweight Networks – CIFAR-10

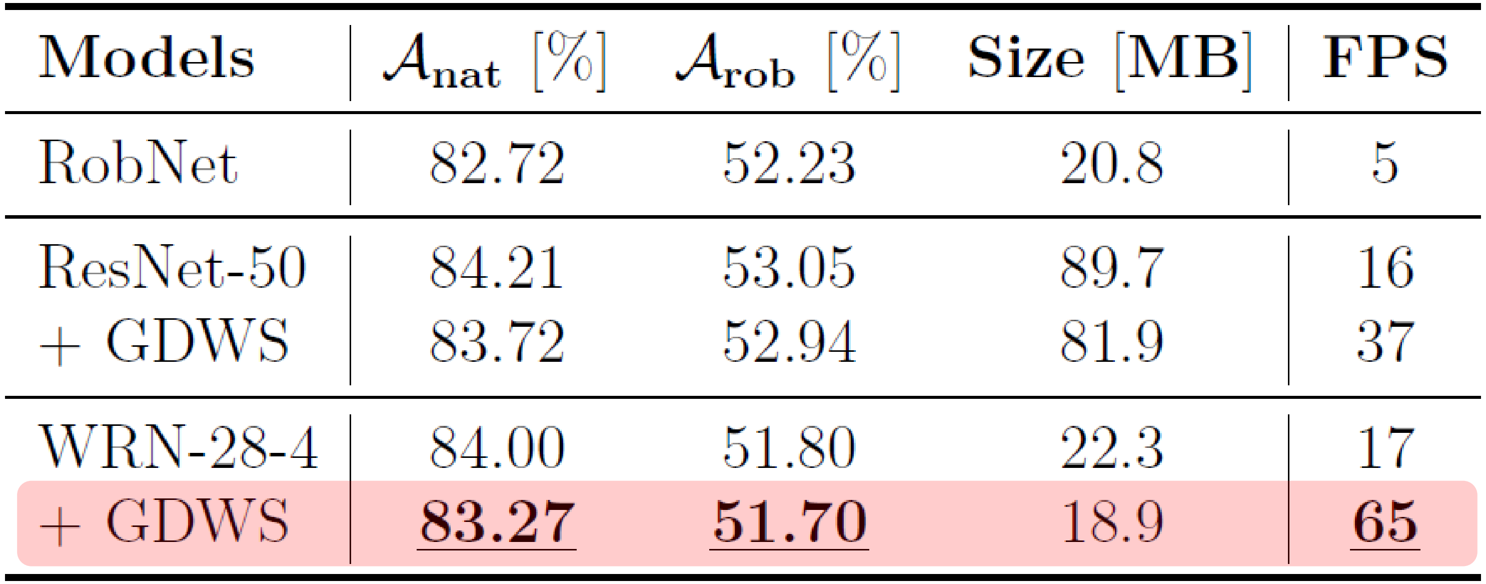
- natural question: *why not train lightweight networks from scratch, instead of approximating pre-trained complex networks with GDWS?*



Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
ResNet-18 + GDWS	81.17	50.98	29.1	104
VGG-16 + GDWS	77.17	49.56	28.7	129
MobileNetV1	79.92	49.08	12.3	125
MobileNetV2	79.59	48.55	8.5	70
ResNet-18 (DWS)	80.12	48.52	5.5	120
ResNet-20	74.82	47.00	6.4	125

- better $\mathcal{A}_{rob}$ and $\mathcal{A}_{nat}$ than all lightweight networks
- DWS-like **FPS** and requiring no extra training

