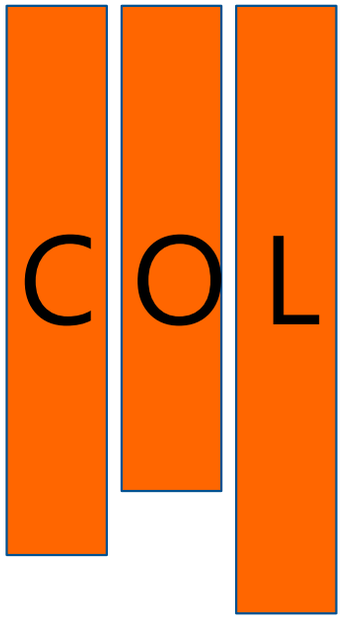
Input and Model Compression for Adaptive and Robust CNNs

Adam Dziedzic

6th December, 2019

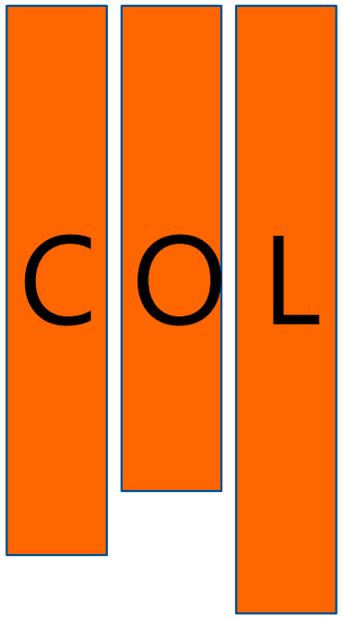
Compression is crucial in Systems and ML

DBMS

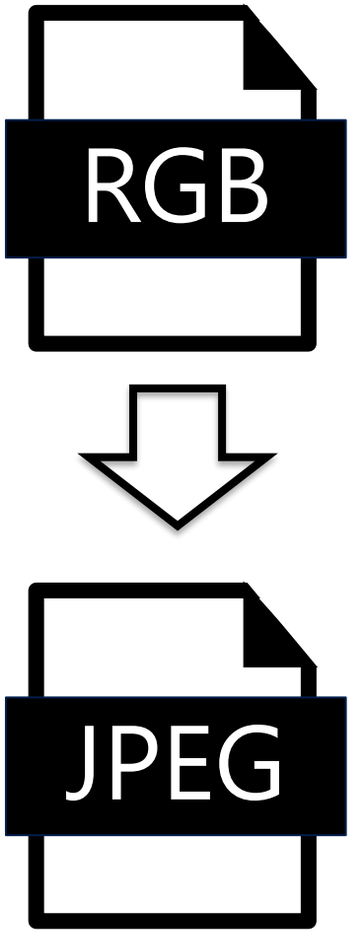


Compression is crucial in Systems and ML

DBMS

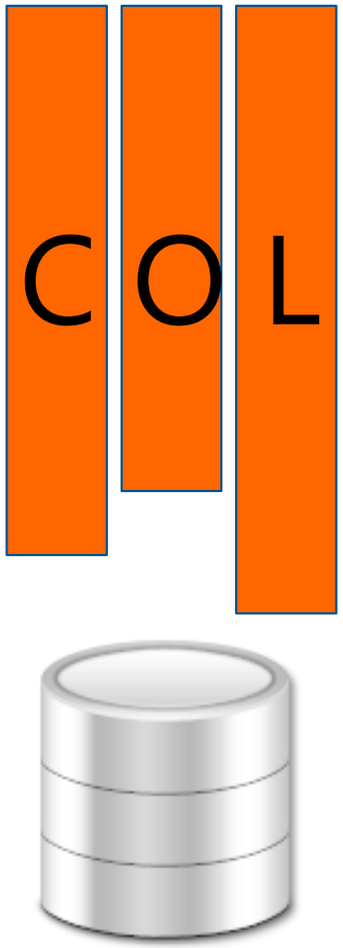


STORAGE

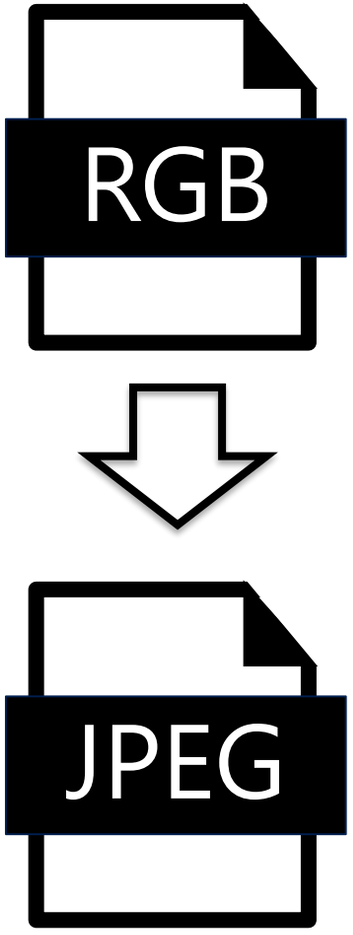


Compression is crucial in Systems and ML

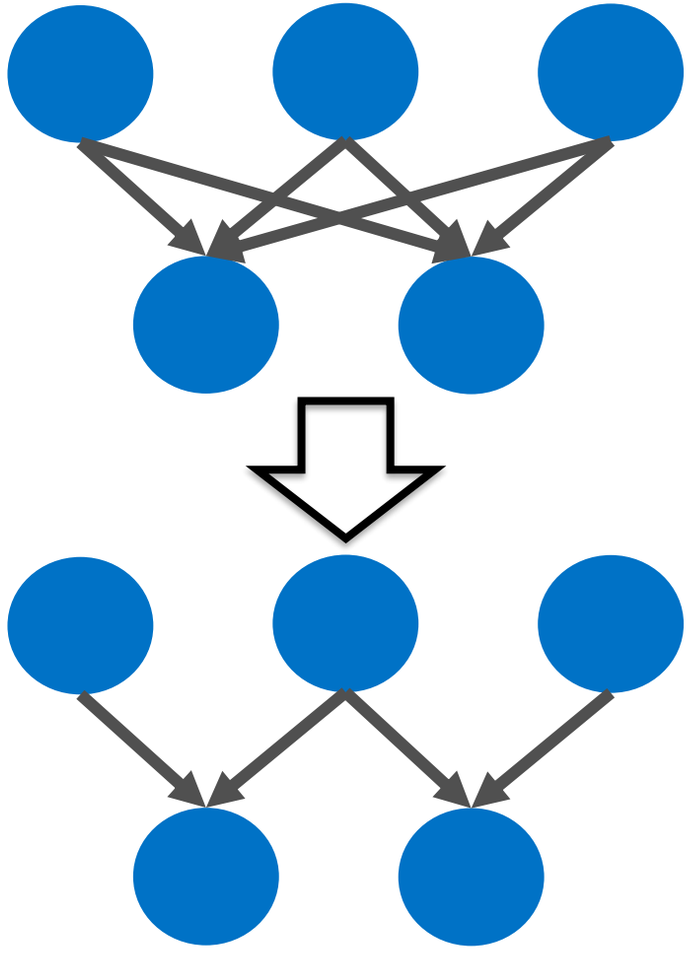
DBMS



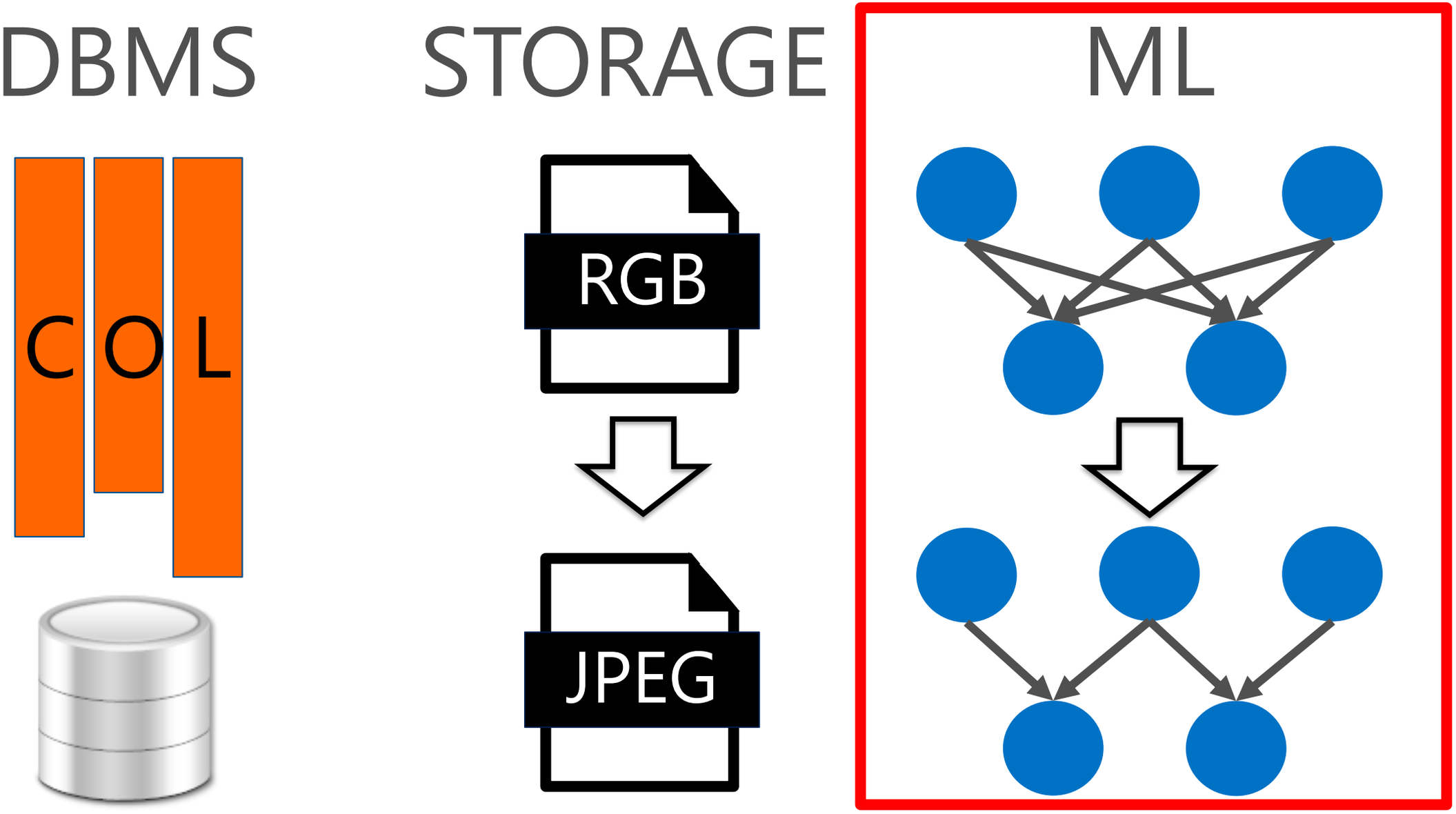
STORAGE



ML



Compression is crucial in Systems and ML



Compression of Neural Networks

1. Inference

[1] Han et al. Compressing deep neural networks with **pruning, trained quantization and huffman encoding**.

[2] You et al. **Gate *Decorator***: *Global Filter Pruning Method*.

Compression of Neural Networks

1. Inference

[1] Han et al. Compressing deep neural networks with **pruning, trained quantization and huffman encoding**.

[2] You et al. **Gate *Decorator***: *Global Filter Pruning Method*.

Compression of Neural Networks

1. Inference

[1] Han et al. Compressing deep neural networks with **pruning, trained quantization and huffman encoding**.

[2] You et al. **Gate *Decorator***: *Global Filter Pruning Method*.

2. Training

[3] Mickievicius et al. **Mixed precision training** (from NVIDIA).

[4] Gueguen et al. **Faster neural networks** straight from **jpeg.** (from UBER)

Compression of Neural Networks

1. Inference

[1] Han et al. Compressing deep neural networks with **pruning, trained quantization and huffman encoding**.

[2] You et al. **Gate *Decorator***: *Global Filter Pruning Method*.

2. Training

[3] Mickievicius et al. **Mixed precision training** (from NVIDIA).

[4] Gueguen et al. **Faster neural networks** straight from **jpeg.** (from UBER)

3. Robustness

[5] Dziugaite et al. *Study of the effect of **jpg compression** on adversarial images.*

[6] Xu et al. ***Feature squeezing**: Detecting adversarial examples in deep neural networks.*

Thesis question

How and when does compression improve:

Convolution

Band-limited training and inference for CNNs

Thesis question

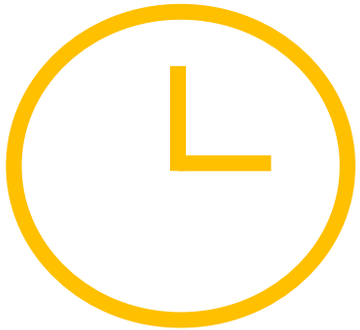
How and when does compression improve:

Convolution

Band-limited training and inference for CNNs

Robustness

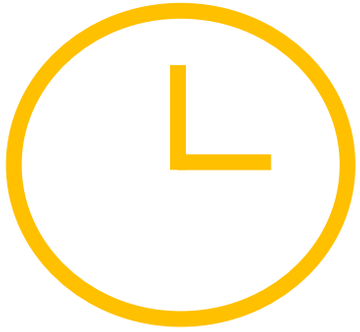
Perturbation analysis of adversarial examples



Thesis question

How and when does compression improve:

Convolution	Robustness	Training
Band-limited training and inference for CNNs	Perturbation analysis of adversarial examples	Adaptive and robust training of CNNs