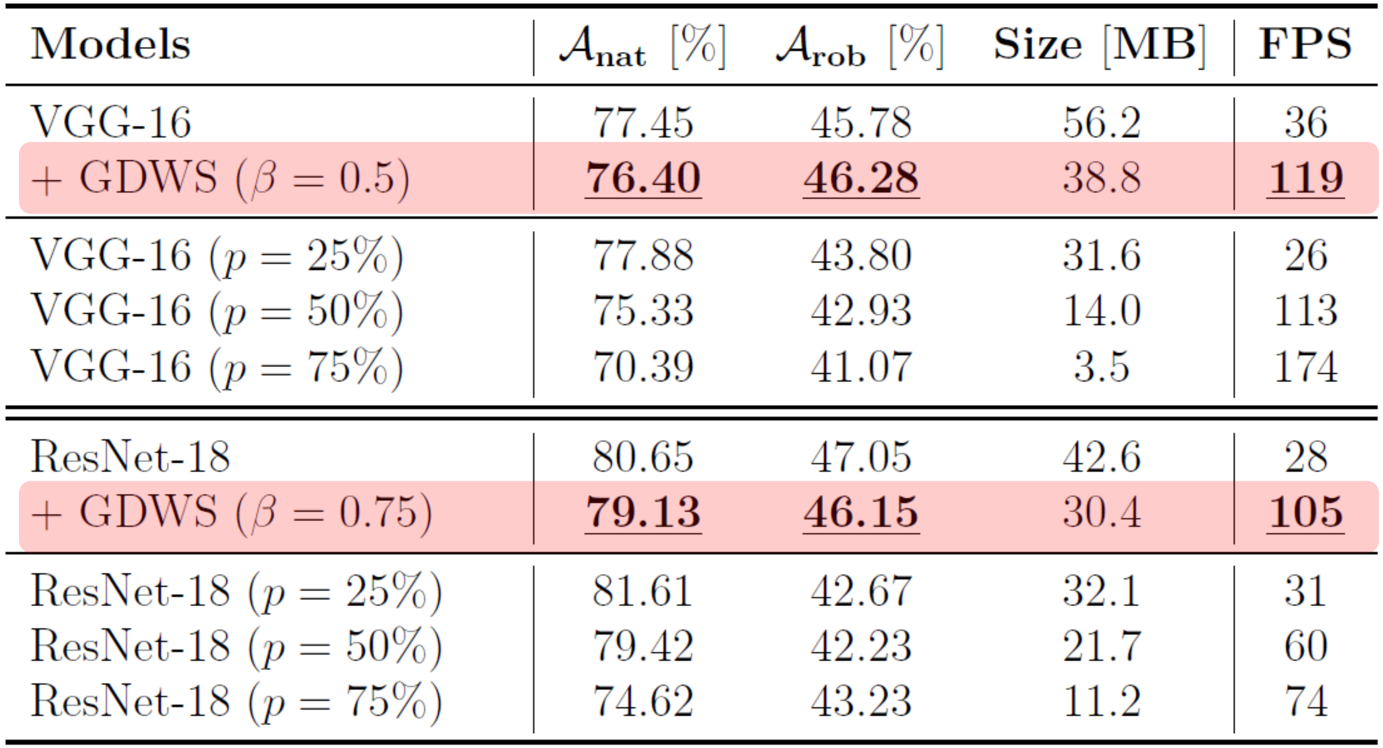
Comparison with RobNet [CVPR'20]– CIFAR-10



Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
RobNet	82.72	52.23	20.8	5
ResNet-50	84.21	53.05	89.7	16
+ GDWS	83.72	52.94	81.9	37
WRN-28-4	84.00	51.80	22.3	17
+ GDWS	**83.27**	**51.70**	18.9	**65**

- RobNet: irregular cell structure leads to poor **FPS** on Jetson
- GDWS + WRN-28-4: similar **robustness**, drastic improvements in **FPS**