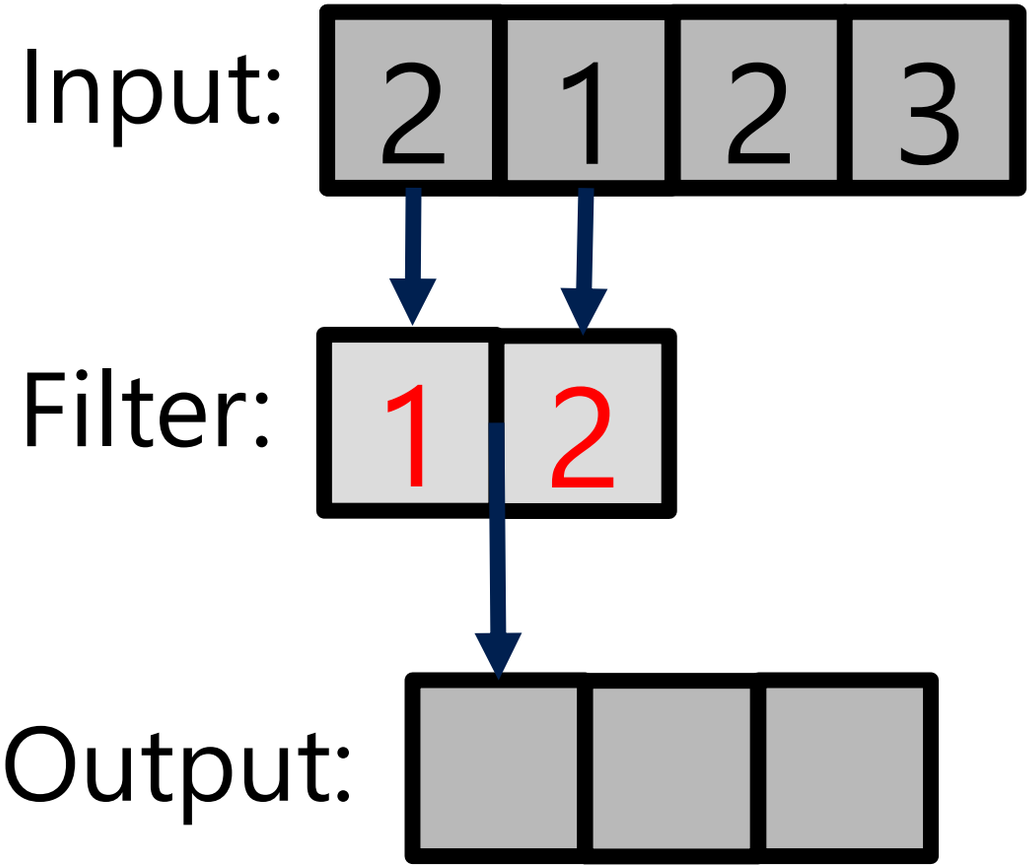
1. Introduction & FFT-based convolution

2. Attacks and defenses

3. Research plan

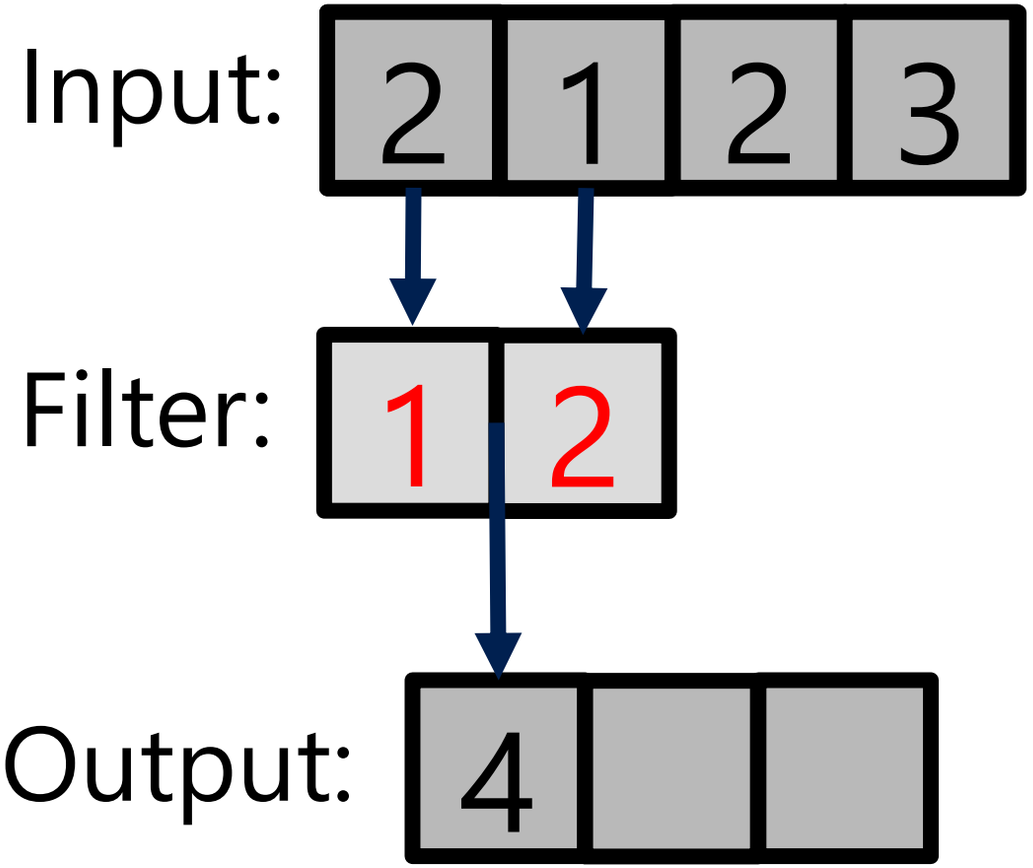
Convolutions in Deep Neural Networks

Direct

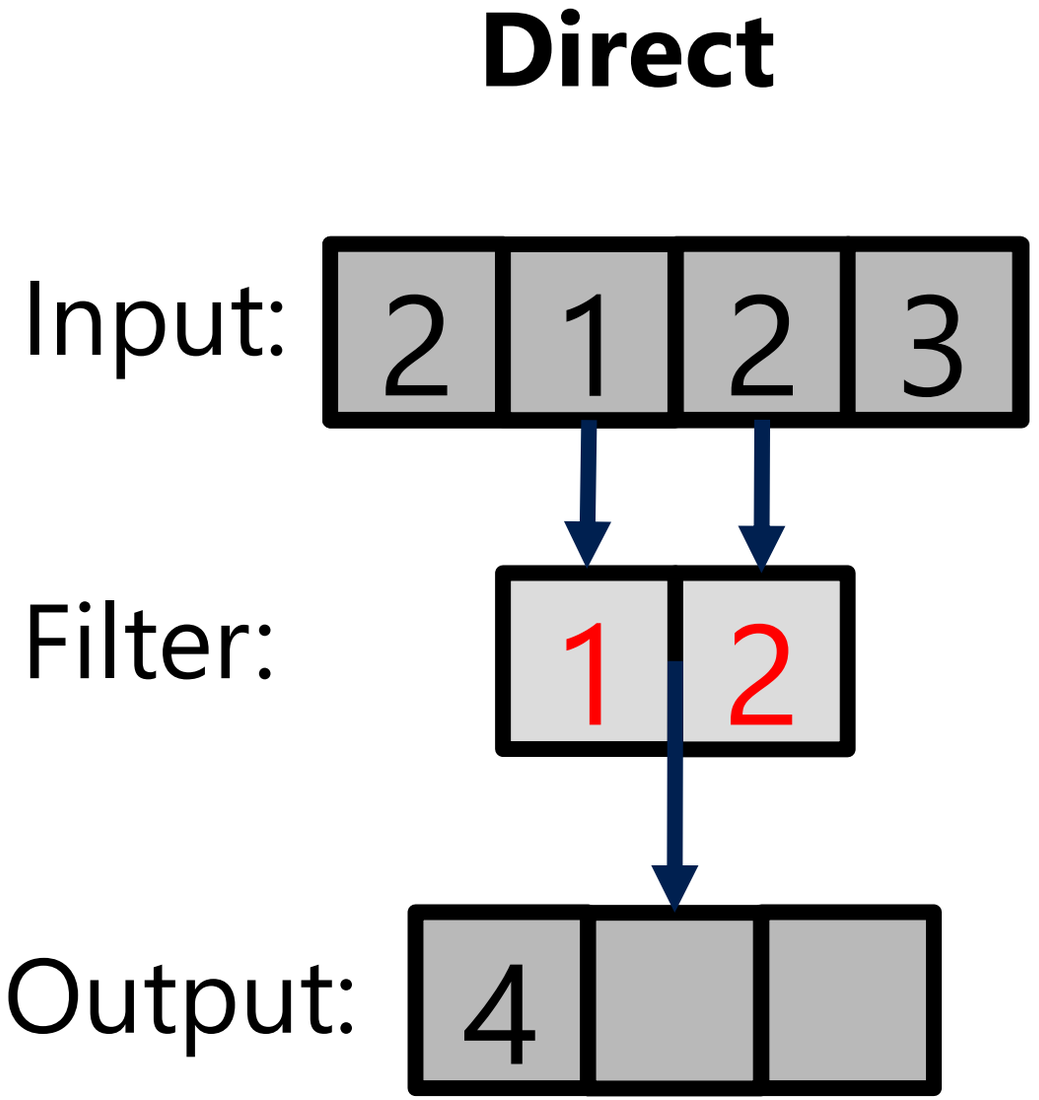


Convolutions in Deep Neural Networks

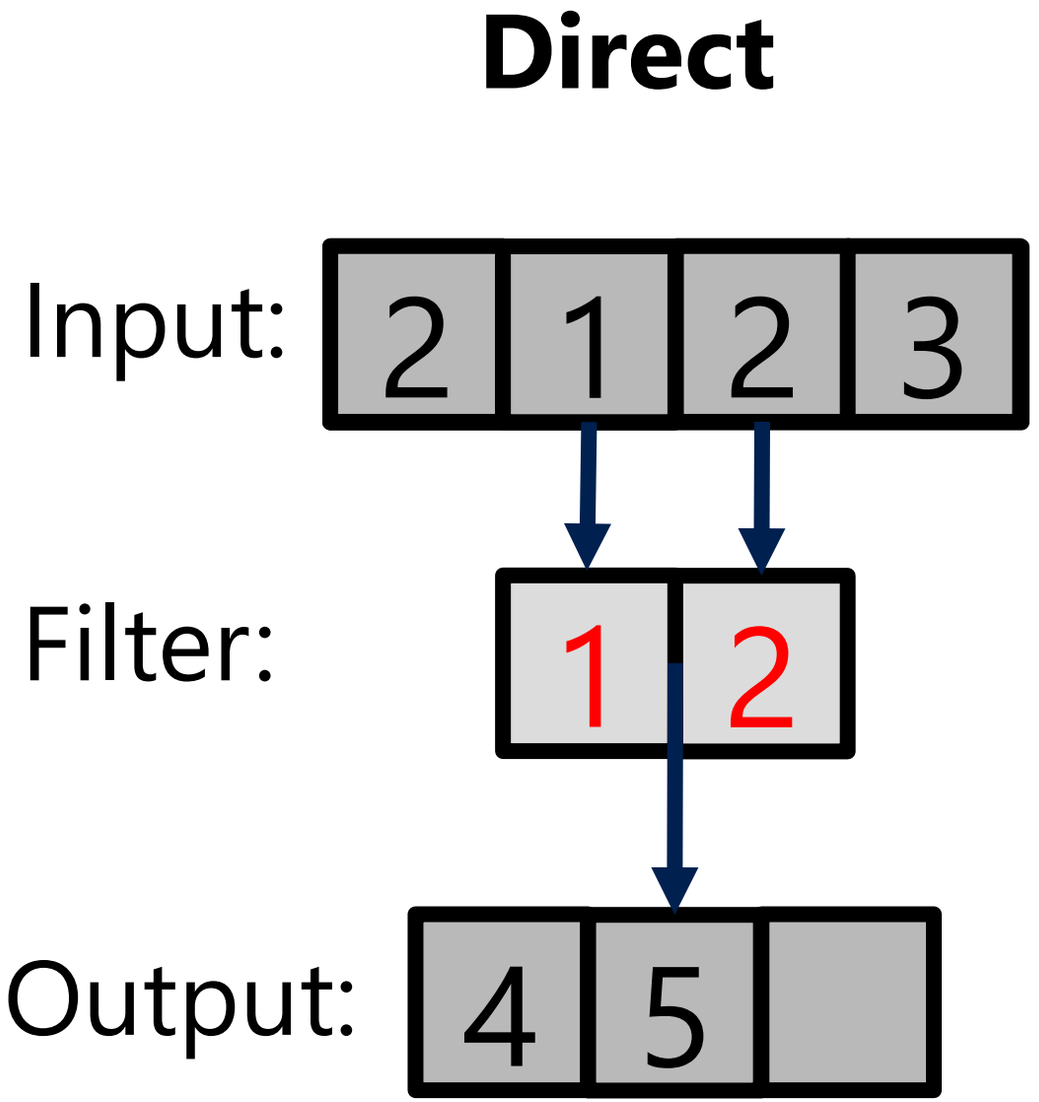
Direct



Convolutions in Deep Neural Networks

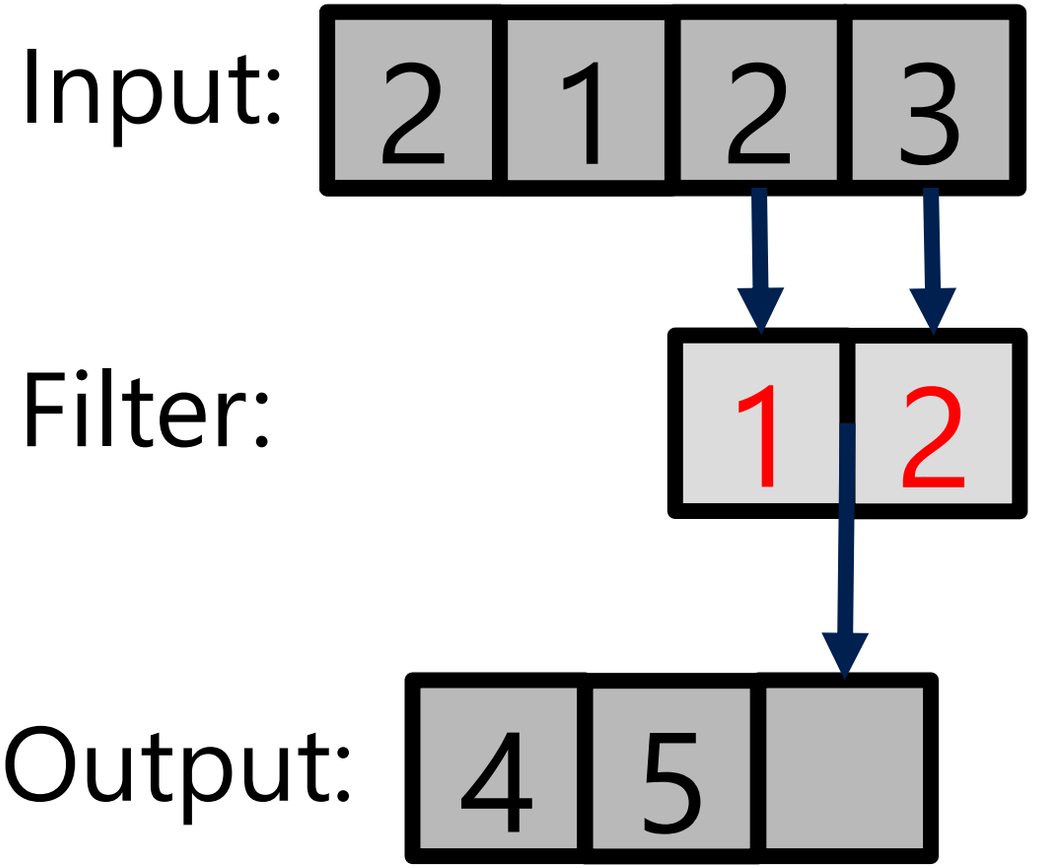


Convolutions in Deep Neural Networks



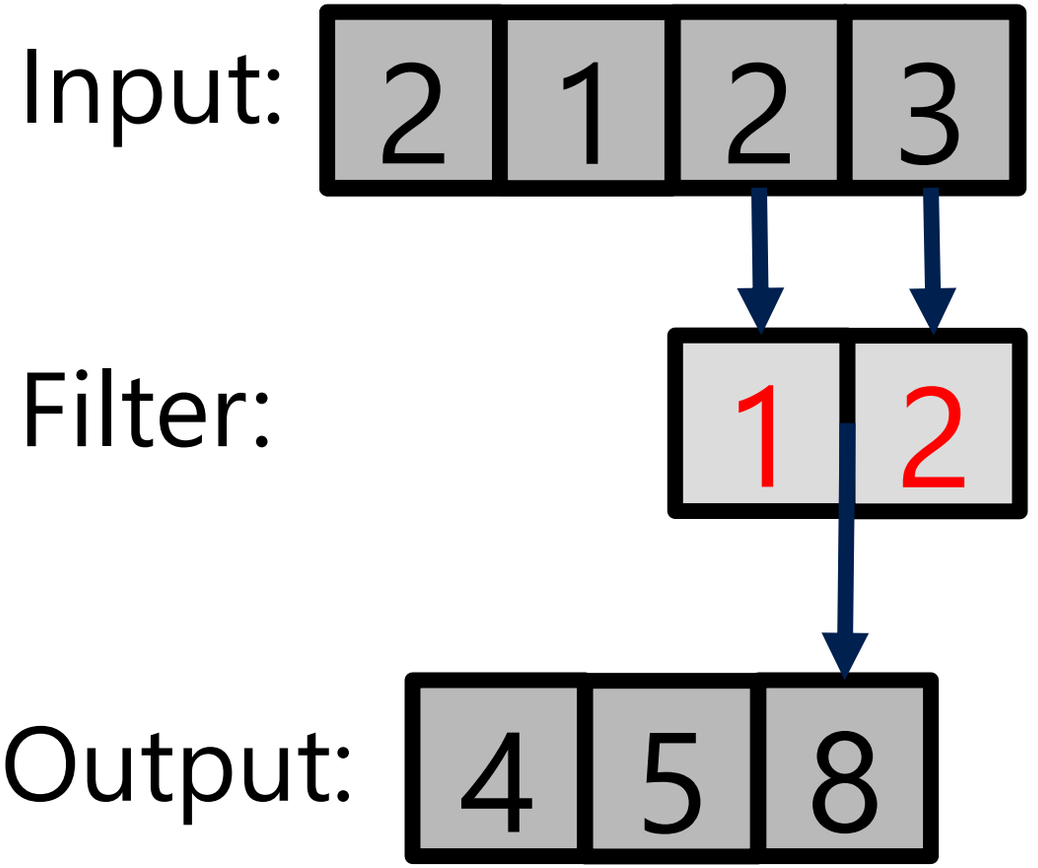
Convolutions in Deep Neural Networks

Direct



Convolutions in Deep Neural Networks

Direct

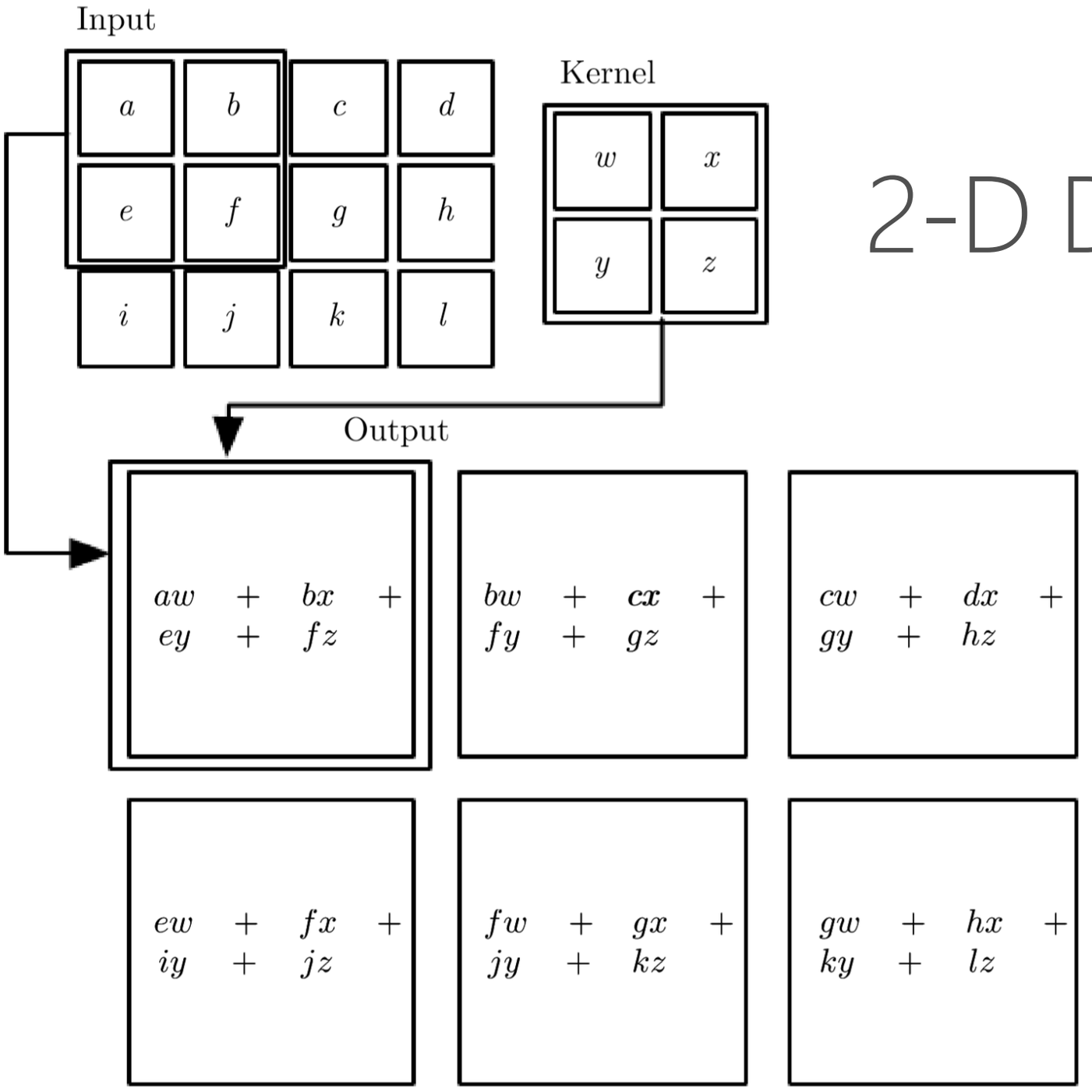


Convolution formally

1D convolution

$$O_x = \sum_{u=1}^{s} I_{x+u} F_u$$

2-D Direct Convolution



Deep Learning Book
https://www.deeplearningbook.org/

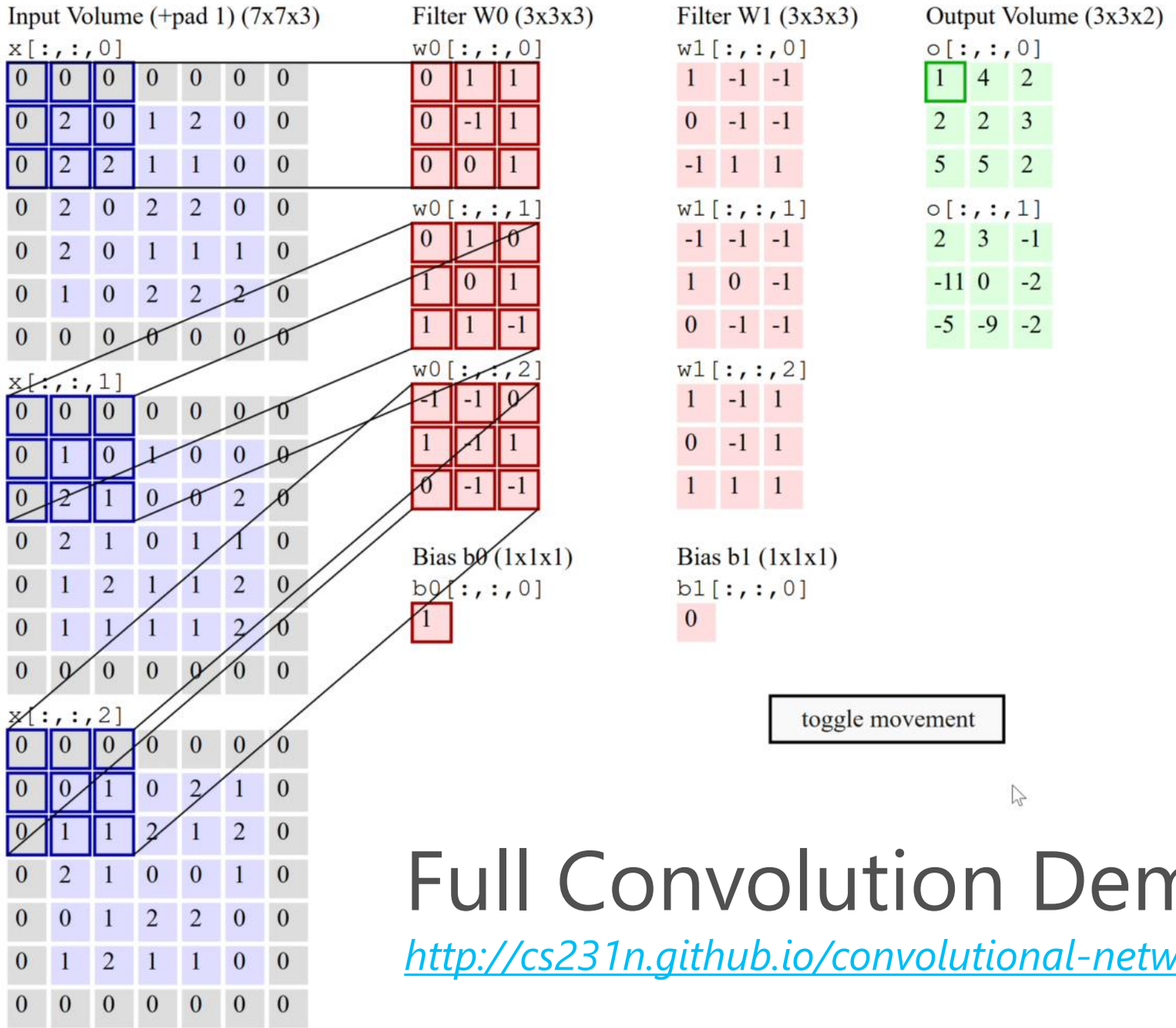
Convolution formally

1D convolution

$$O_x = \sum_{u=1}^{S} I_{x+u} F_u$$

2D convolution

$$O_{x,y} = \sum_{v=1}^{R} \sum_{u=1}^{S} I_{x+u,y+v} F_{u,v}$$



Full Convolution Demo

http://cs231n.github.io/convolutional-networks/#conv

Convolution formally

1D convolution

$$O_x = \sum_{u=1}^{S} I_{x+u} F_u$$

2D convolution

$$O_{x,y} = \sum_{v=1}^{R} \sum_{u=1}^{S} I_{x+u,y+v} F_{u,v}$$

Conv Layer

$$O_{i,k,x,y} = \sum_{c=1}^{C} \sum_{v=1}^{R} \sum_{u=1}^{S} I_{i,c,x+u,y+v} F_{k,c,u,v}$$

Compare Convolution Algorithms

	Filter=k	*Filter=n/2*
	Input=n	*Input=n*
	Small filter	**Big filter**

Compare Convolution Algorithms

	Filter=k *Input=n* **Small filter**	*Filter=n/2* *Input=n* **Big filter**
Direct	$O(nk)$	$O(n^2)$

Compare Convolution Algorithms

	Filter=k *Input=n* **Small filter**	*Filter=n/2* *Input=n* **Big filter**
Direct	$O(nk)$	$O(n^2)$
Winograd	$O(n)$*	$O(n^2)$*

Compare Convolution Algorithms

	Filter=k *Input=n* **Small filter**	*Filter=n/2* *Input=n* **Big filter**
Direct	$O(nk)$	$O(n^2)$
Winograd	$O(n)$*	$O(n^2)$*
FFT	$O(n \log n)$	$O(n \log n)$

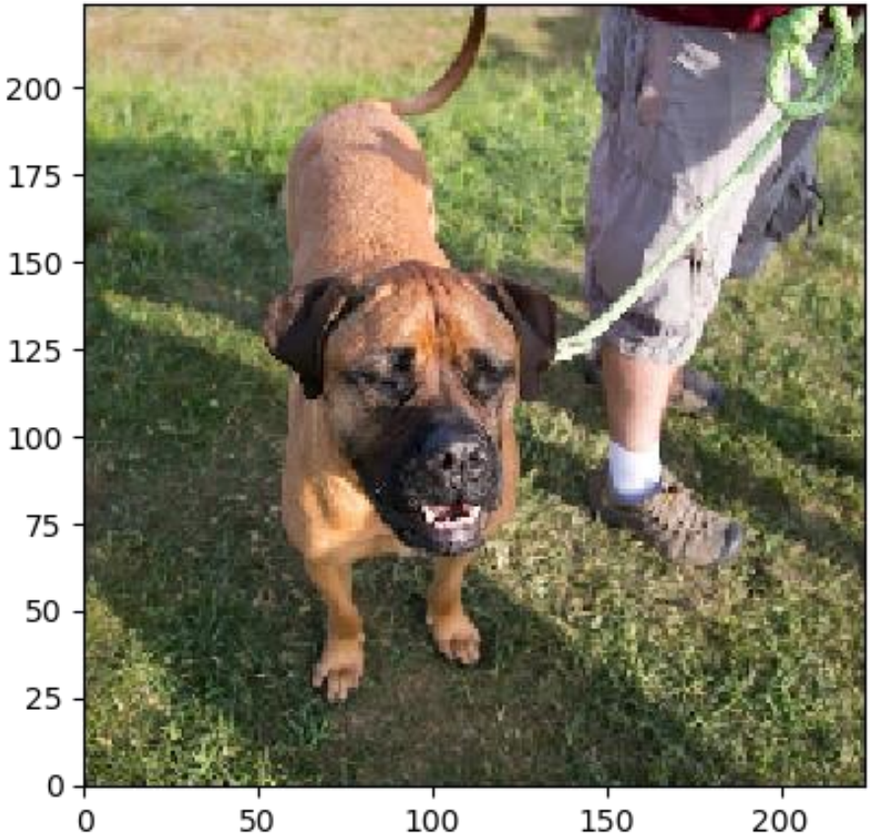
Compare Convolution Algorithms

	Filter=k *Input=n* **Small filter**	*Filter=n/2* *Input=n* **Big filter**
Direct	$O(nk)$	$O(n^2)$
Winograd	$O(n)$*	$O(n^2)$*
FFT	$O(n \log n)$	$O(n \log n)$

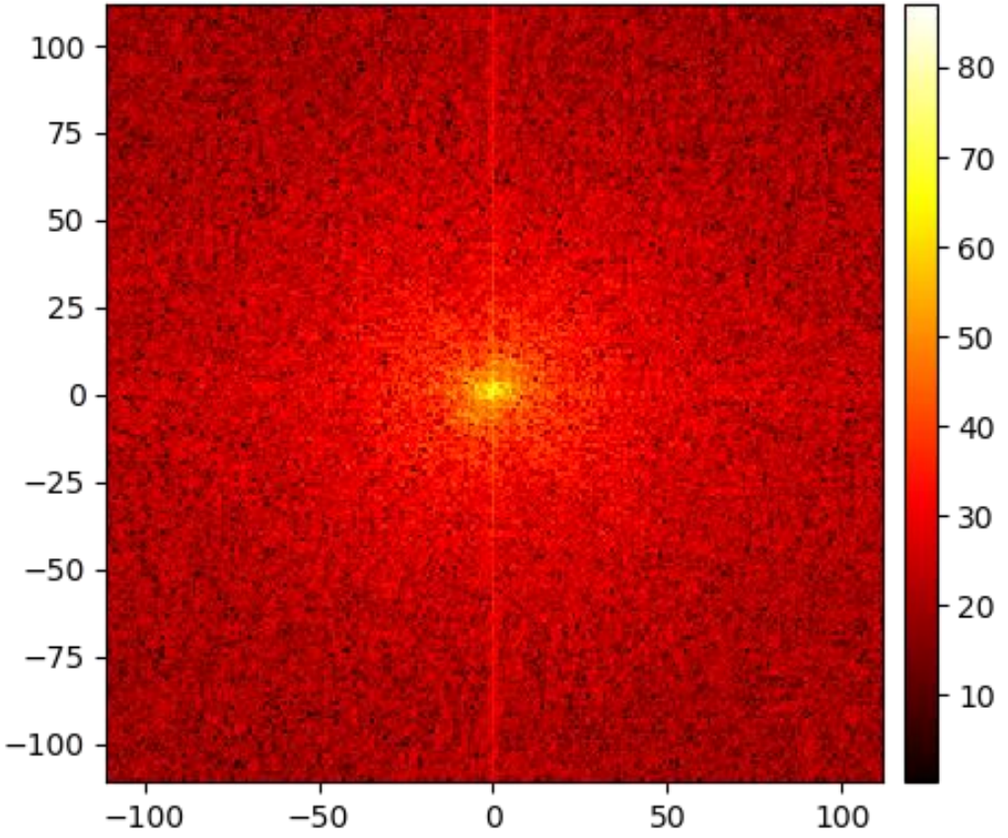
FFT-based convolution

Natural images
More information put in lower frequencies

Original image



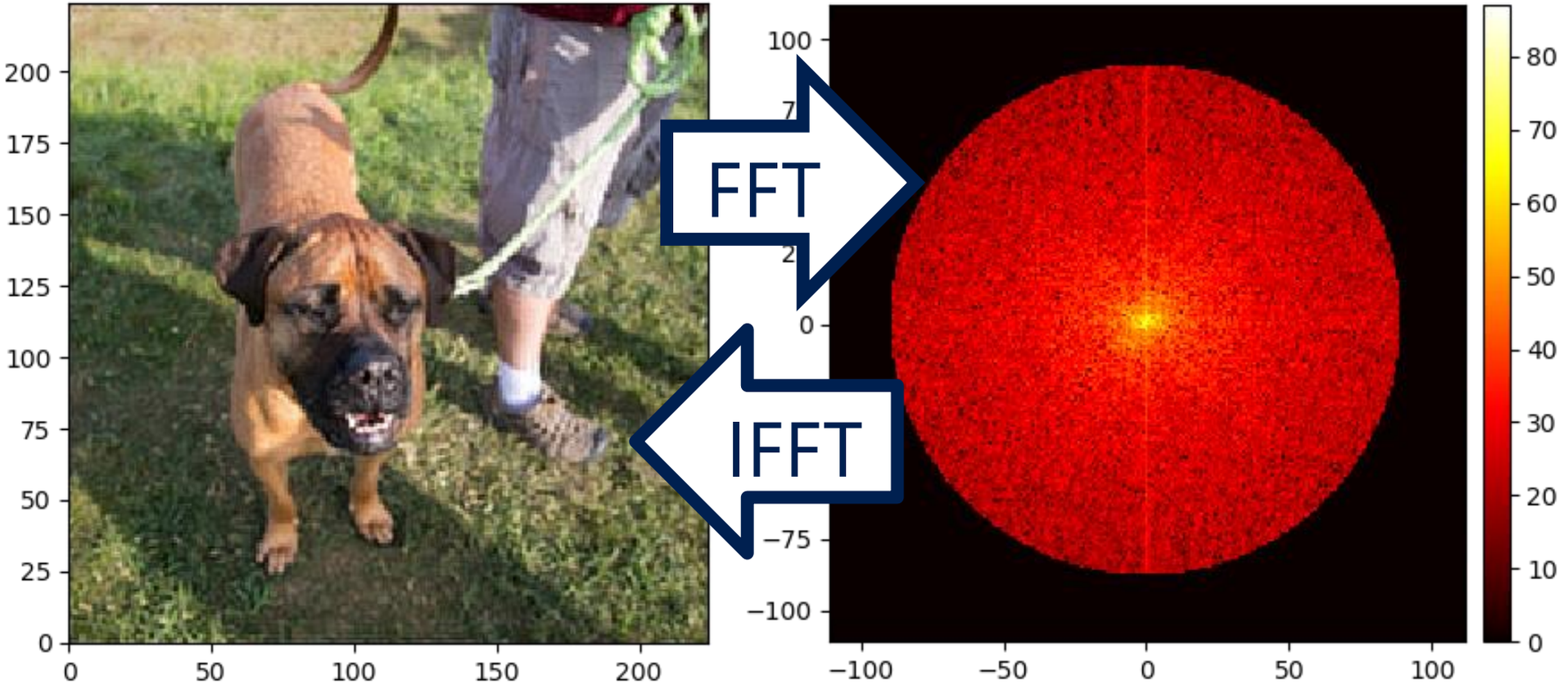
Spatial domain



Frequency domain

Natural images
Transformations between the domains

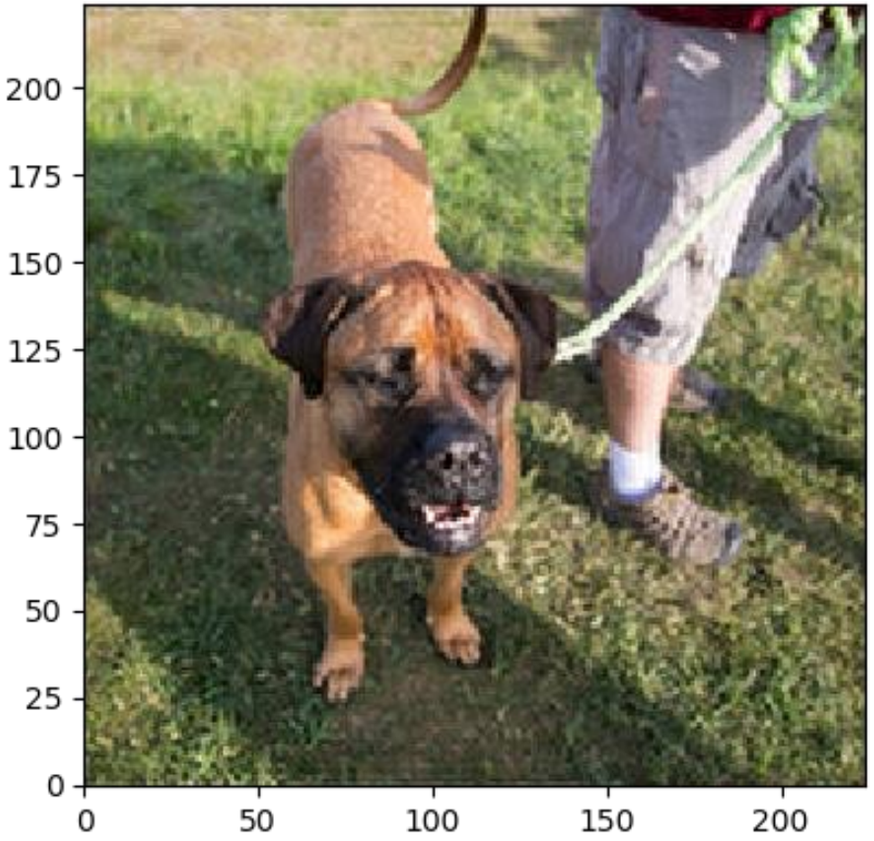
Compression 50%



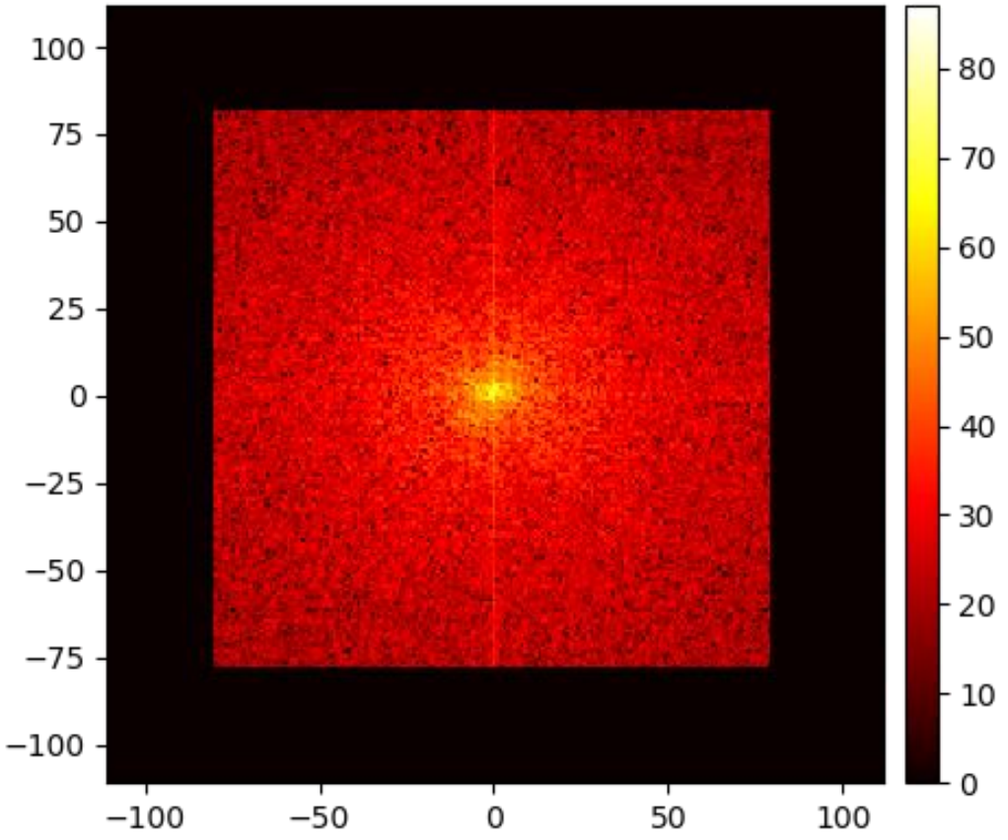
Spatial domain **Frequency domain**

Method for ConvNets to constrain the frequency band in convolution operation for efficiency

Compression 50% in practice



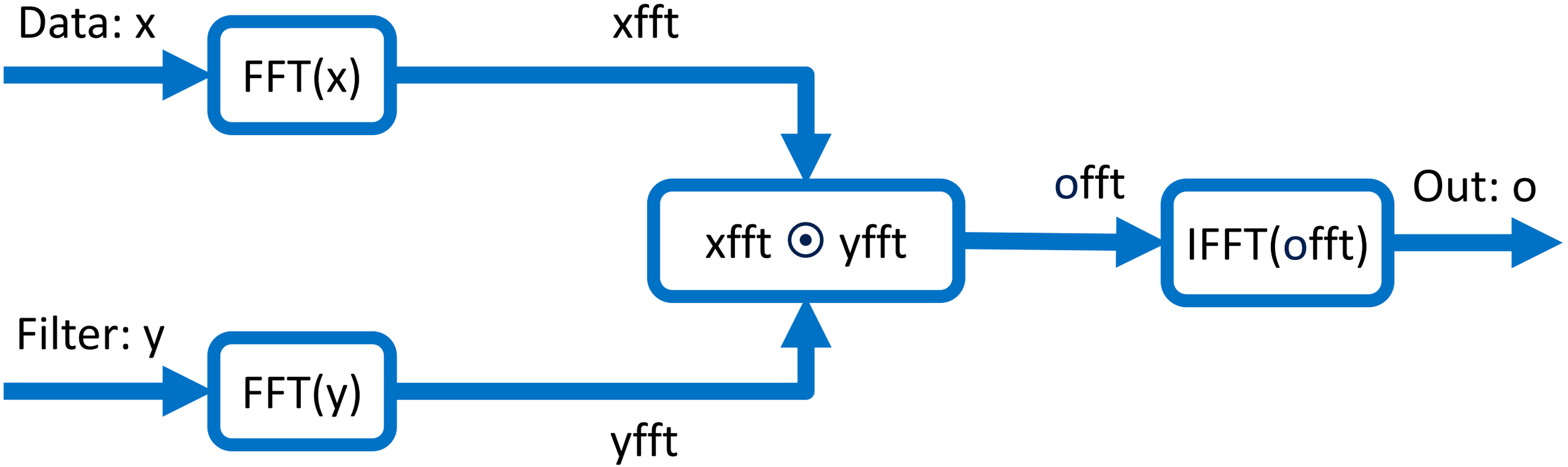
Spatial domain



Frequency domain

FFT based convolution

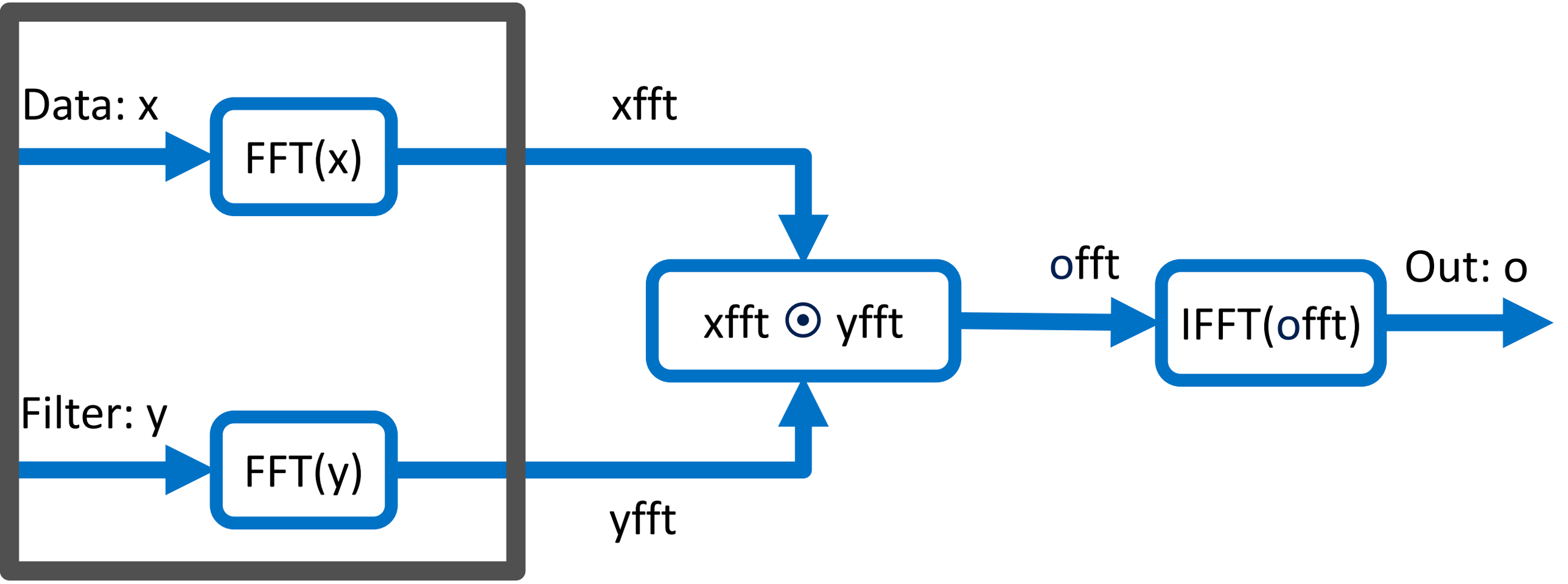
Mathieu et al.: "Fast Training of Convolutional Networks through FFTs"
Vasilache et al.: "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation"



FFT based convolution

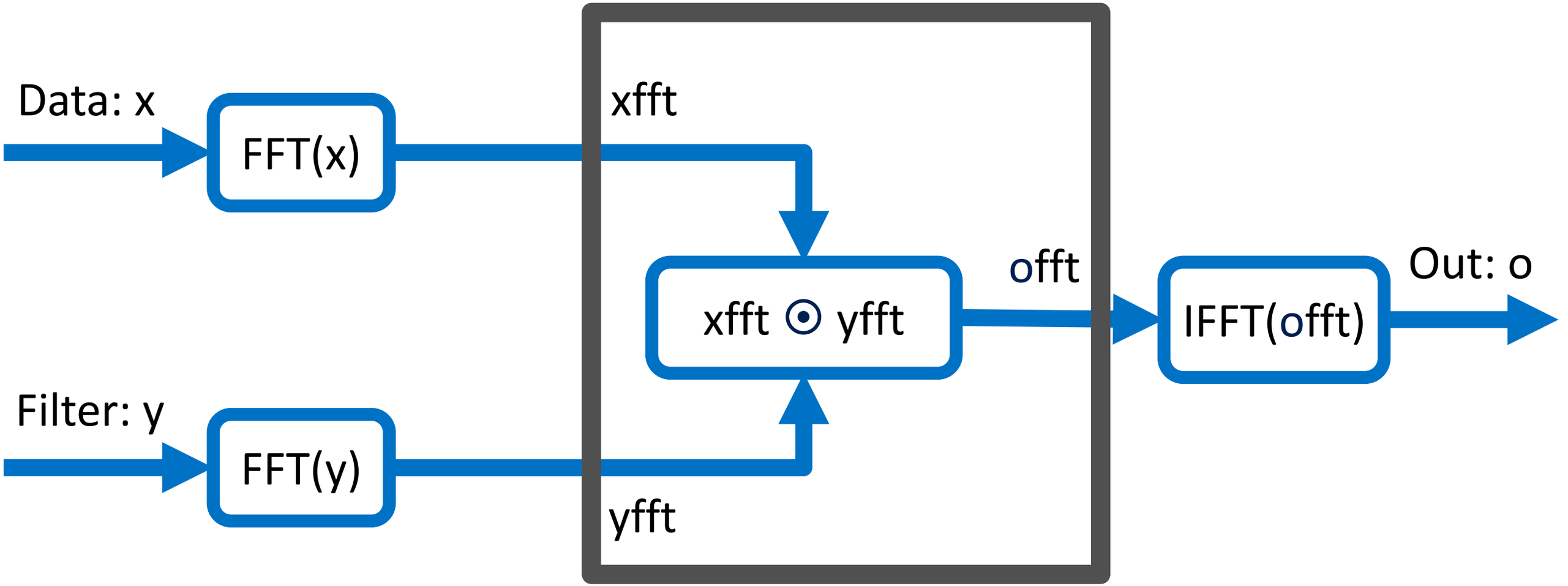
Mathieu et al.: "Fast Training of Convolutional Networks through FFTs"

Vasilache et al.: "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation"



FFT based convolution

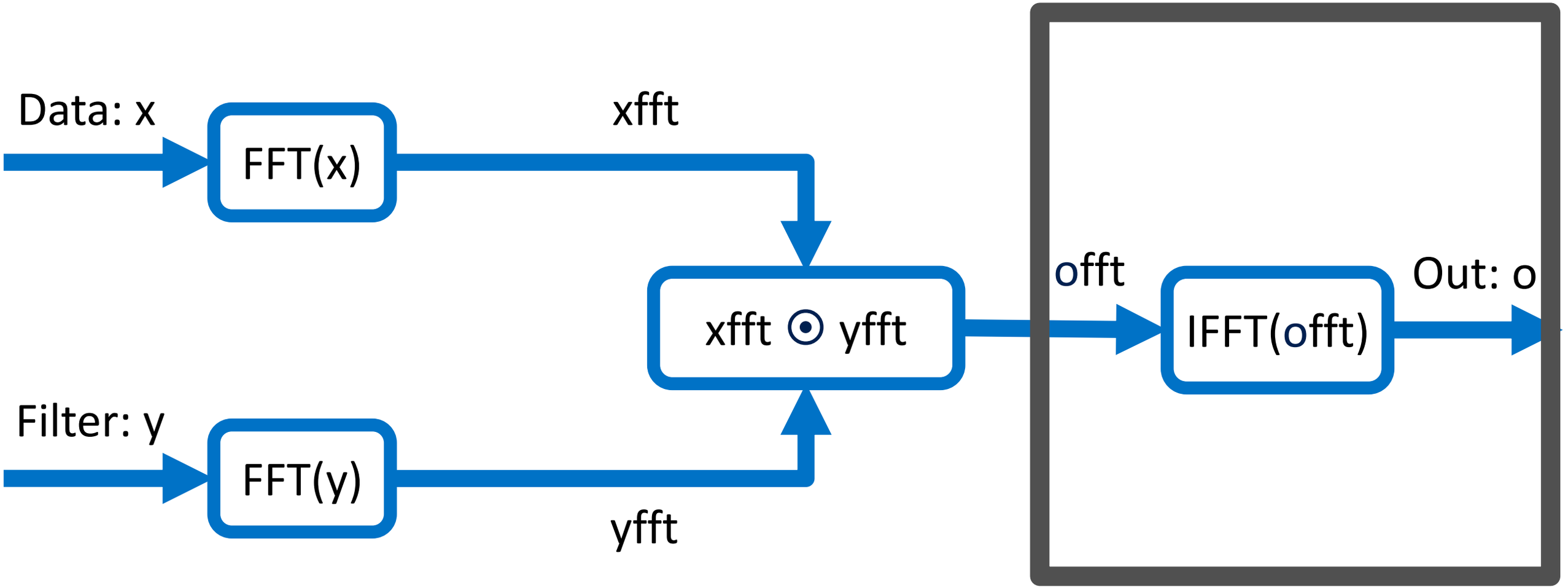
Mathieu et al.: "Fast Training of Convolutional Networks through FFTs"
Vasilache et al.: "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation"



FFT based convolution

Mathieu et al.: "Fast Training of Convolutional Networks through FFTs"

Vasilache et al.: "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation"

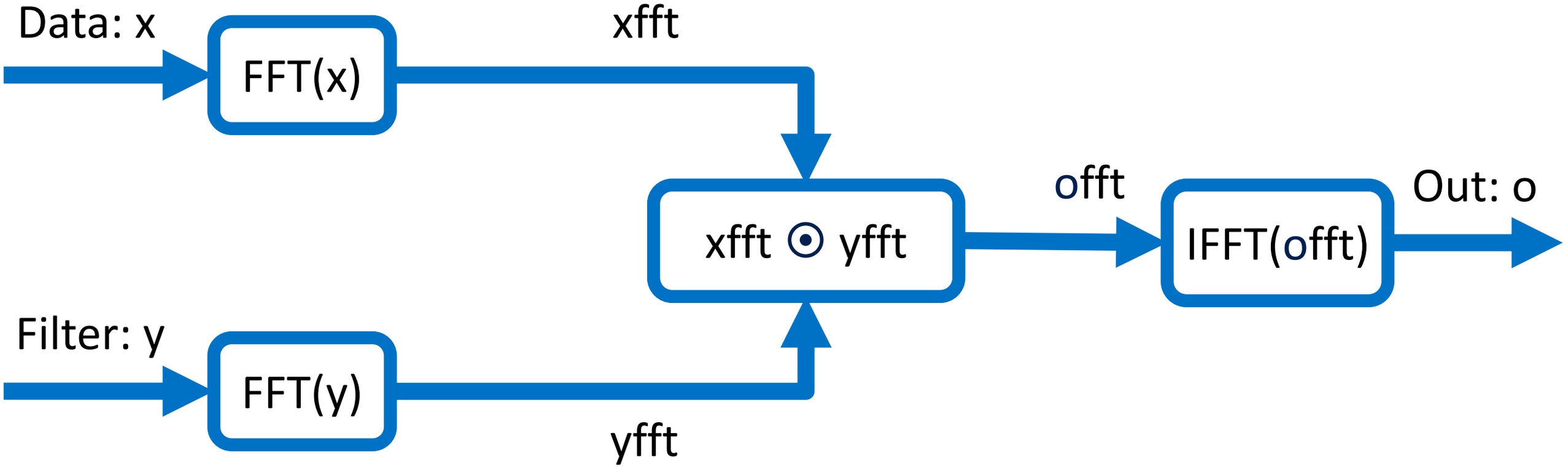


FFT based convolution

Mathieu et al.: "Fast Training of Convolutional Networks through FFTs"

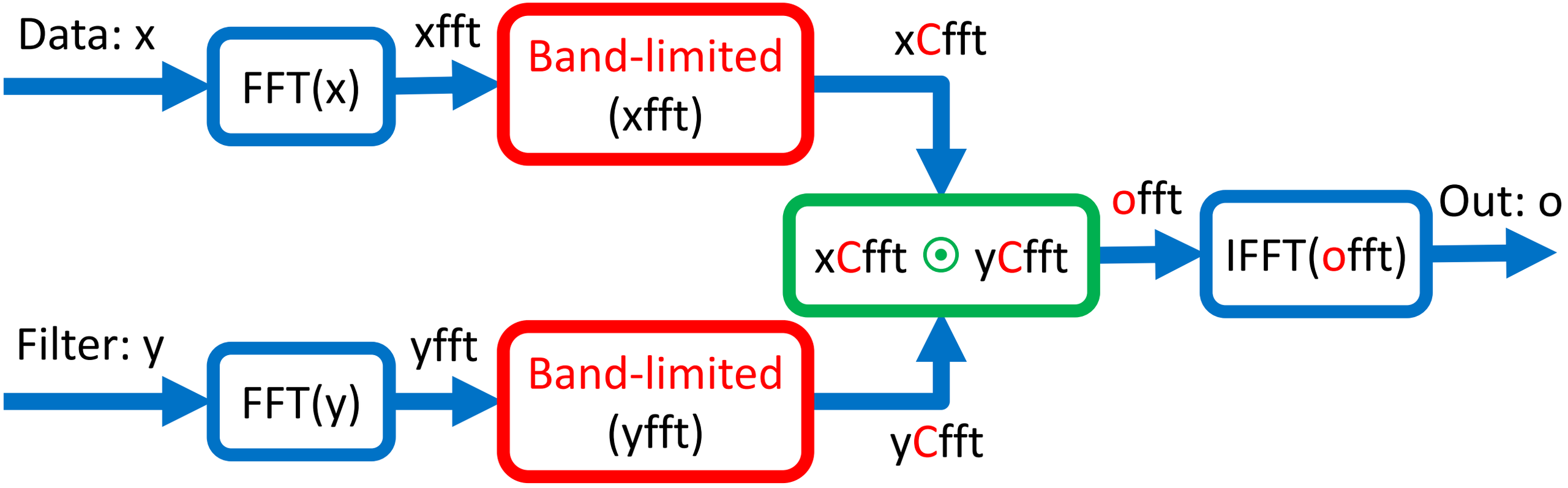
Vasilache et al.: "Fast Convolutional Nets With fbfft: A GPU Performance Evaluation"

cuDNN: Substantial memory workspace needed for intermediate results.