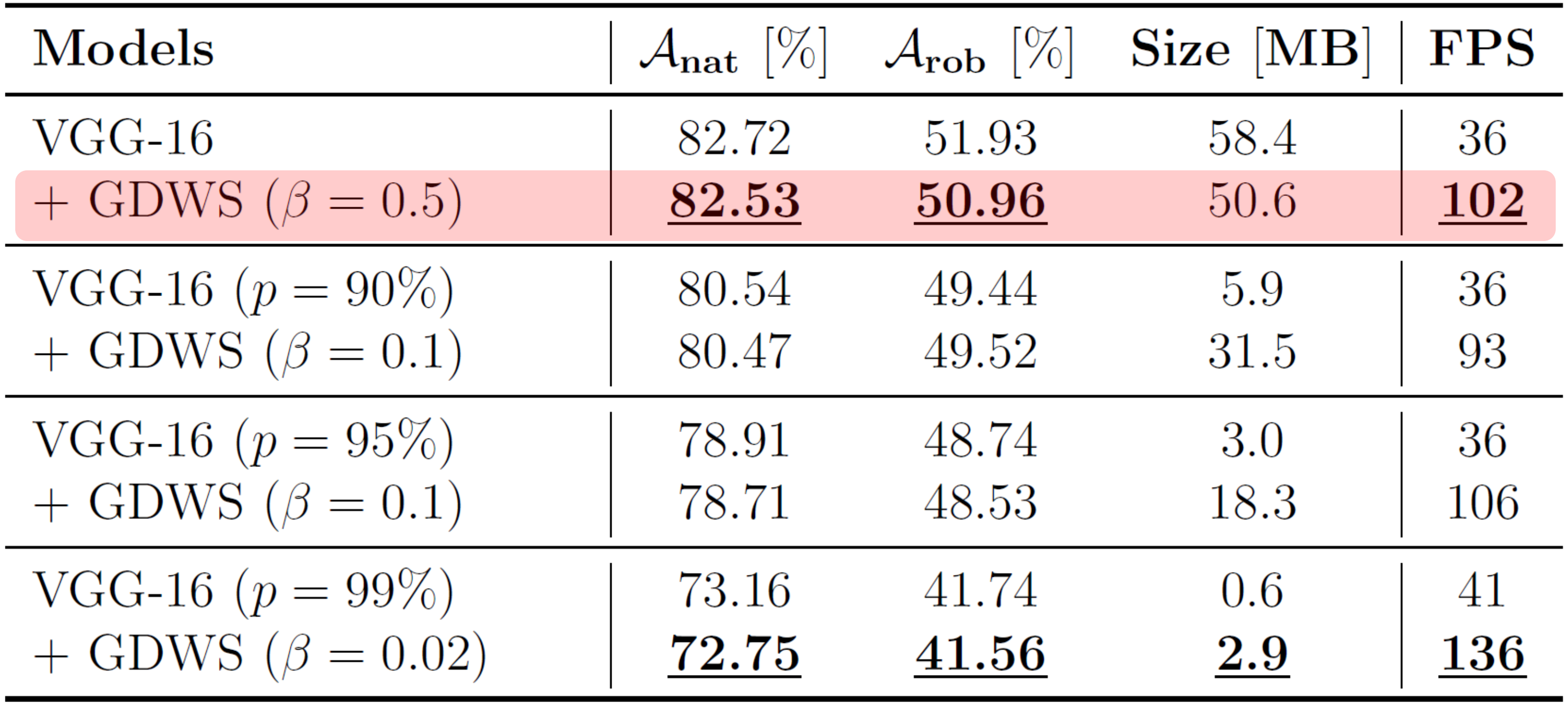
Comparison with ADMM [ICCV’19]– CIFAR-10



Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
VGG-16	77.45	45.78	56.2	36
+ GDWS ($\beta = 0.5$)	**76.40**	**46.28**	38.8	**119**
VGG-16 ($p = 25\%$)	77.88	43.80	31.6	26
VGG-16 ($p = 50\%$)	75.33	42.93	14.0	113
VGG-16 ($p = 75\%$)	70.39	41.07	3.5	174
ResNet-18	80.65	47.05	42.6	28
+ GDWS ($\beta = 0.75$)	**79.13**	**46.15**	30.4	**105**
ResNet-18 ($p = 25\%$)	81.61	42.67	32.1	31
ResNet-18 ($p = 50\%$)	79.42	42.23	21.7	60
ResNet-18 ($p = 75\%$)	74.62	43.23	11.2	74

- ADMM: high **FPS**, compromises **robustness**
- GDWS: high **FPS**, preserves **robustness**

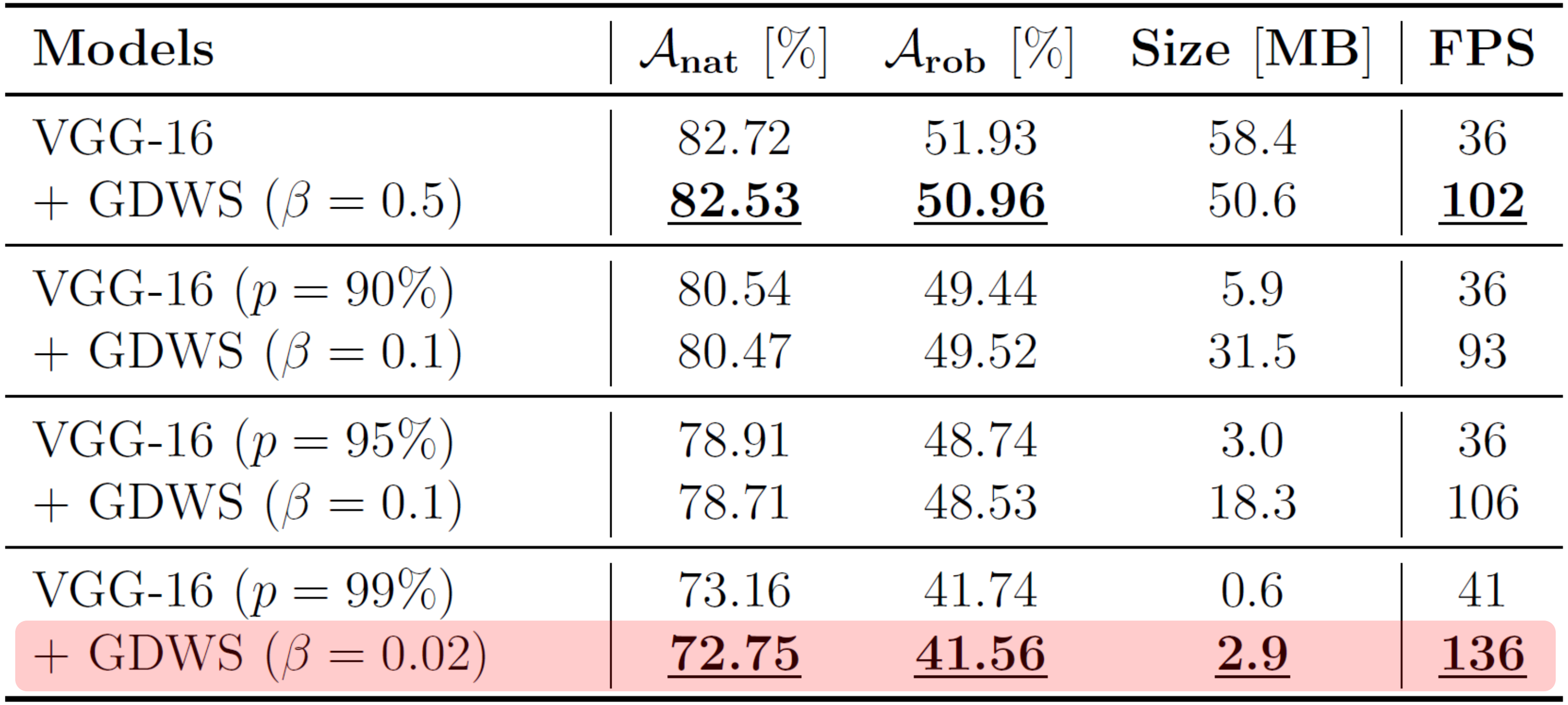
Comparison with HYDRA [NeurIPs'20]– CIFAR-10



Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
VGG-16	82.72	51.93	58.4	36
+ GDWS ($\beta = 0.5$)	**82.53**	**50.96**	50.6	**102**
VGG-16 ($p = 90\%$)	80.54	49.44	5.9	36
+ GDWS ($\beta = 0.1$)	80.47	49.52	31.5	93
VGG-16 ($p = 95\%$)	78.91	48.74	3.0	36
+ GDWS ($\beta = 0.1$)	78.71	48.53	18.3	106
VGG-16 ($p = 99\%$)	73.16	41.74	0.6	41
+ GDWS ($\beta = 0.02$)	**72.75**	**41.56**	**2.9**	**136**