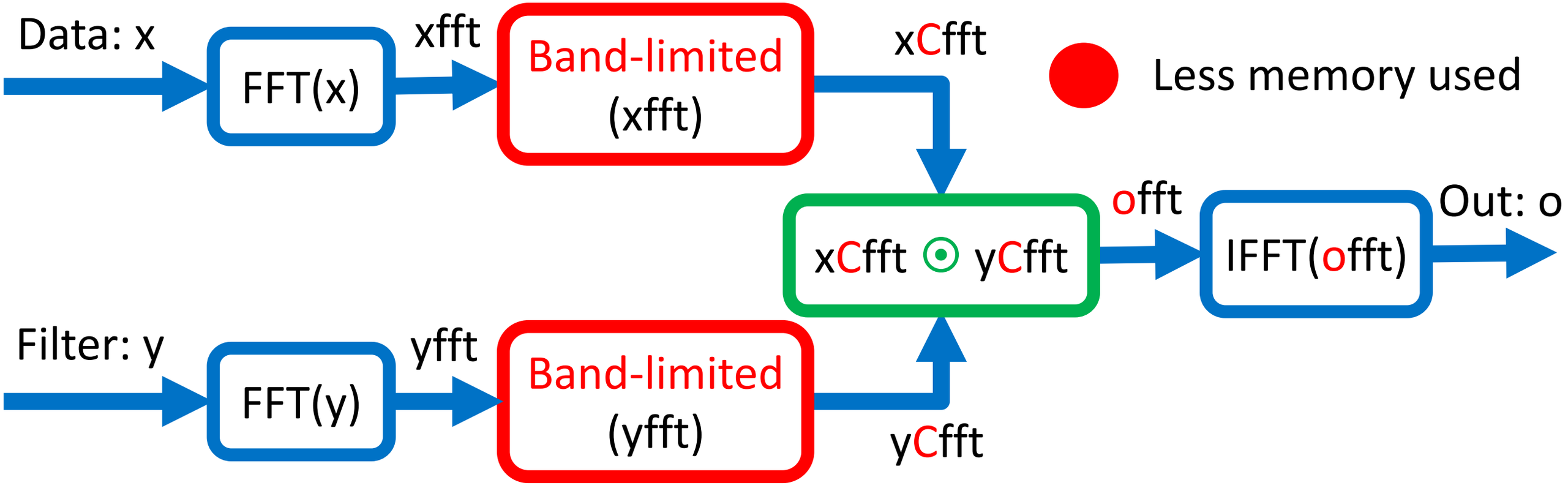


Band-limited FFT based convolution

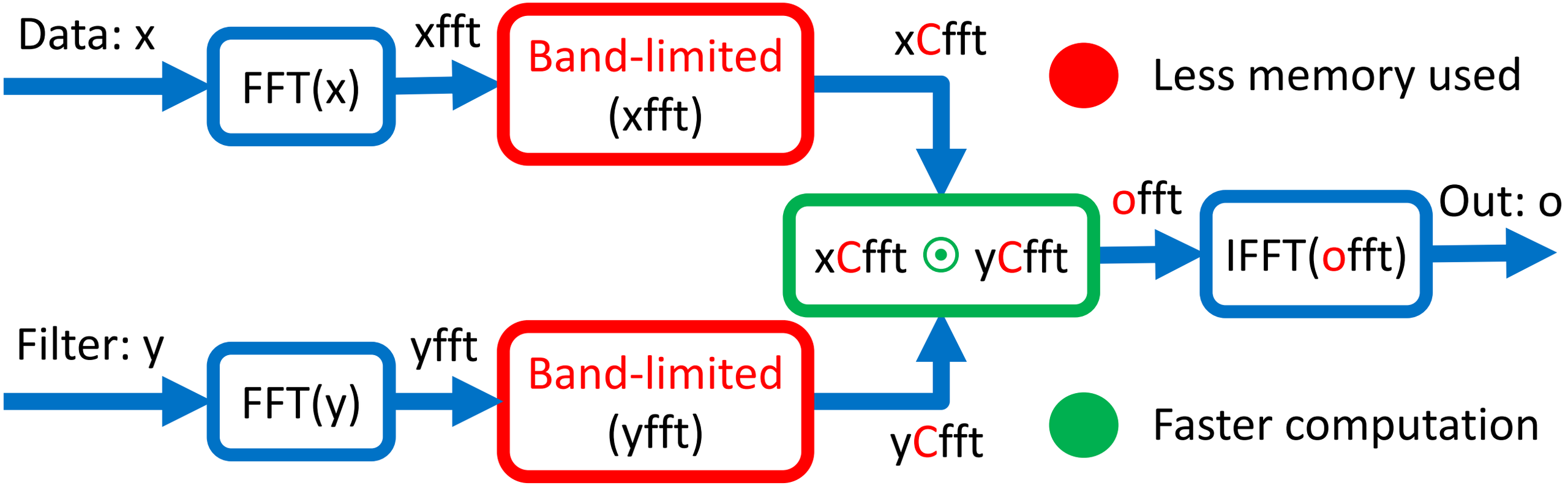
Band-limiting = masking out high frequencies



Band-limited FFT based convolution

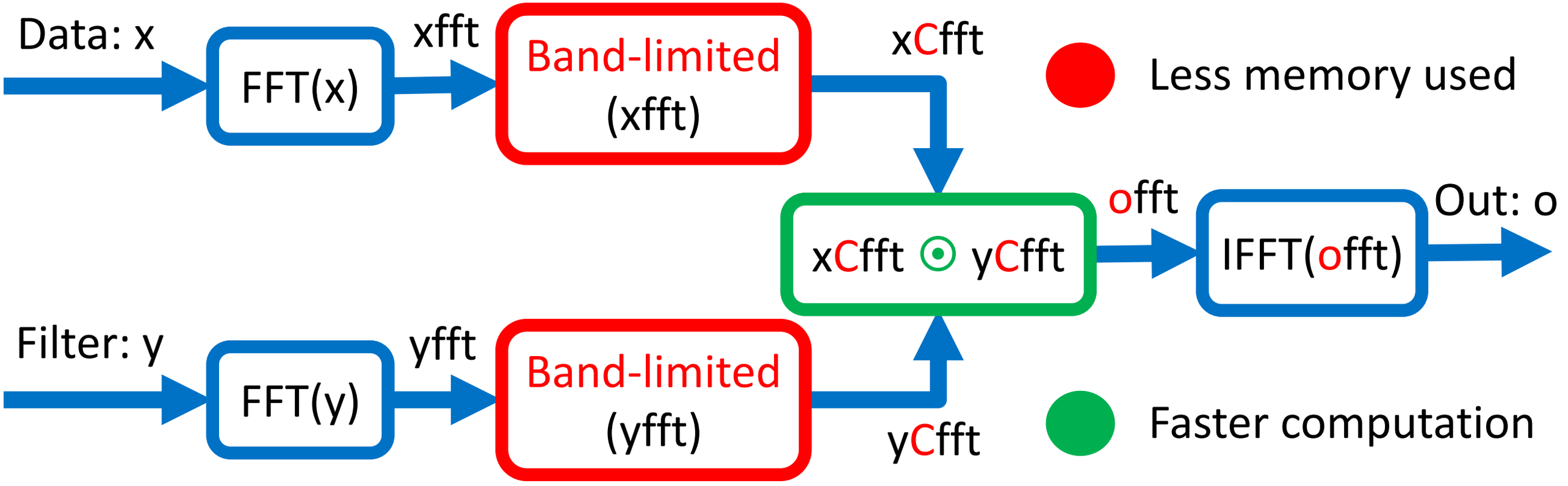


Band-limited FFT based convolution

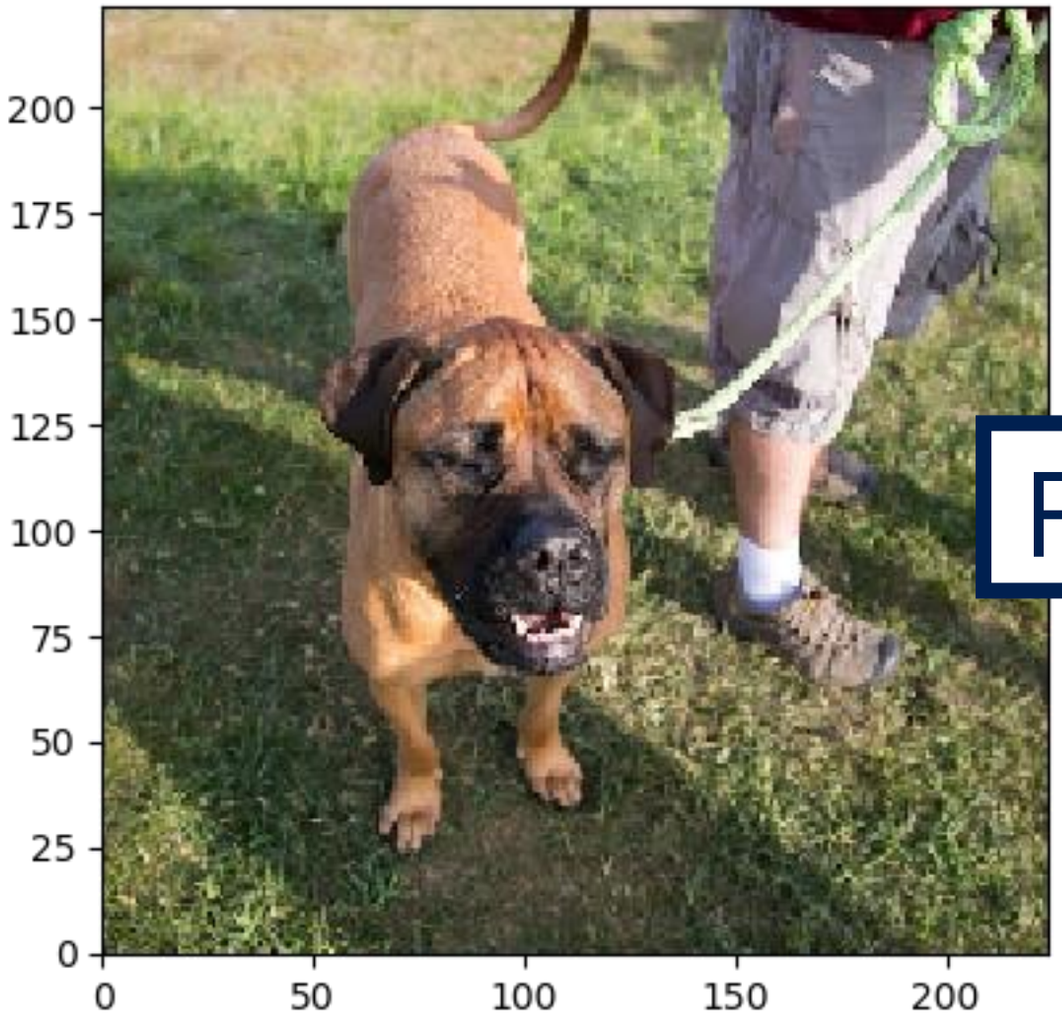


Band-limited FFT based convolution

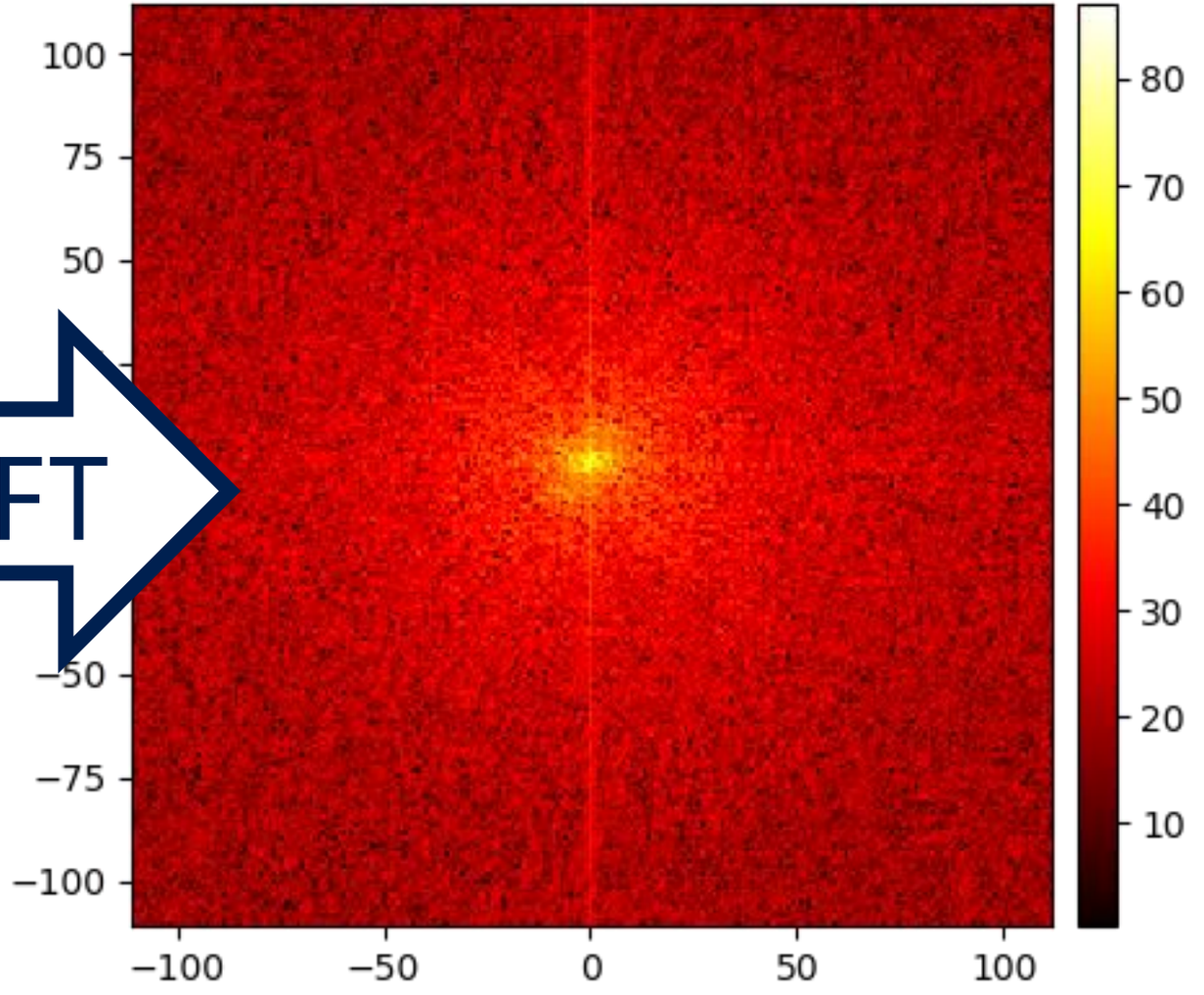
Preserve enough of the spectrum to retain high accuracy of models.



Compression Technique in Fourier Domain



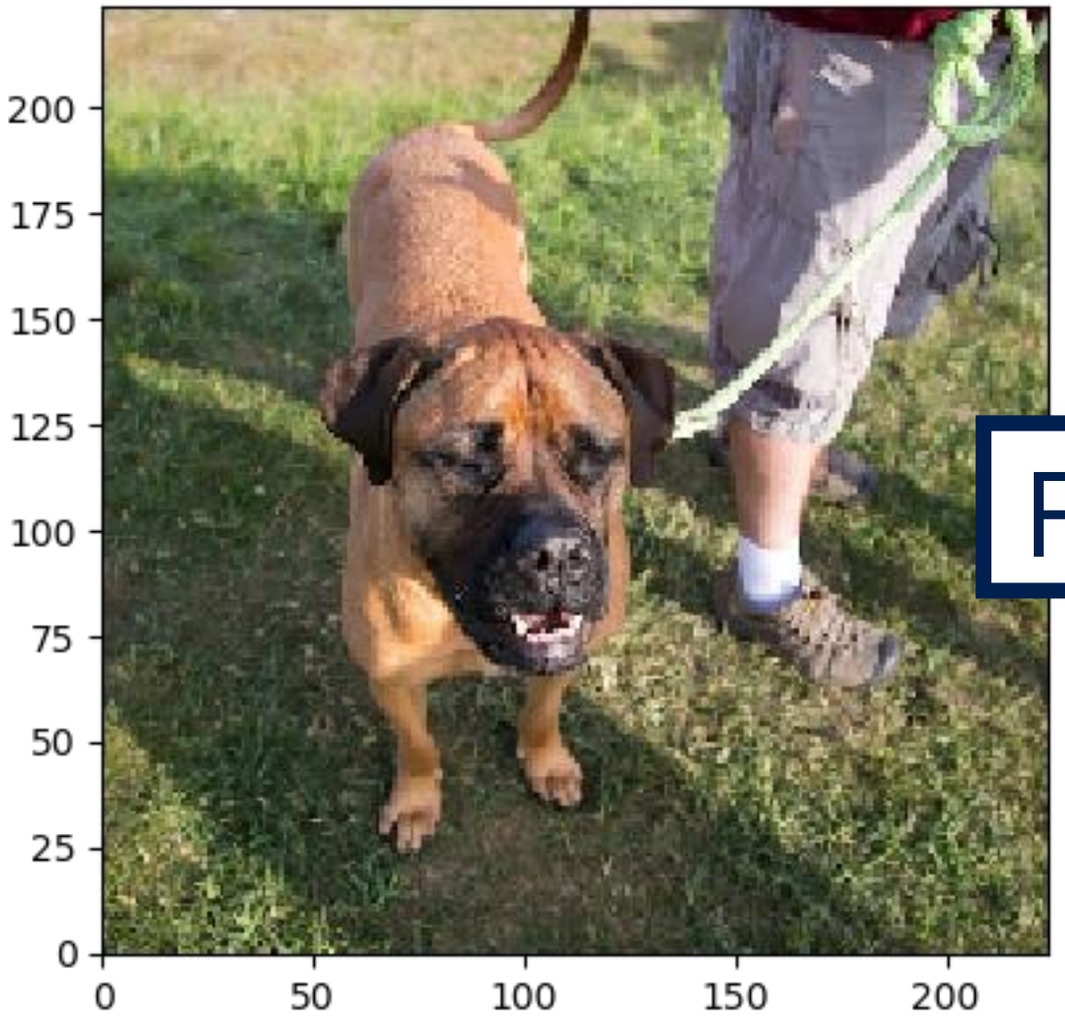
FFT



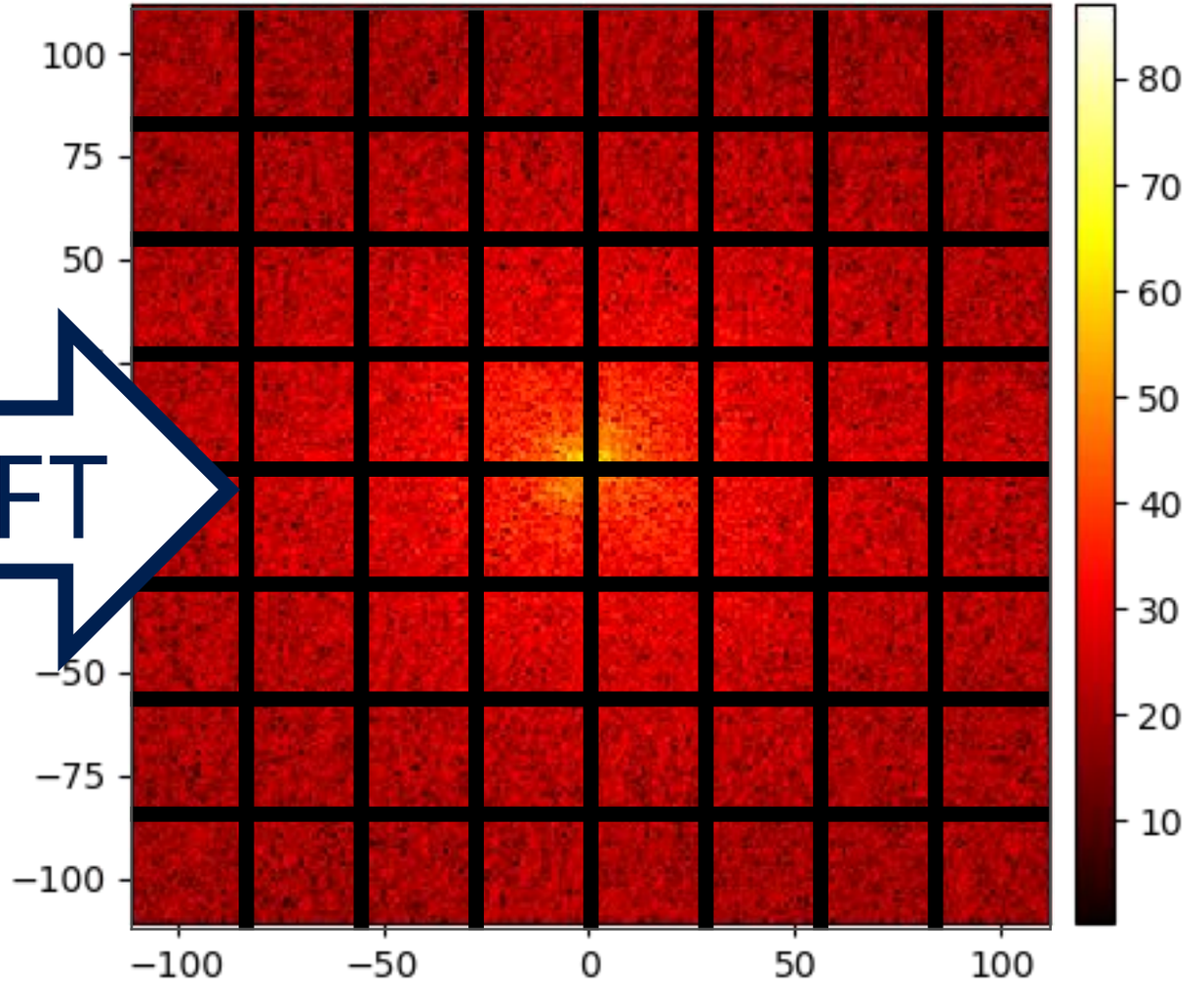
Spatial domain

Frequency domain

Compression Technique in Fourier Domain



FFT

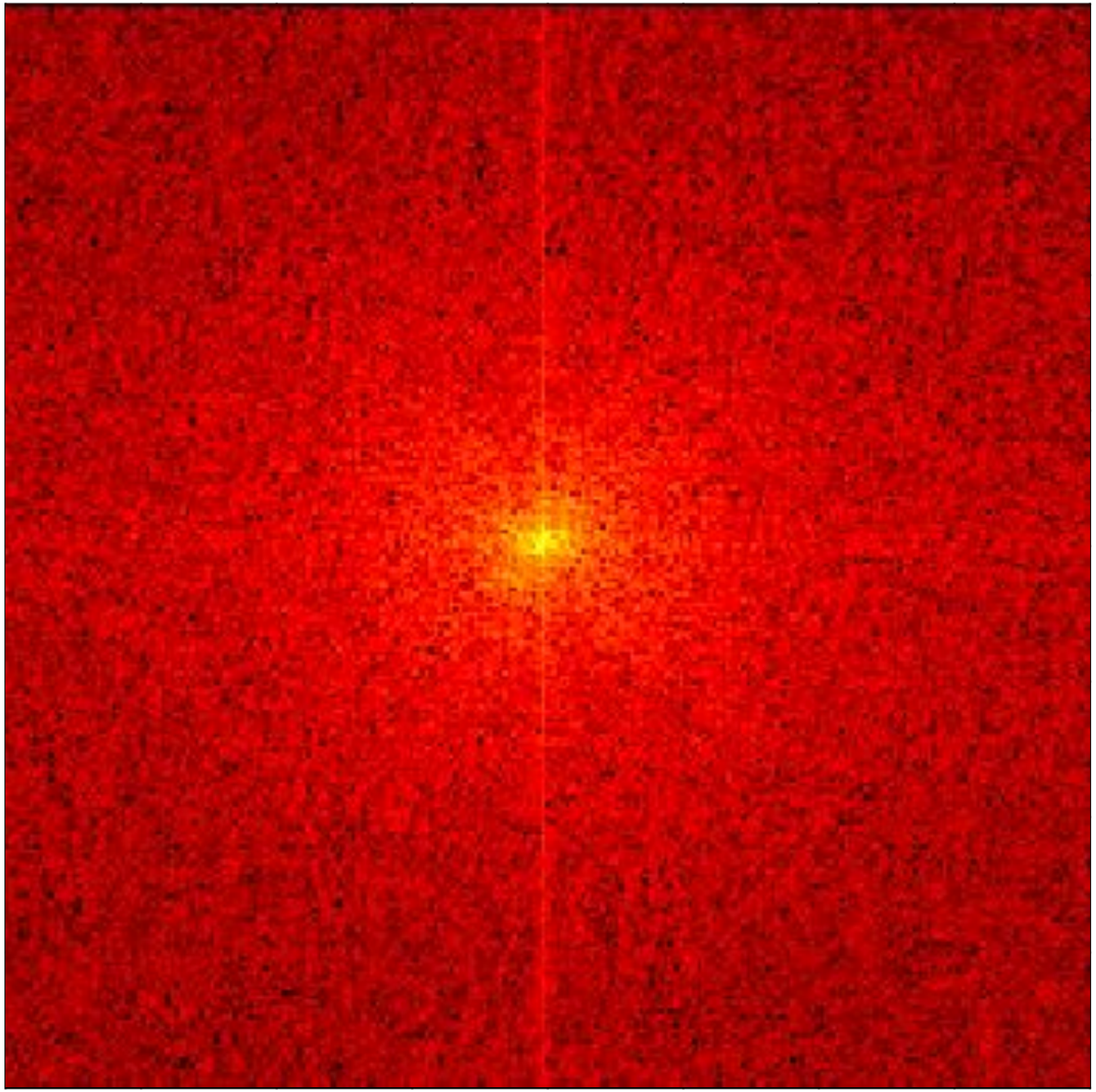


Spatial domain

Frequency domain

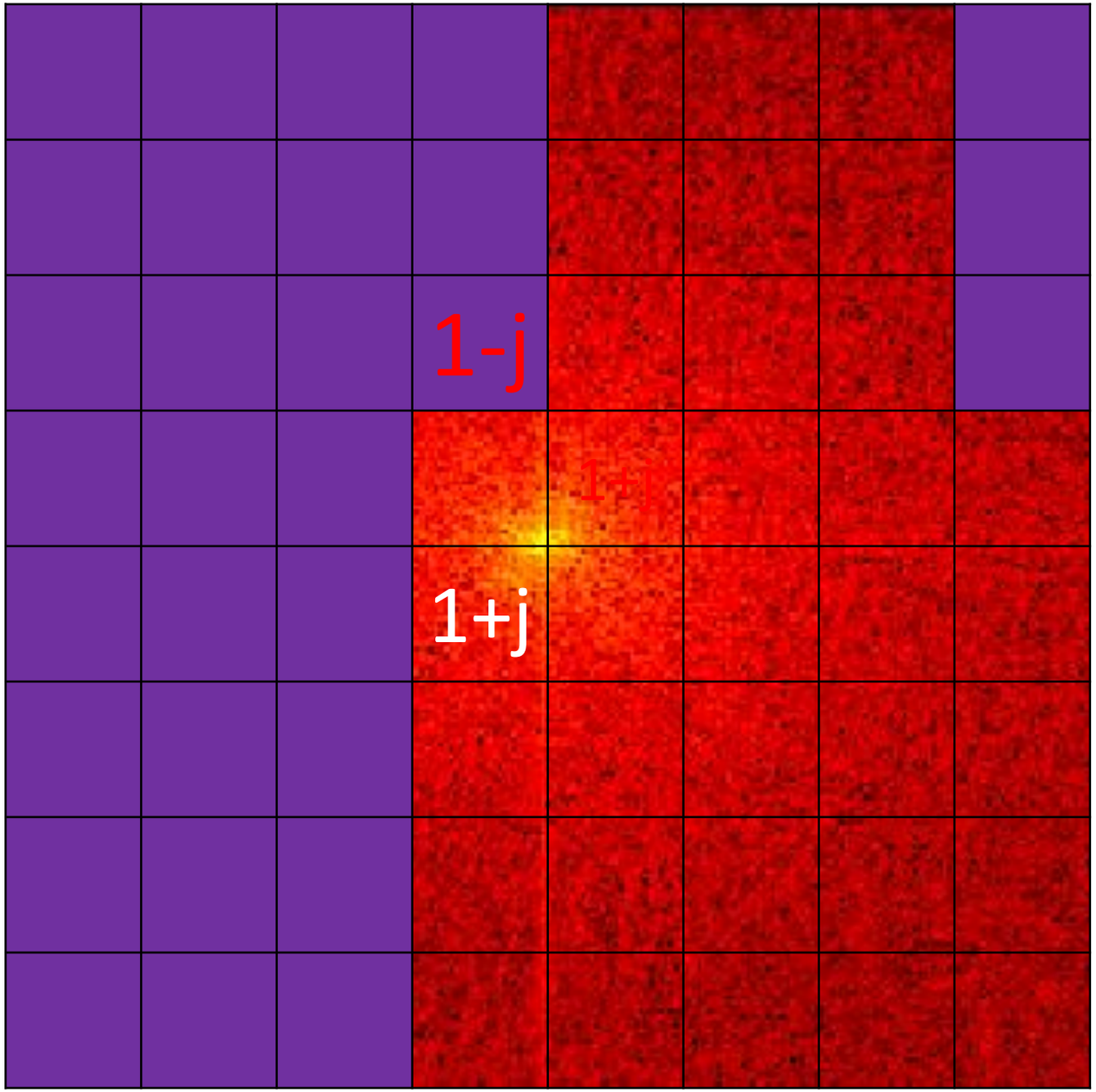
Band-limiting Technique

1. FFT of an input data



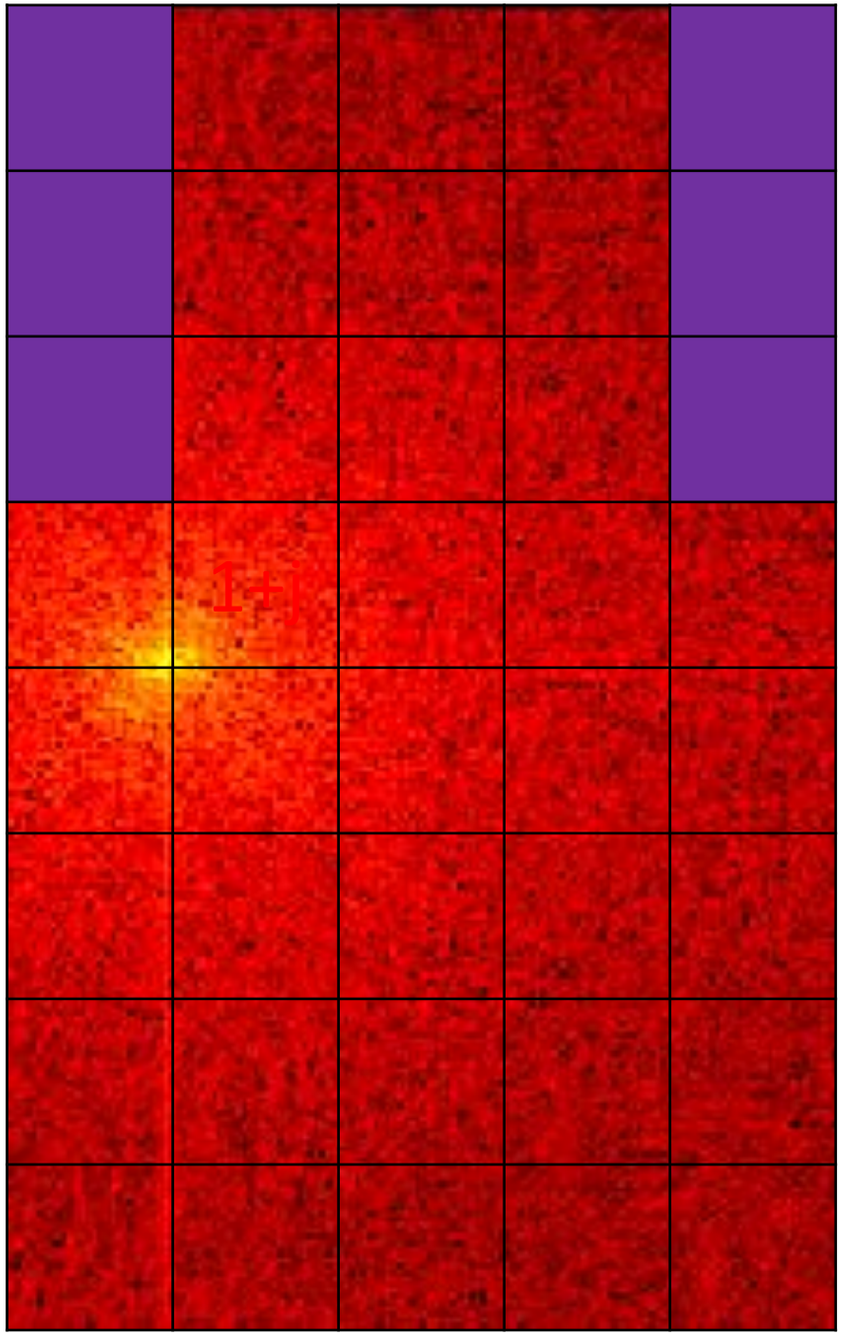
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry



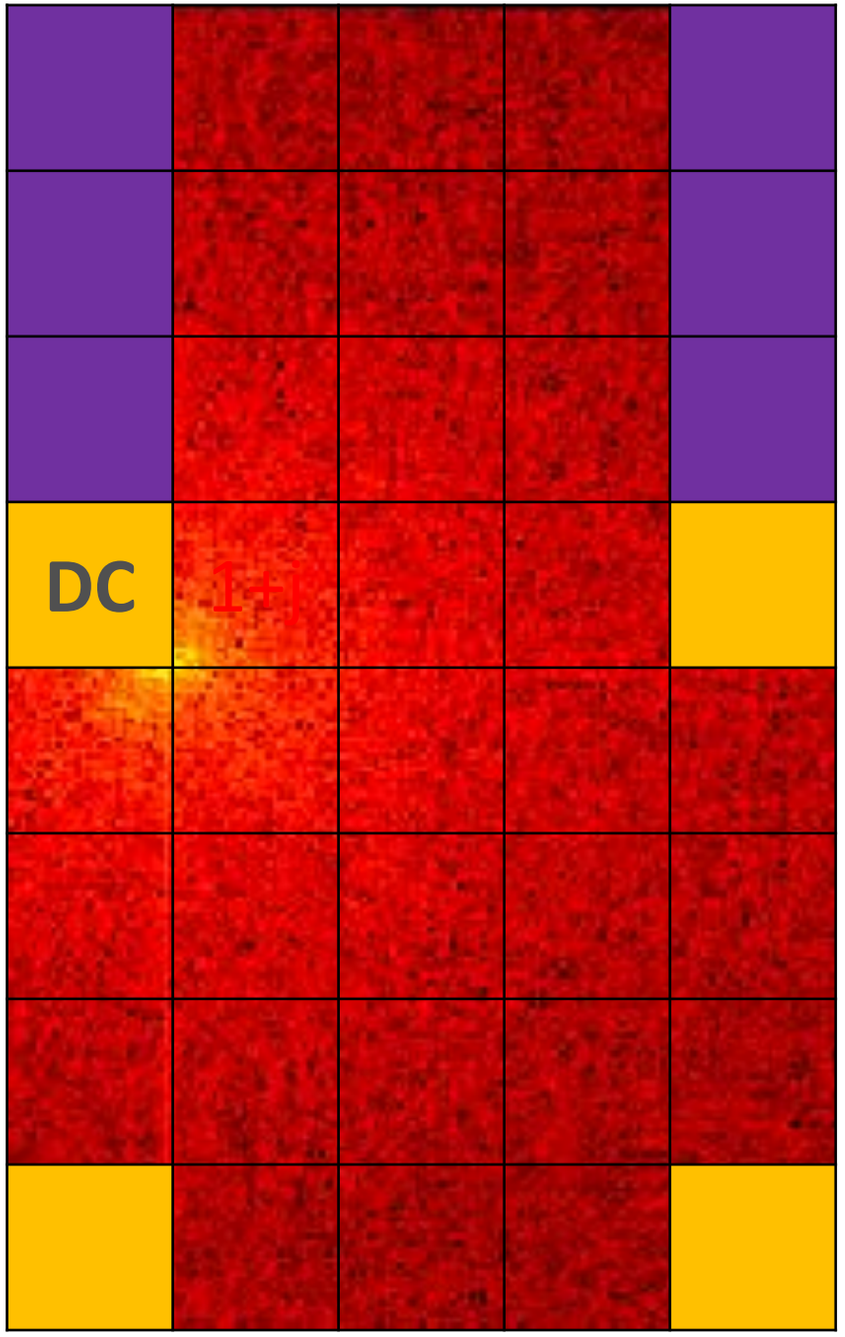
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry



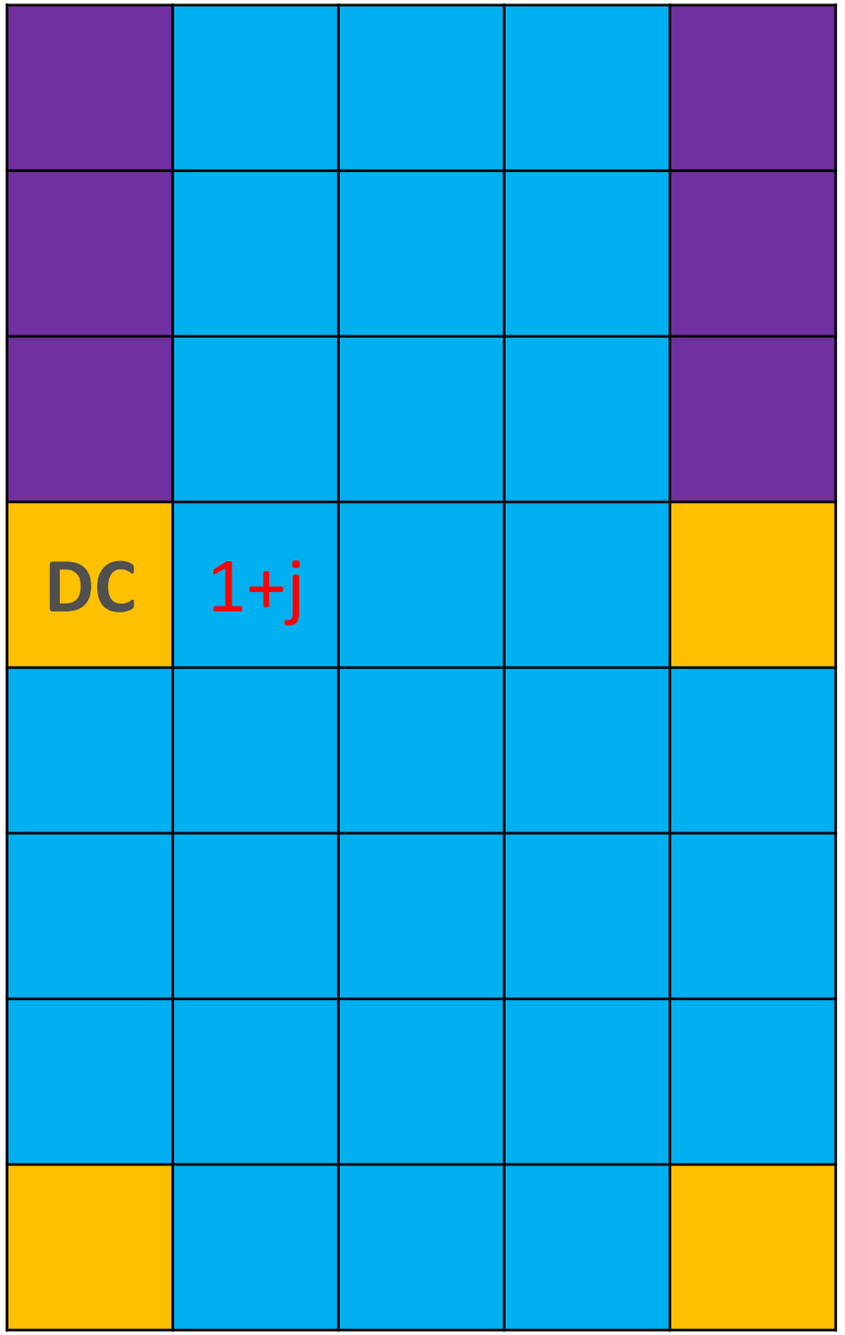
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values



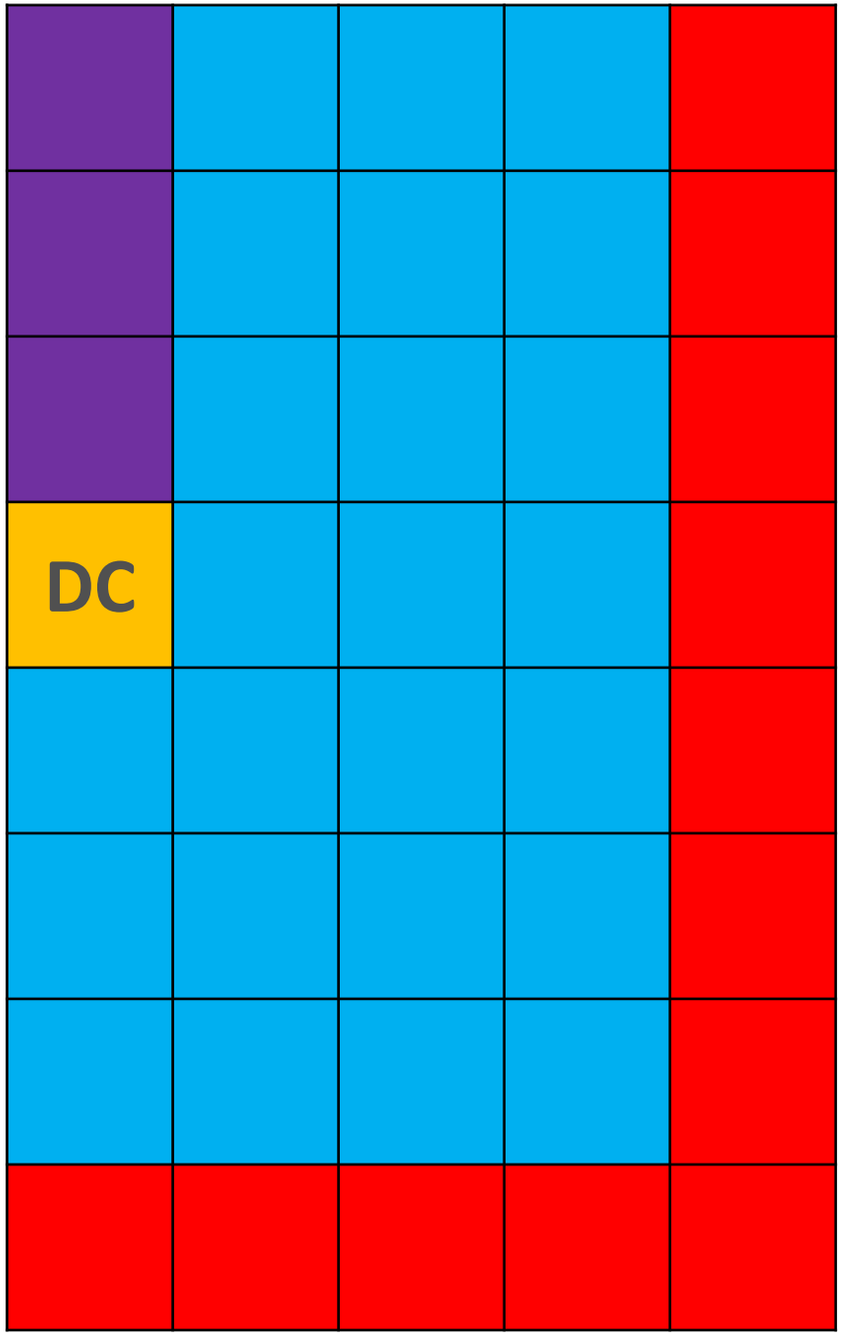
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints



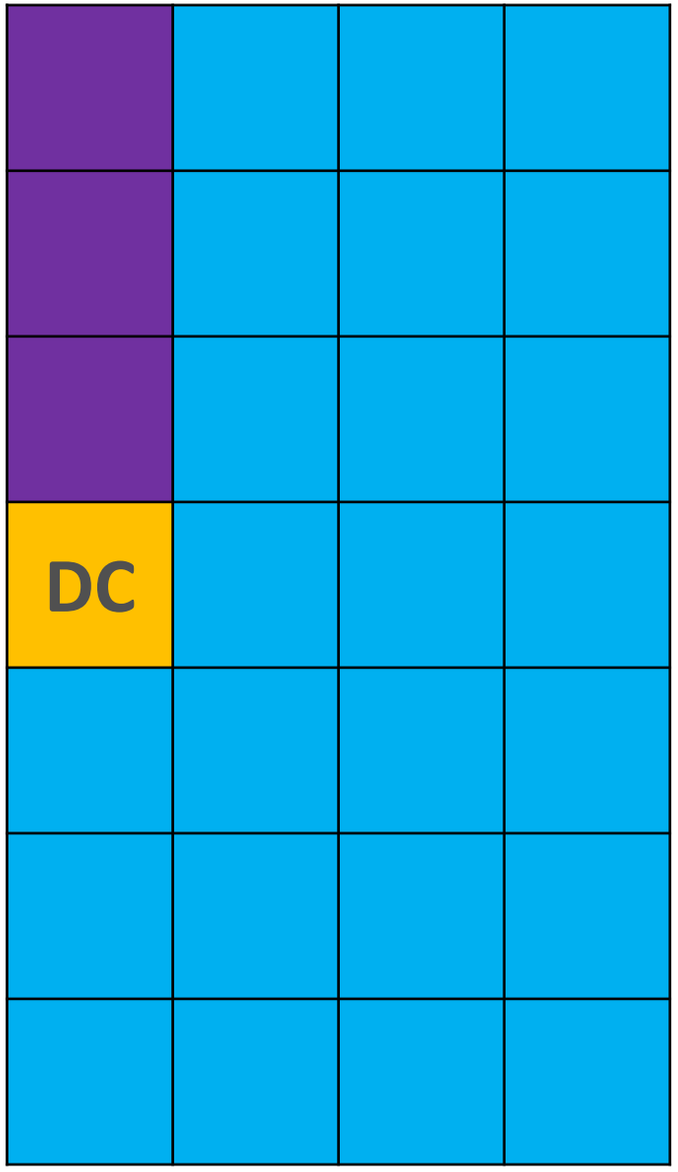
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints
5. 1st compression



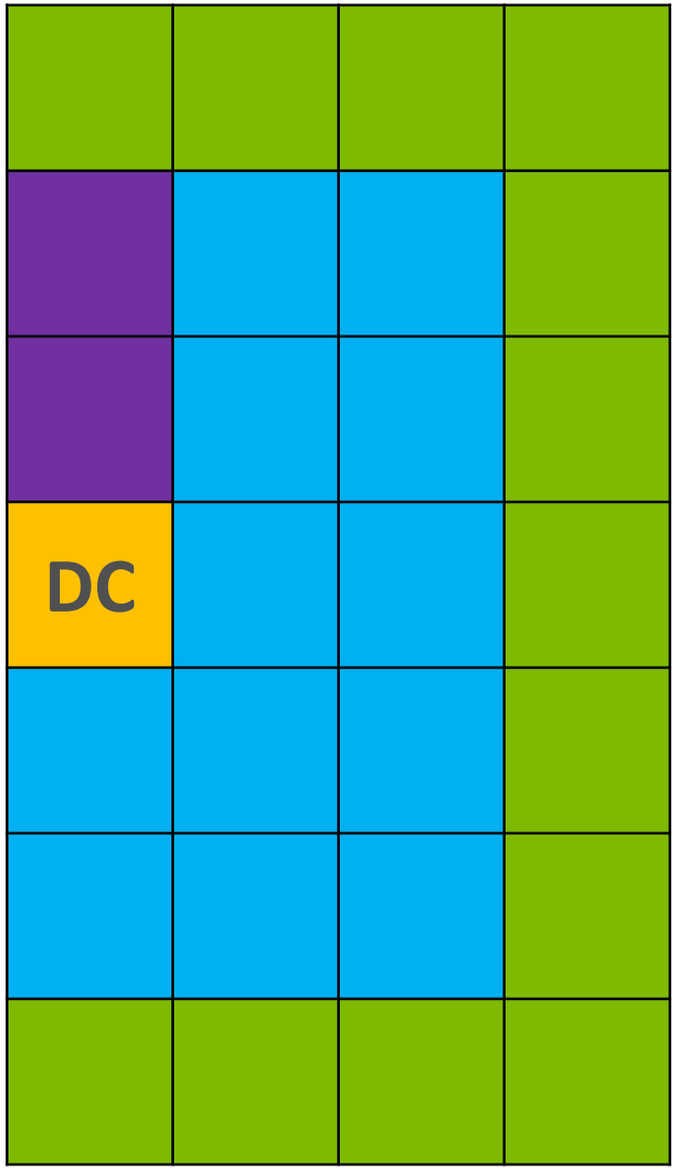
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints
5. 1st compression



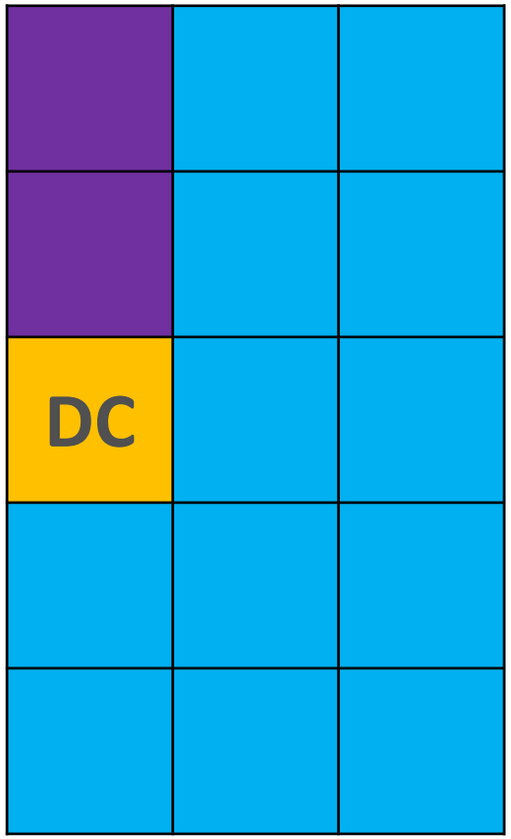
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints
5. 1st compression
6. 2nd compression



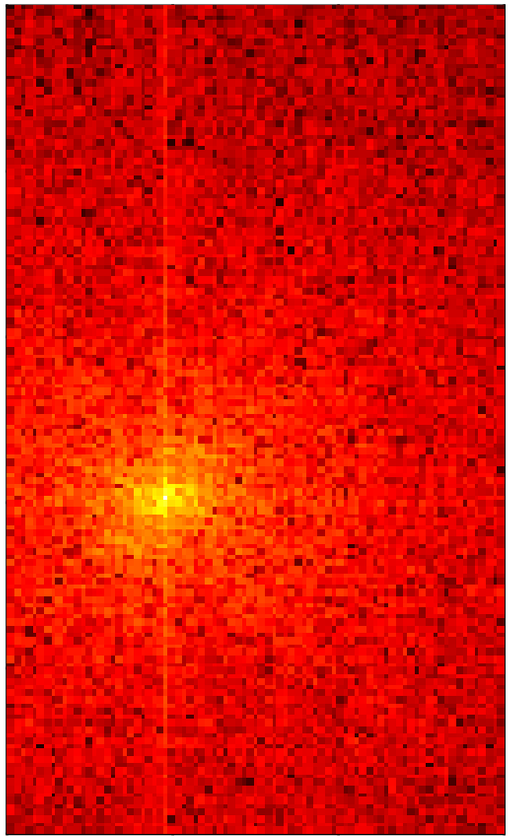
Band-limiting Technique

1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints
5. 1st compression
6. 2nd compression