- HYDRA: compromises **robustness**, minimal improvements in **FPS**
- GDWS: preserves **robustness** and boosts **FPS** significantly

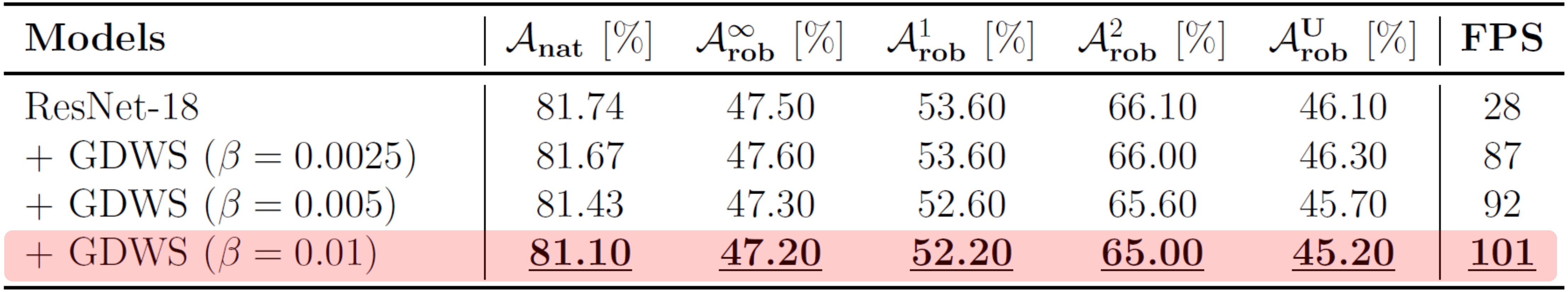
Comparison with HYDRA [NeurIPs’20]– CIFAR-10



Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}_{\text{rob}}$ [%]	**Size** [MB]	**FPS**
VGG-16	82.72	51.93	58.4	36
+ GDWS ($\beta = 0.5$)	**82.53**	**50.96**	50.6	**102**
VGG-16 ($p = 90\%$)	80.54	49.44	5.9	36
+ GDWS ($\beta = 0.1$)	80.47	49.52	31.5	93
VGG-16 ($p = 95\%$)	78.91	48.74	3.0	36
+ GDWS ($\beta = 0.1$)	78.71	48.53	18.3	106
VGG-16 ($p = 99\%$)	73.16	41.74	0.6	41
+ GDWS ($\beta = 0.02$)	**72.75**	**41.56**	**2.9**	**136**

- GDWS + HYDRA: high compression ratios, preserves **robustness**, and massive improvements in **FPS** compared to the pruned baseline