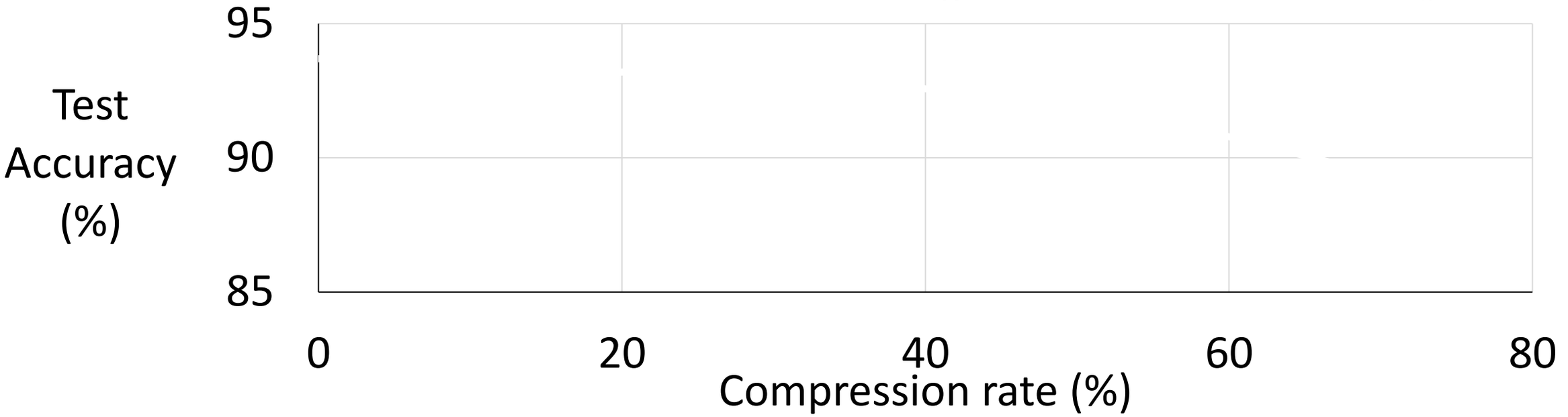


Band-limiting Technique

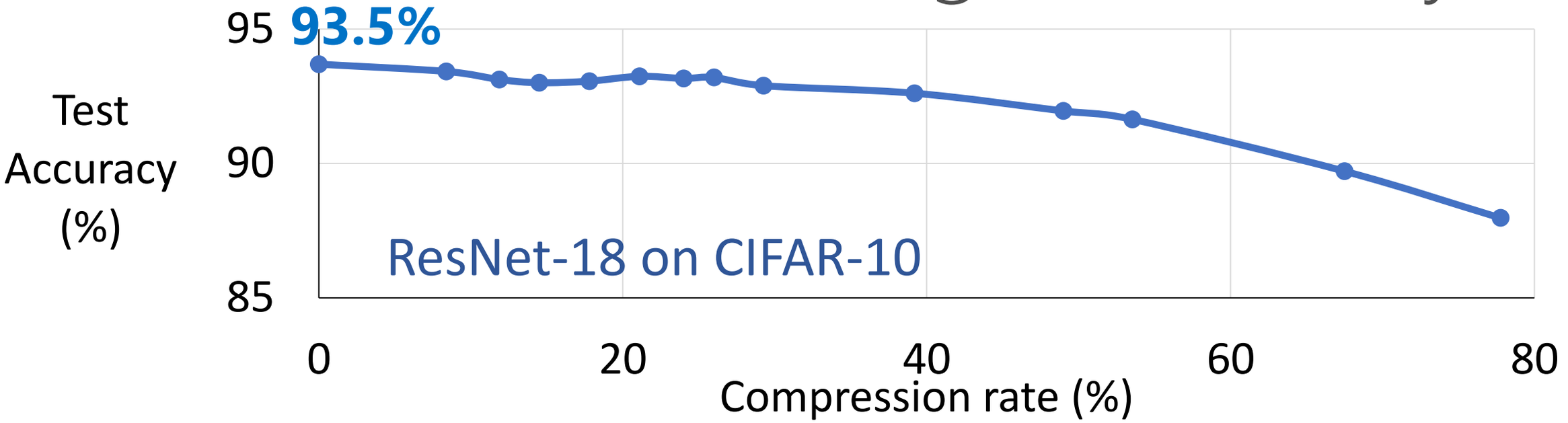
1. FFT of an input data
2. Conjugate symmetry
3. Real values
4. No constraints
5. 1st compression
6. 2nd compression



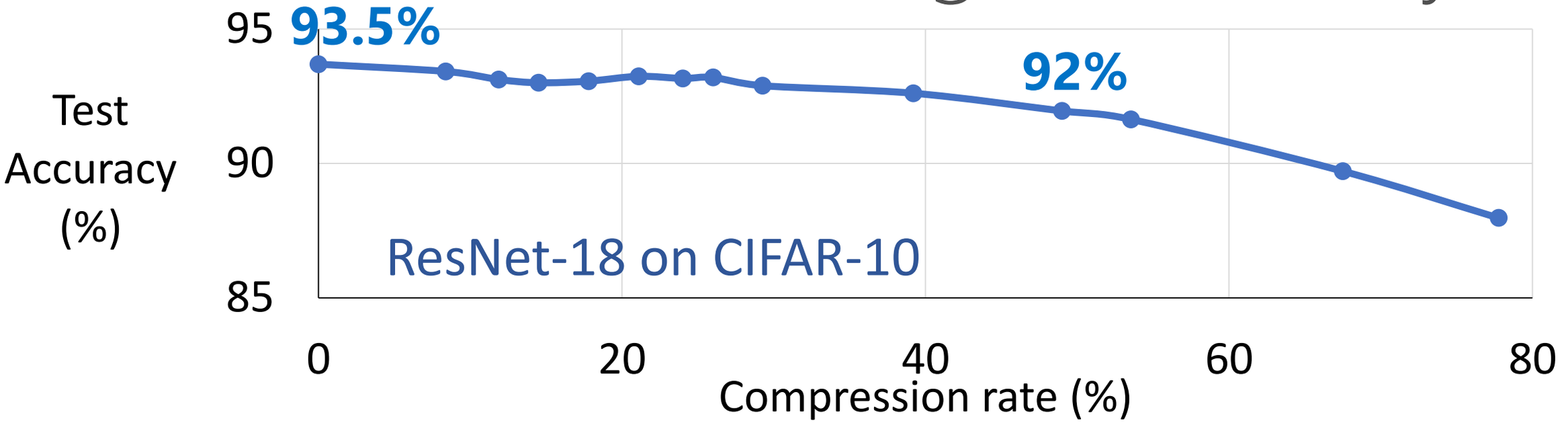
Effects of band-limiting on accuracy



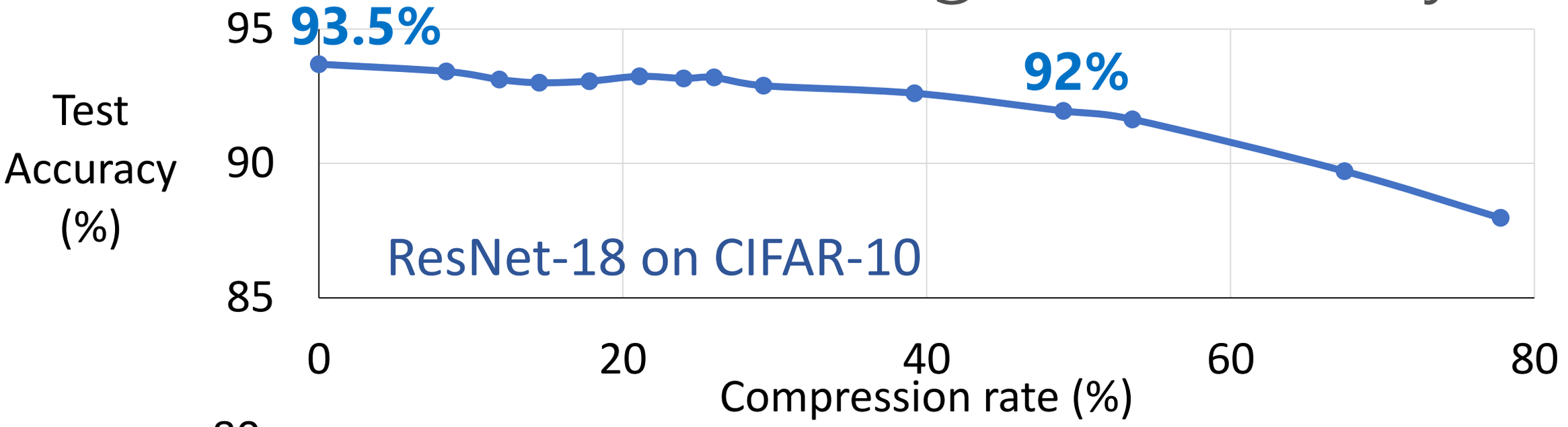
Effects of band-limiting on accuracy

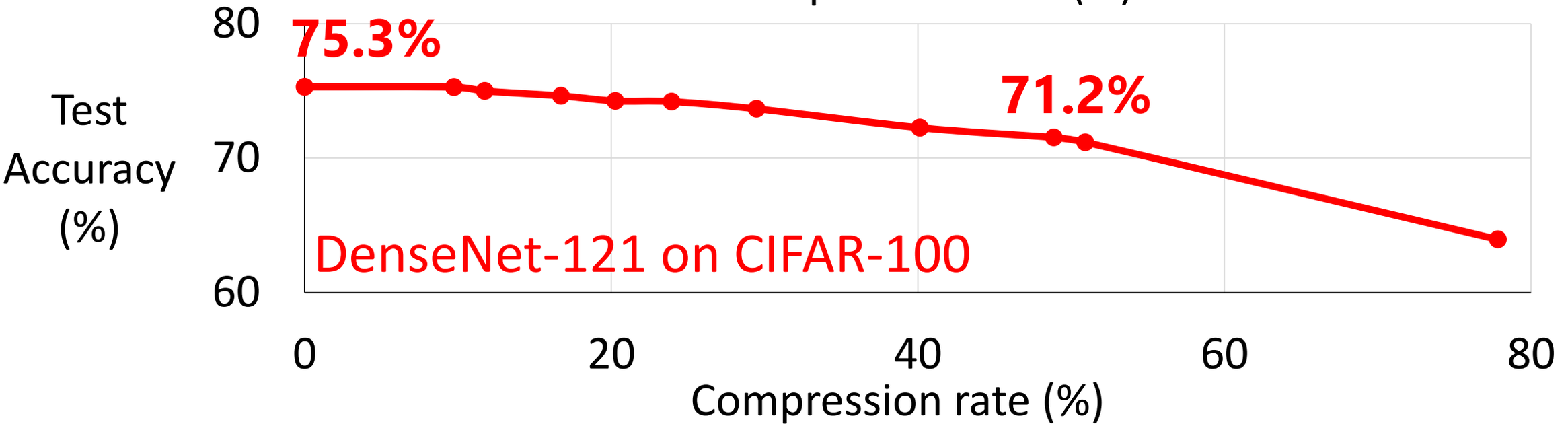


Effects of band-limiting on accuracy

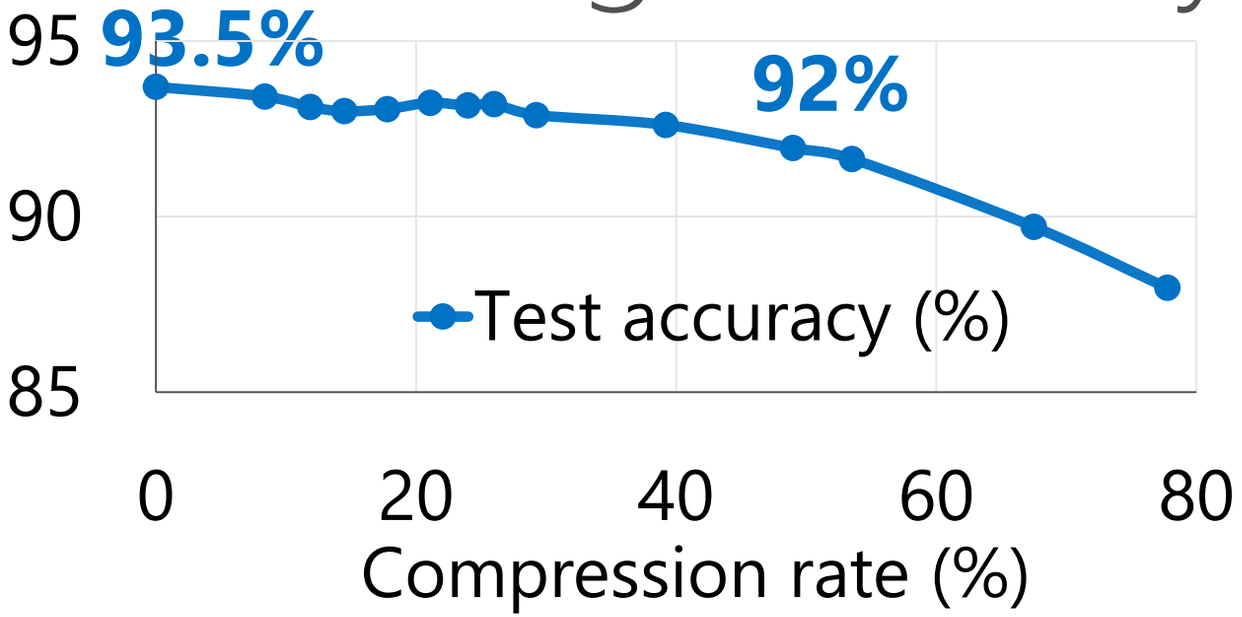


Effects of band-limiting on accuracy

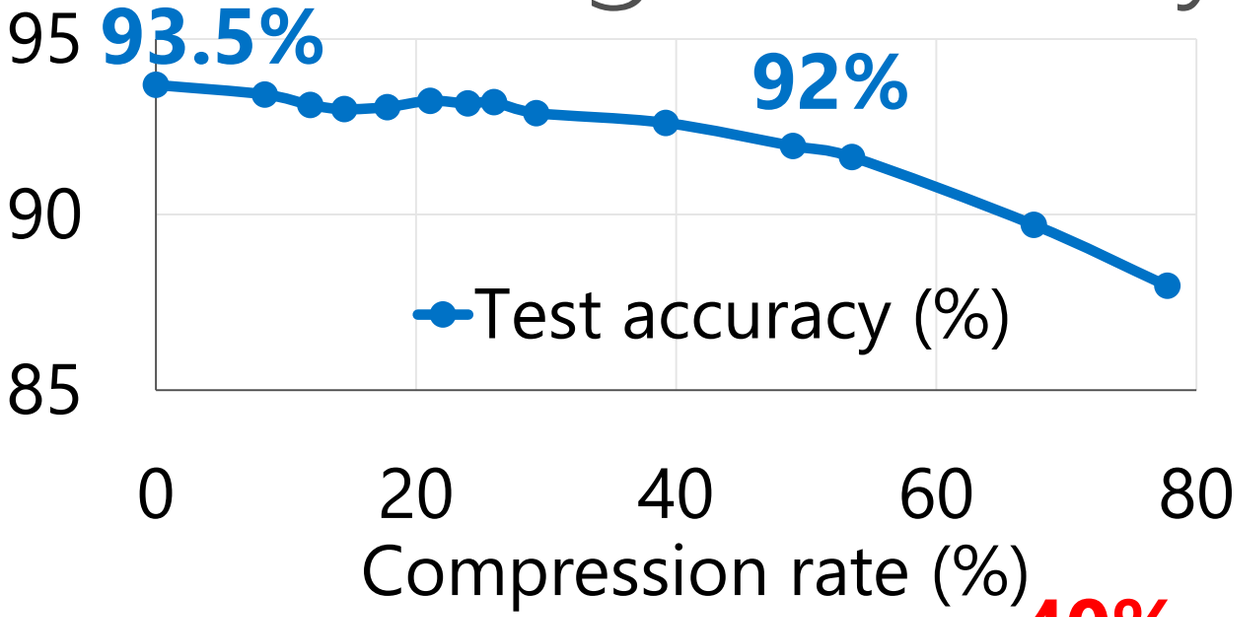


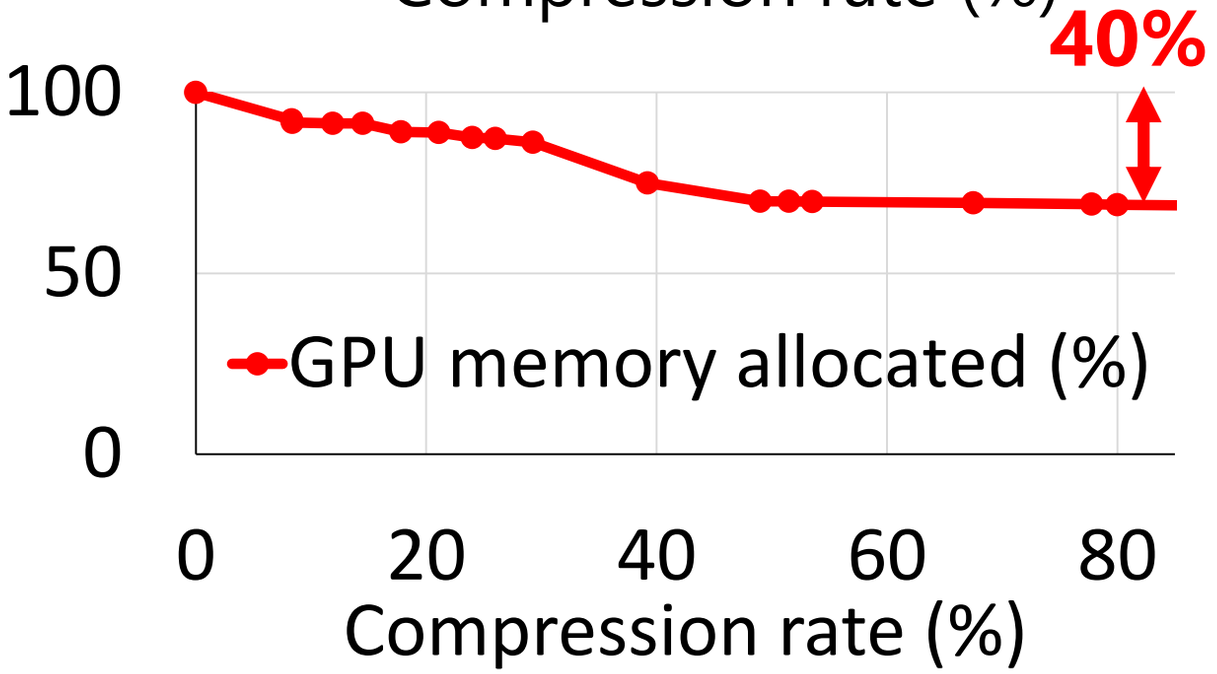


Tuning: Accuracy vs Performance

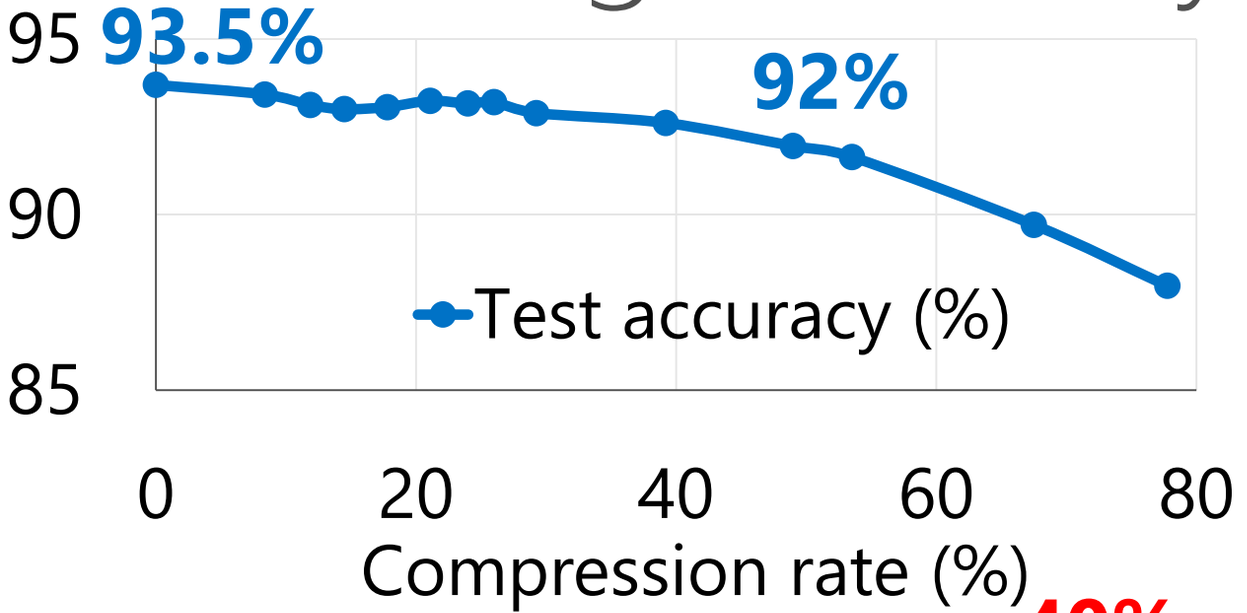


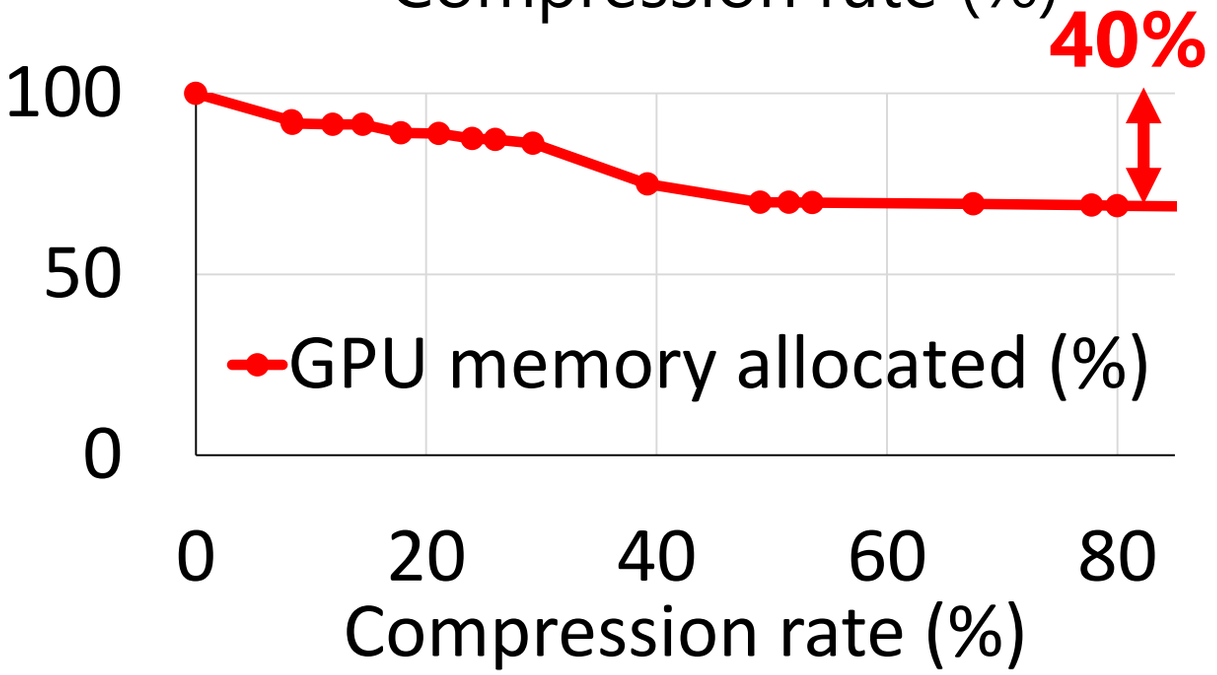
Tuning: Accuracy vs Performance

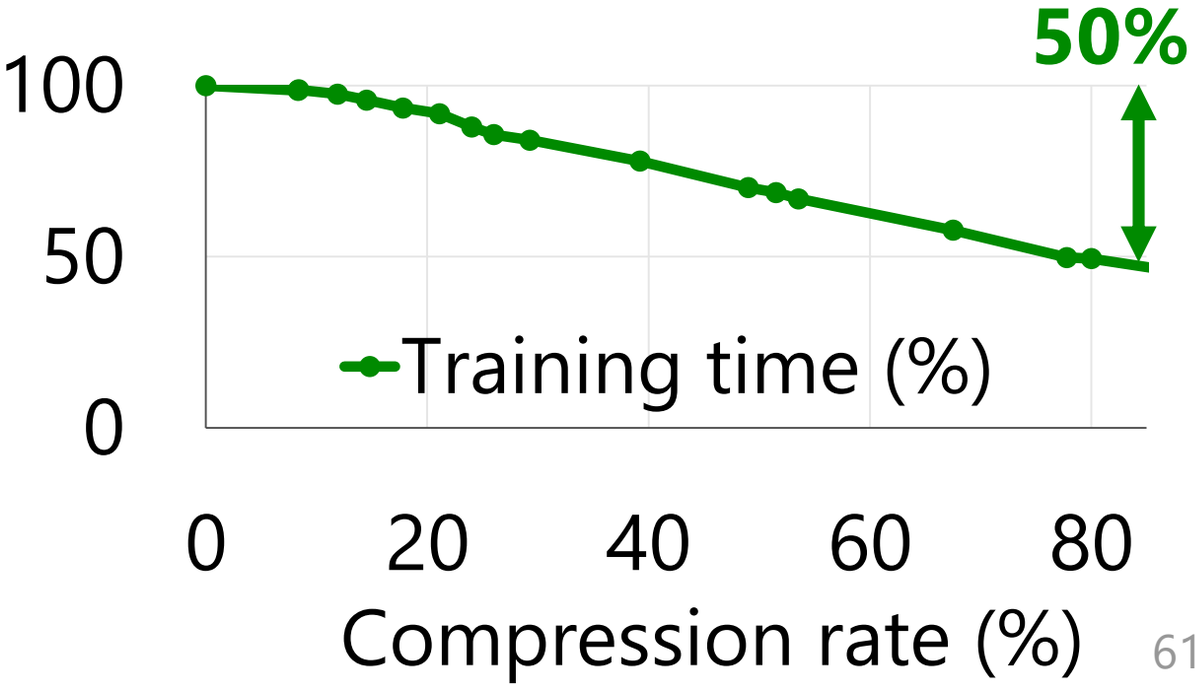




Tuning: Accuracy vs Performance

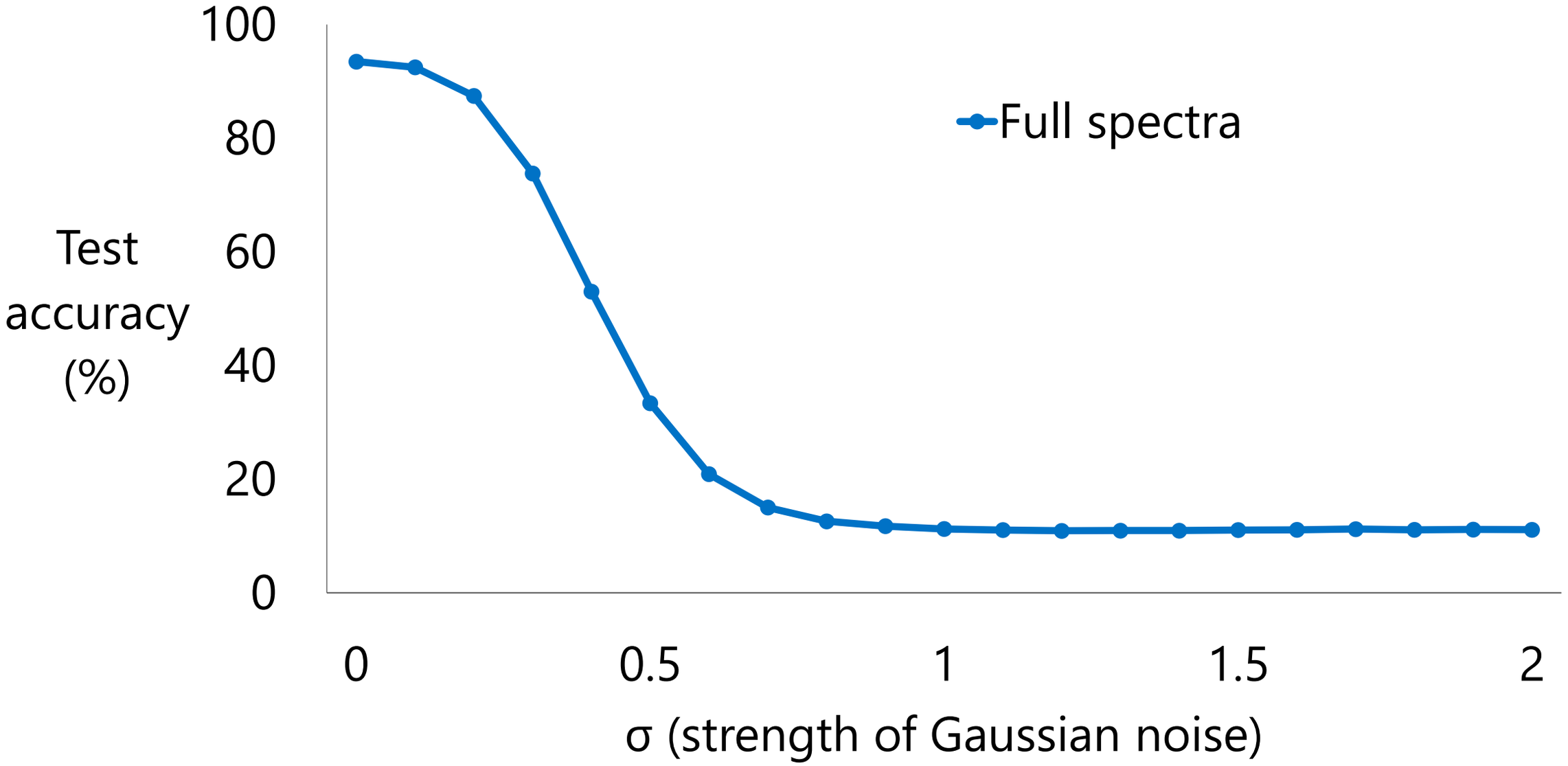




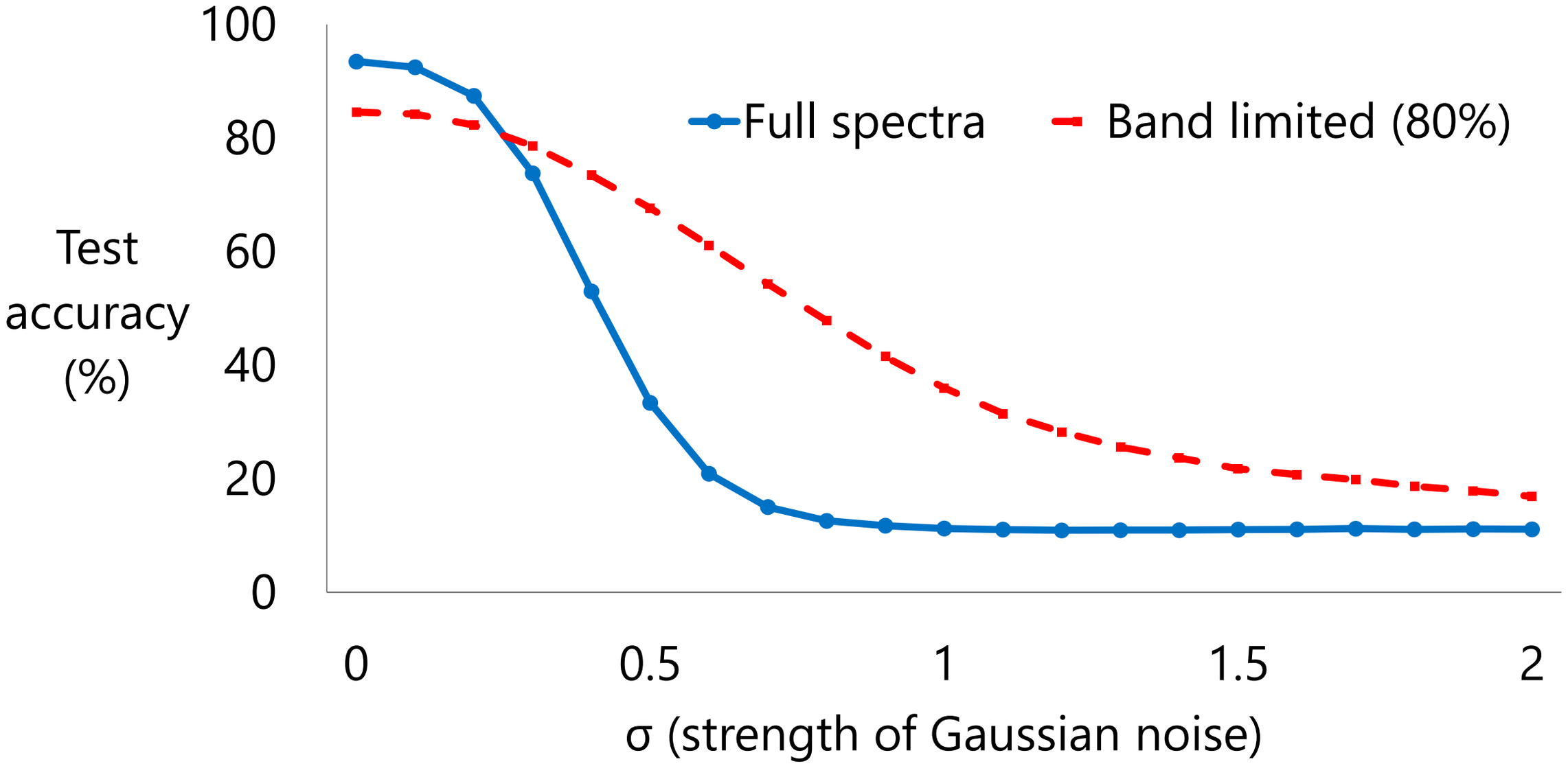


Robustness to noise

Robustness to noise



Robustness to noise



1. Introduction & FFT-based convolution

2. Attacks and defenses

3. Research plan

Classify the input

Classify the input

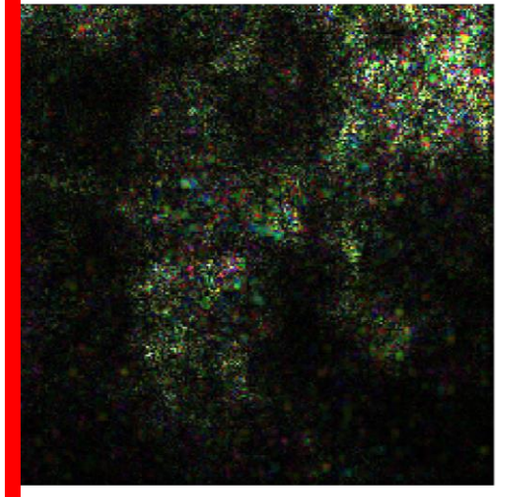
original label: bull mastiff
confidence: 0.9592
L2 distance: 0.0

Adversarial examples

ATTACK

original label: bull mastiff
confidence: 0.9592
L2 distance: 0.0

difference 100x
between images:
original and adversarial



adv. label: tiger cat
confidence: 0.21
L2 distance: 1.3871

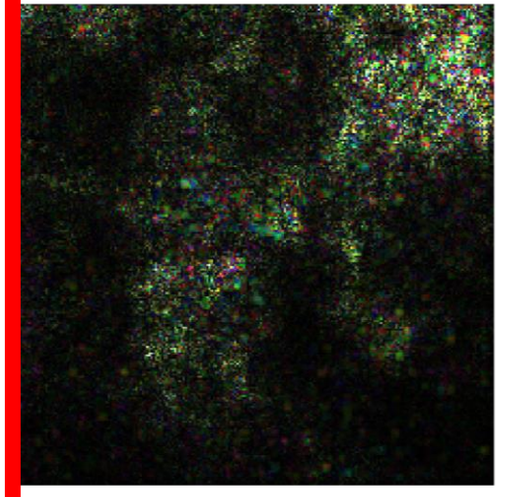
Adversarial examples are not robust and small perturbations recover correct class

ATTACK

DEFENSE

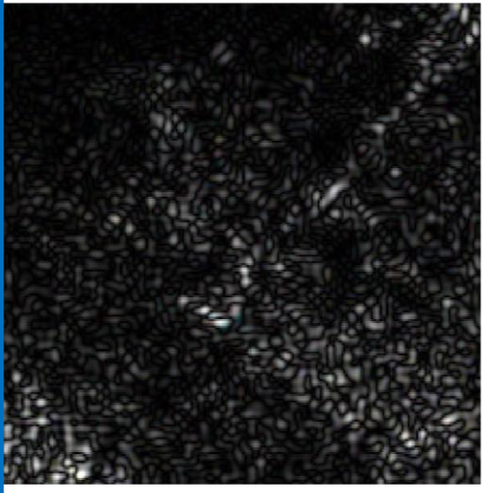
original label: bull mastiff
confidence: 0.9592
L2 distance: 0.0

difference 100x
between images:
original and adversarial



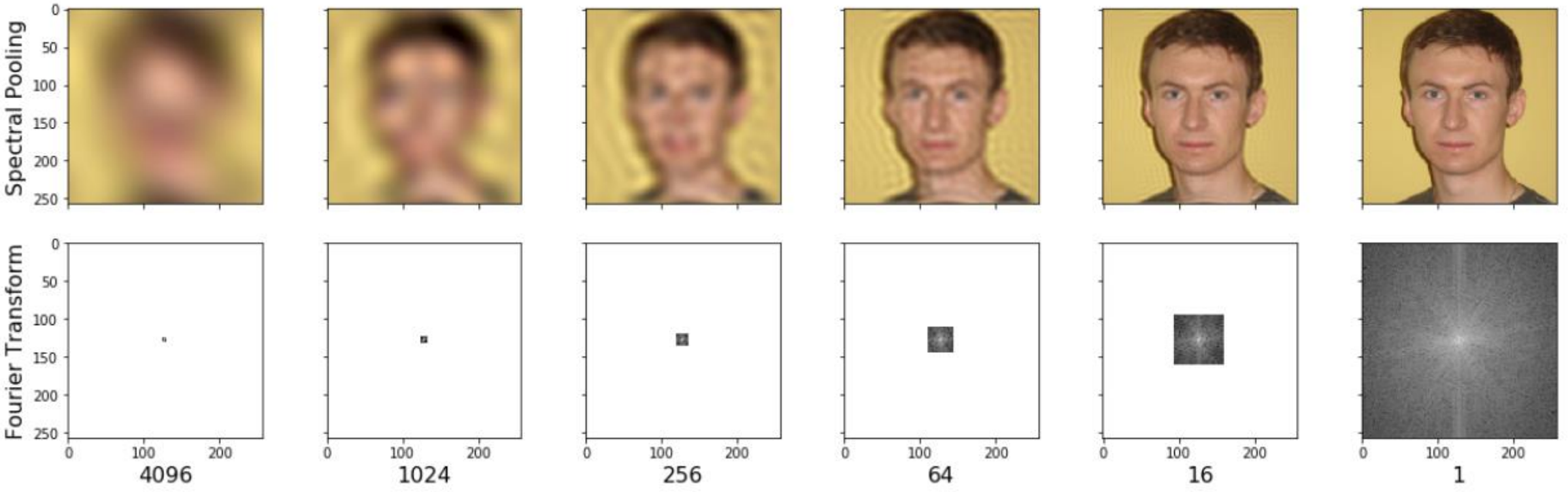
adv. label: tiger cat
confidence: 0.21
L2 distance: 1.3871

difference 10x
between images:
adversarial and compressed



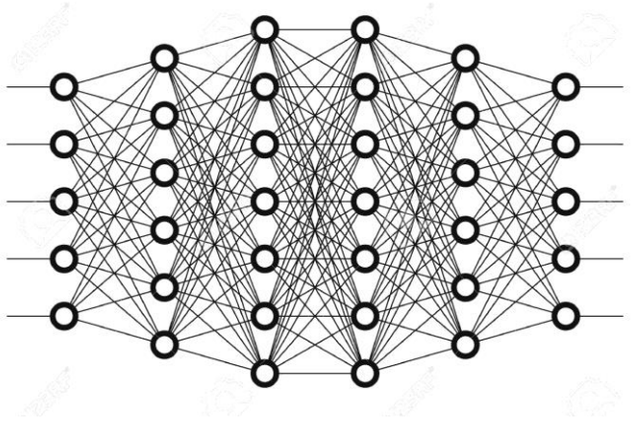
fft label: bull mastiff
confidence: 0.3638
L2 distance: 12.6992

FFT-based compression

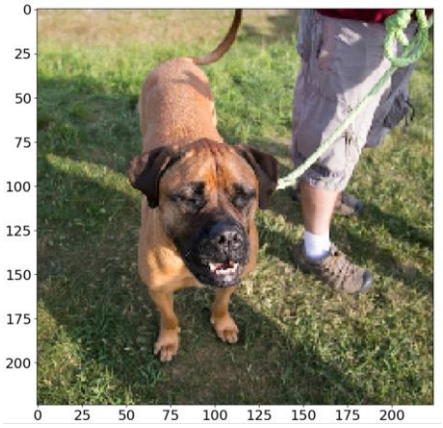


White-box attacks: access the gradients

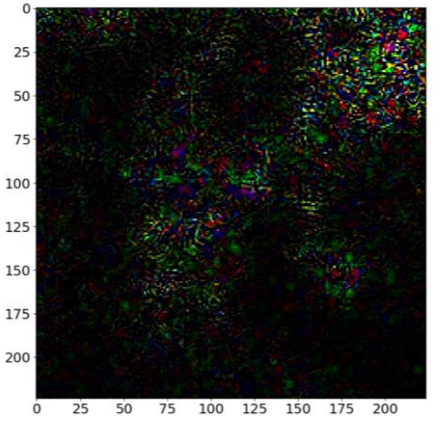
Bull mastiff



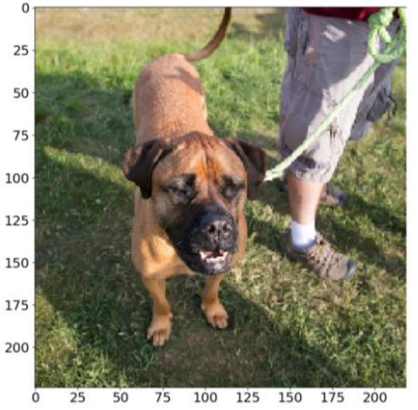
Tiger cat



$+ \quad \varepsilon * \quad$



$=$



$\boldsymbol{x}$ $\qquad$ $\boldsymbol{\nabla_x Loss(p(x), y)}$

Fast Gradient Sign Method (FGSM)

1. One-step attack
2. L_∞ bounded changes to each pixel by ε

$$\hat{x} = x + \varepsilon \, \mathrm{sgn}(\nabla_x L(x, y; \theta))$$

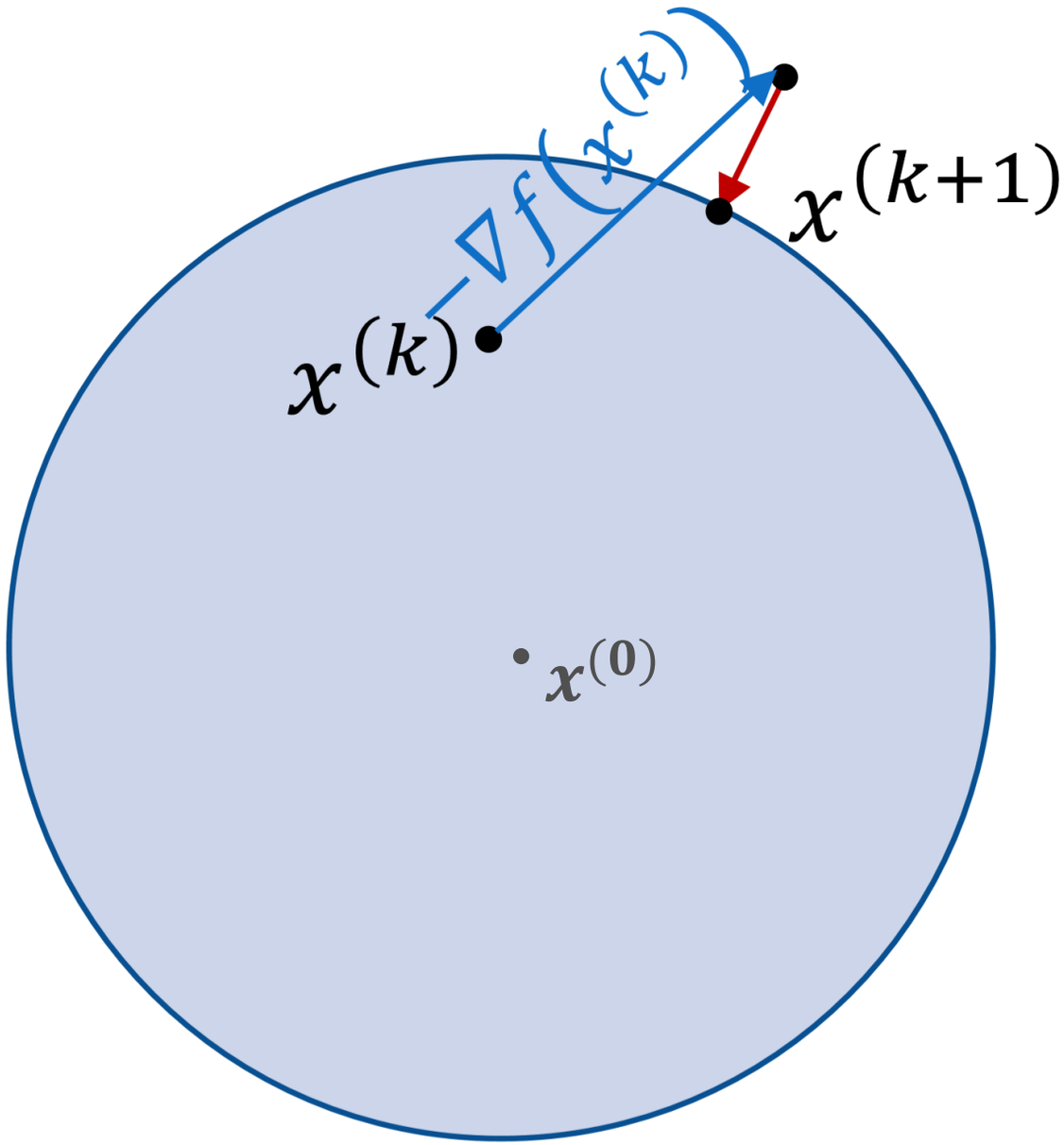
Ultimate white-box attack

Projected Gradient Descent (PGD)

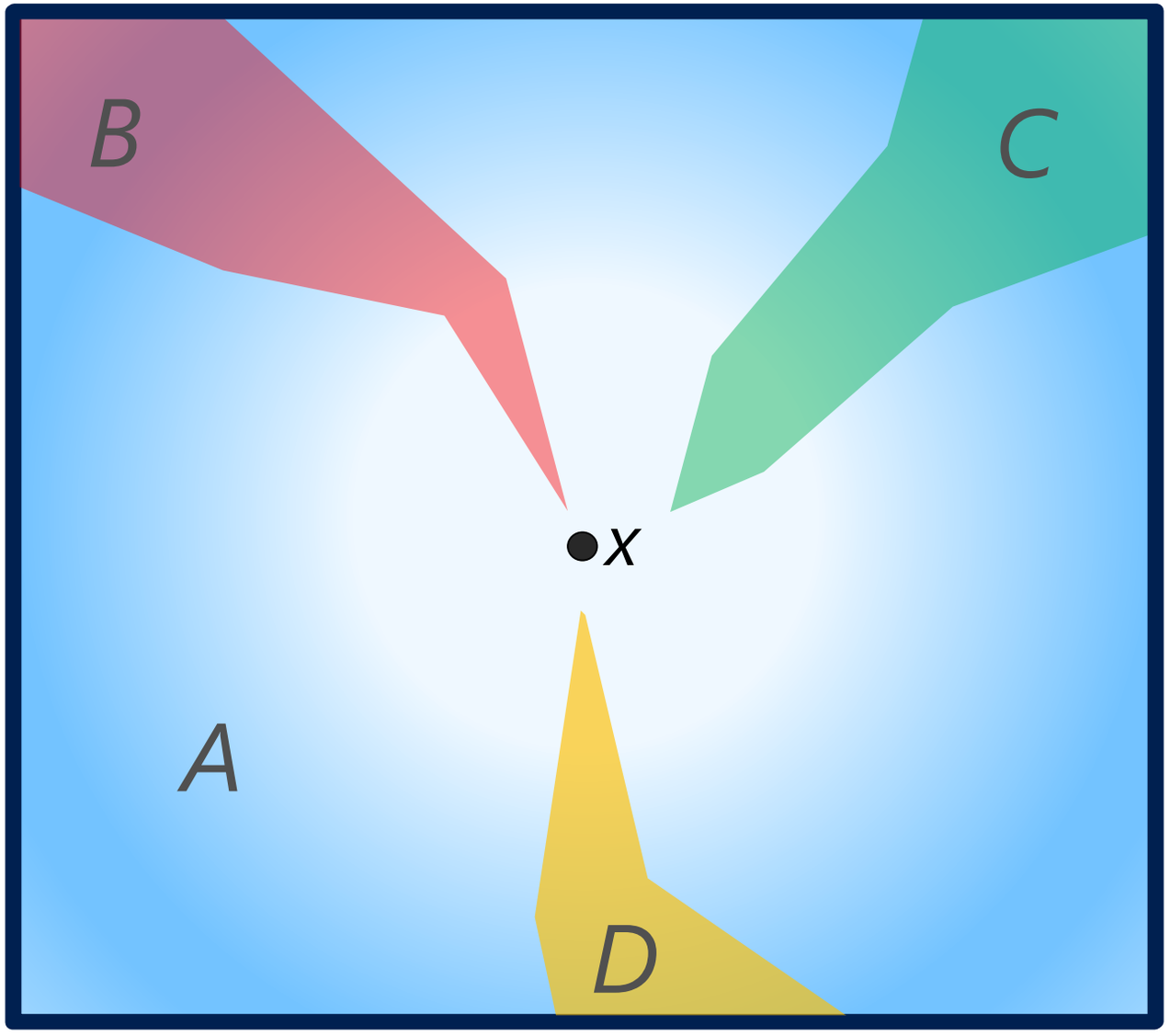
1. Powerful adversary
2. Multi-step variant of the FGSM attack

$$x^{k+1} = Proj_{x+S}(x^k + \varepsilon\, sgn\left(\nabla_{x^k} L\left(x^k, y, \theta\right)\right))$$

PGD: Projected Gradient Descent



Shape of adversarial regions



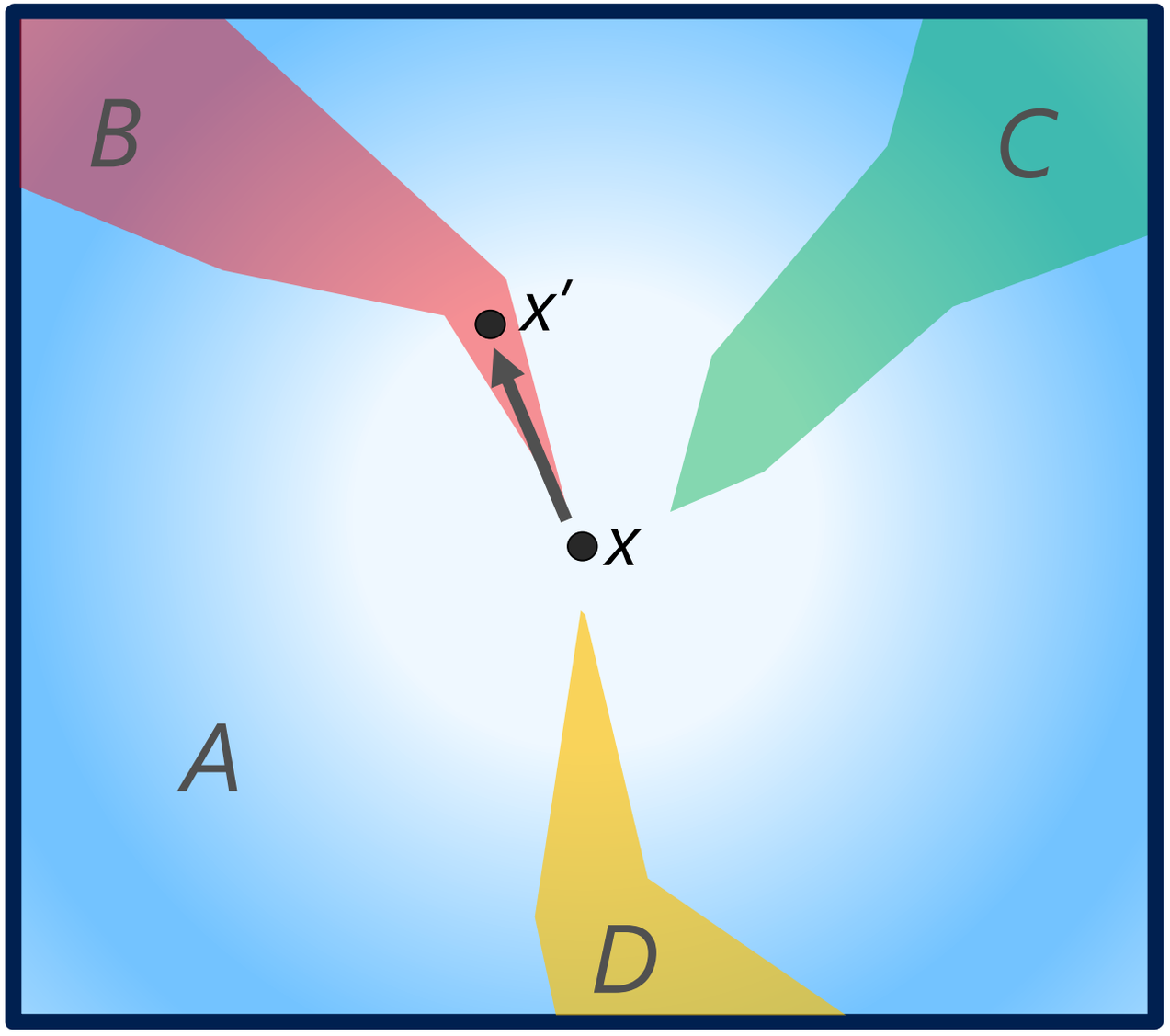
Natural image x:

Criterion 1: robustness to random noise

Criterion 2: close to *decision boundaries*

[7] Hu et al. A new defense against adversarial images: Turning a weakness into a strength.