Defending against Union of Perturbation Models – CIFAR-10

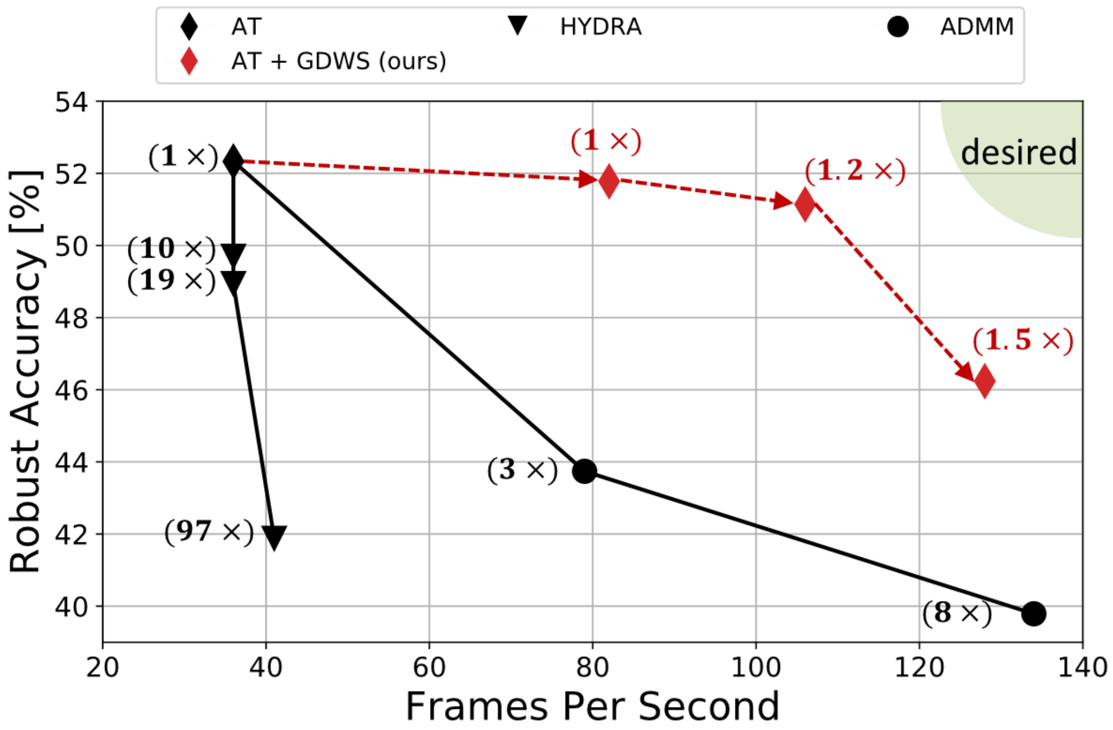


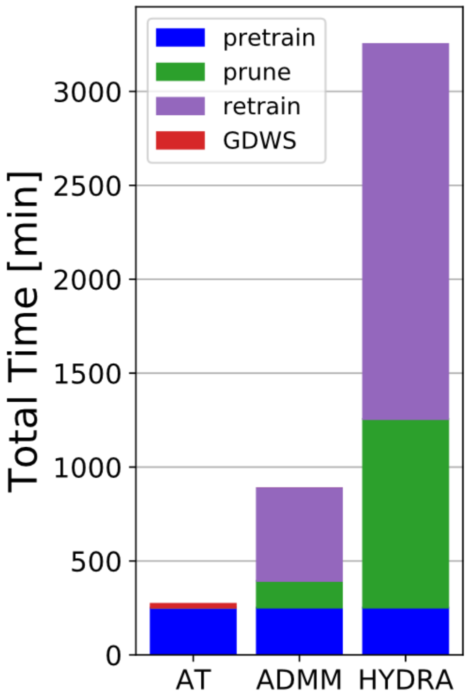
Models	$\mathcal{A}_{\text{nat}}$ [%]	$\mathcal{A}^{\infty}_{\text{rob}}$ [%]	$\mathcal{A}^{1}_{\text{rob}}$ [%]	$\mathcal{A}^{2}_{\text{rob}}$ [%]	$\mathcal{A}^{\text{U}}_{\text{rob}}$ [%]	**FPS**
ResNet-18	81.74	47.50	53.60	66.10	46.10	28
+ GDWS ($\beta = 0.0025$)	81.67	47.60	53.60	66.00	46.30	87
+ GDWS ($\beta = 0.005$)	81.43	47.30	52.60	65.60	45.70	92
+ GDWS ($\beta = 0.01$)	**81.10**	**47.20**	**52.20**	**65.00**	**45.20**	**101**

- pre-trained MSD models from [Maini et al., ICML'20]

- GDWS: negligible drop in $\mathcal{A}_{nat}$ and $\mathcal{A}^{\mathbf{U}}_{rob}$ while improving the **FPS**

Summary





- GDWS convolutions are *universal* and *efficient* approximations of 2D convolutions
- dramatically improve **FPS** while preserving **robust accuracy**
- operate on pre-trained models → *no additional training*

Thank You!

Acknowledgement:

This work was supported by the Center for Brain-Inspired Computing (C-BRIC) and Artificial Intelligence Hardware (AIHW) funded by the Semiconductor Research Corporation (SRC) and the Defense Advanced Research Projects Agency (DARPA).