Shape of adversarial regions



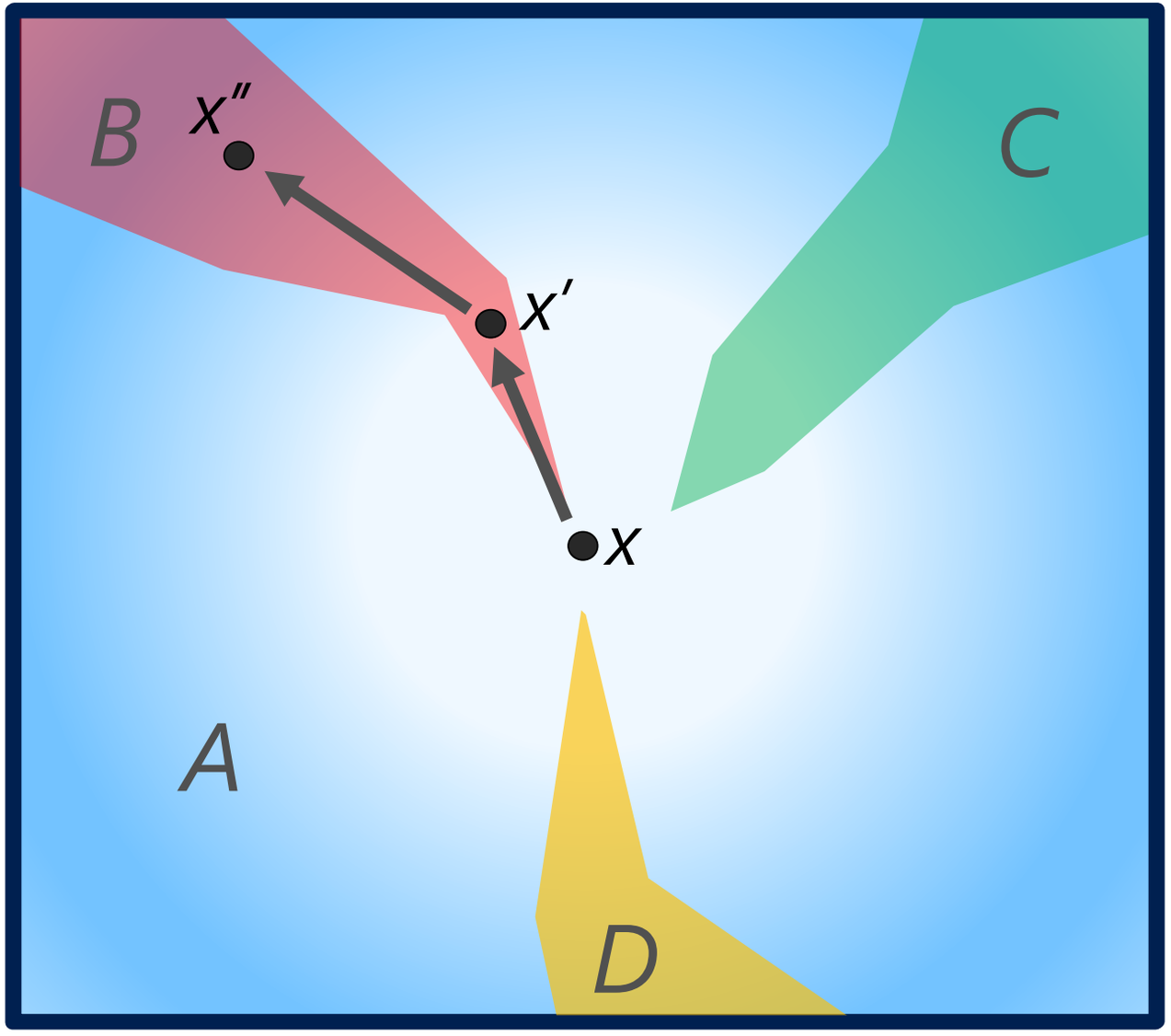
Adversarial image x′:

Criterion 1: not robust to random noise

Criterion 2: close to *decision boundaries*

[7] Hu et al. A new defense against adversarial images: Turning a weakness into a strength.

Shape of adversarial regions: natural images



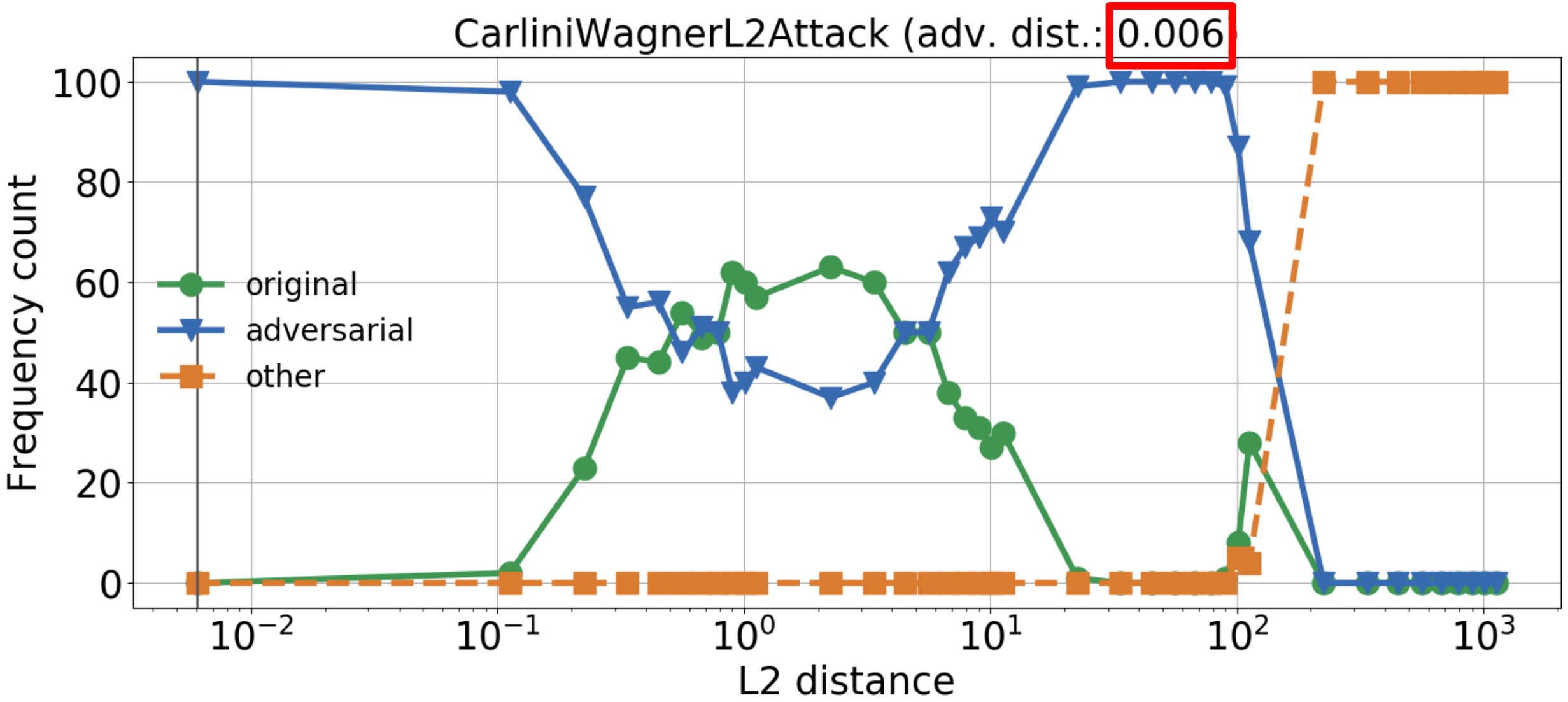
Adversarial image x″:

Criterion 1: robust to random noise

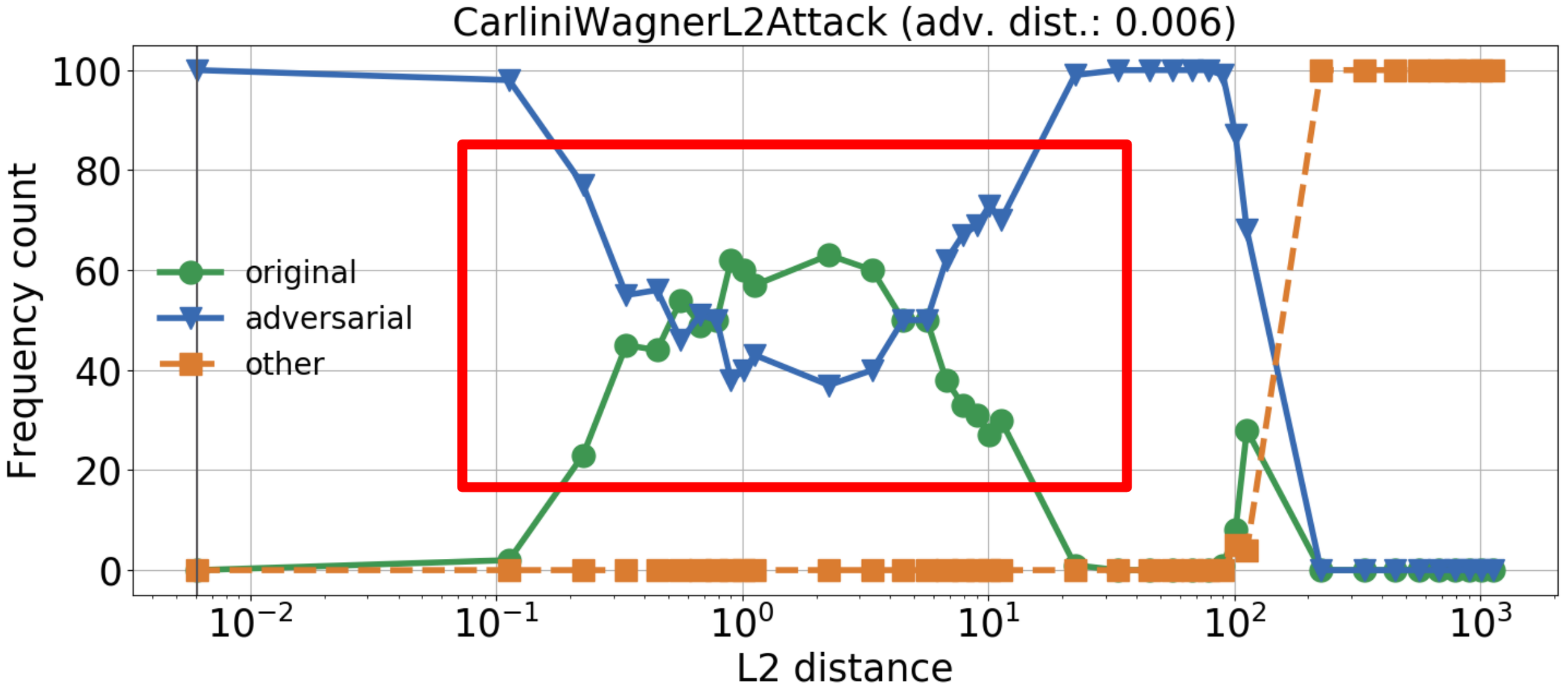
Criterion 2: not close to *decision boundaries*

[7] Hu et al. A new defense against adversarial images: Turning a weakness into a strength.

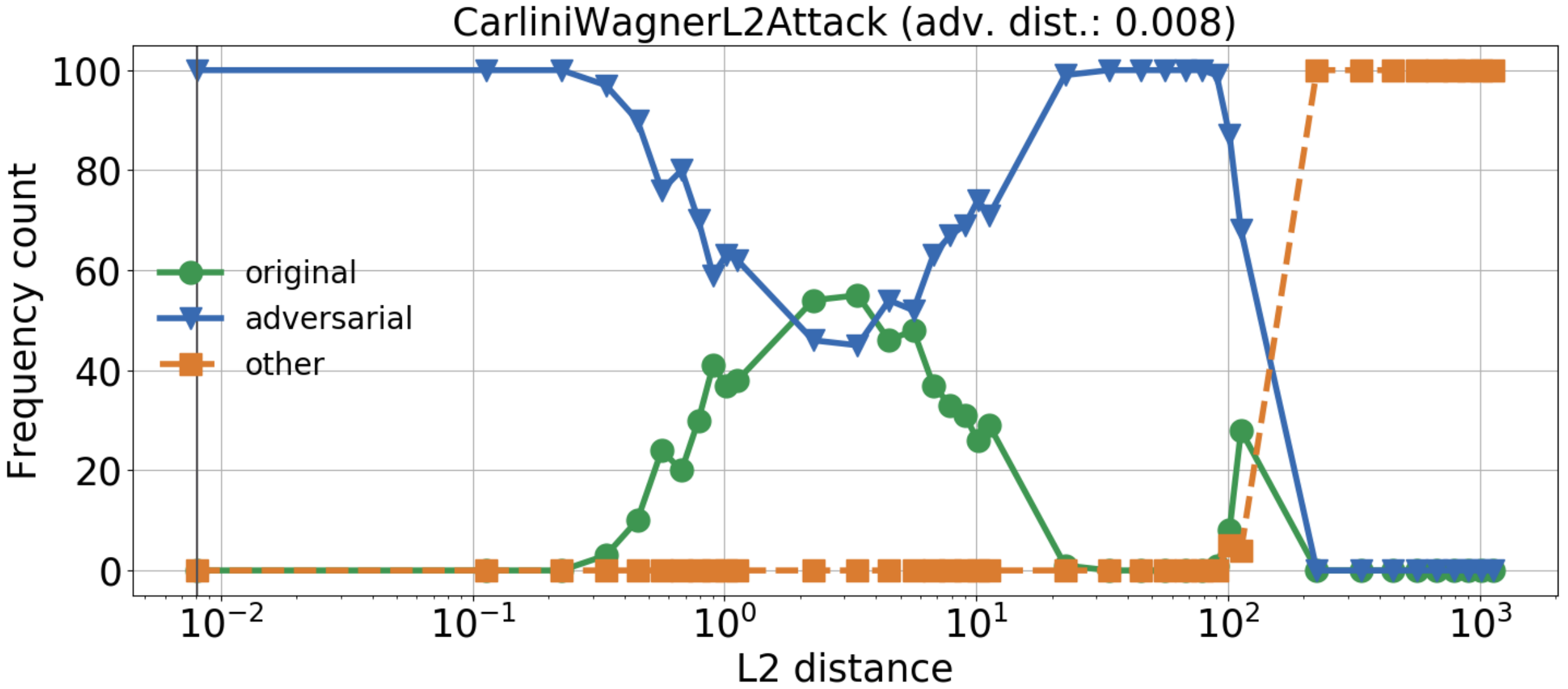
Recovery window shrinks with higher adversarial distortion



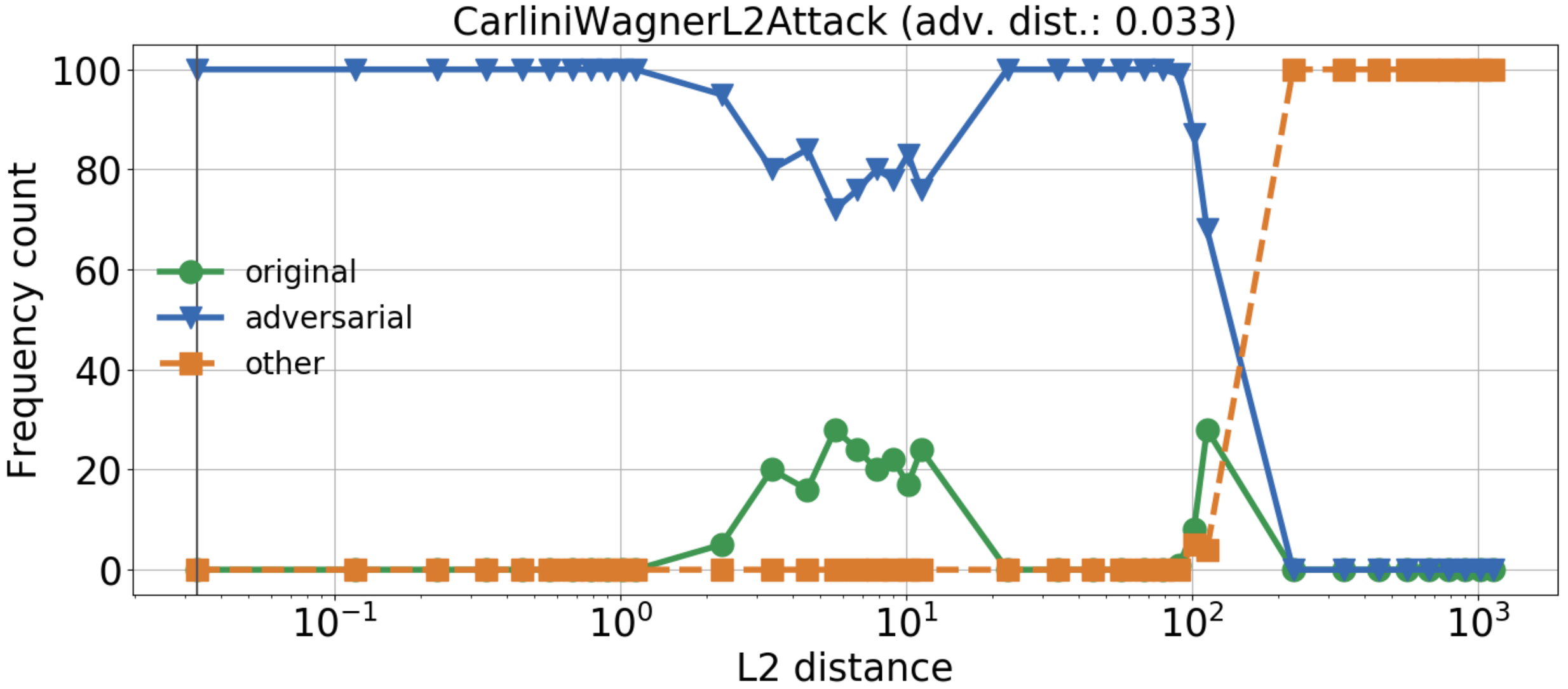
Recovery window shrinks with higher adversarial distortion



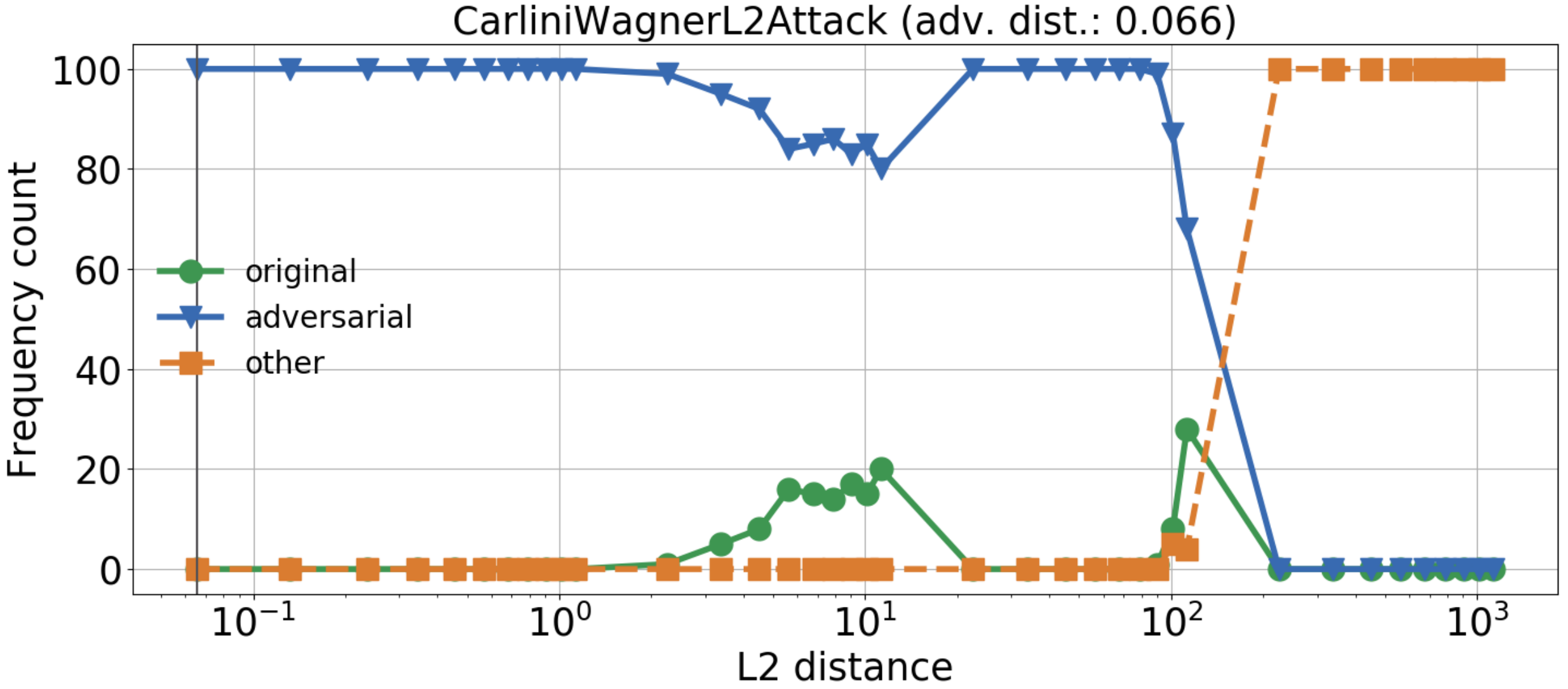
Recovery window shrinks with higher adversarial distortion



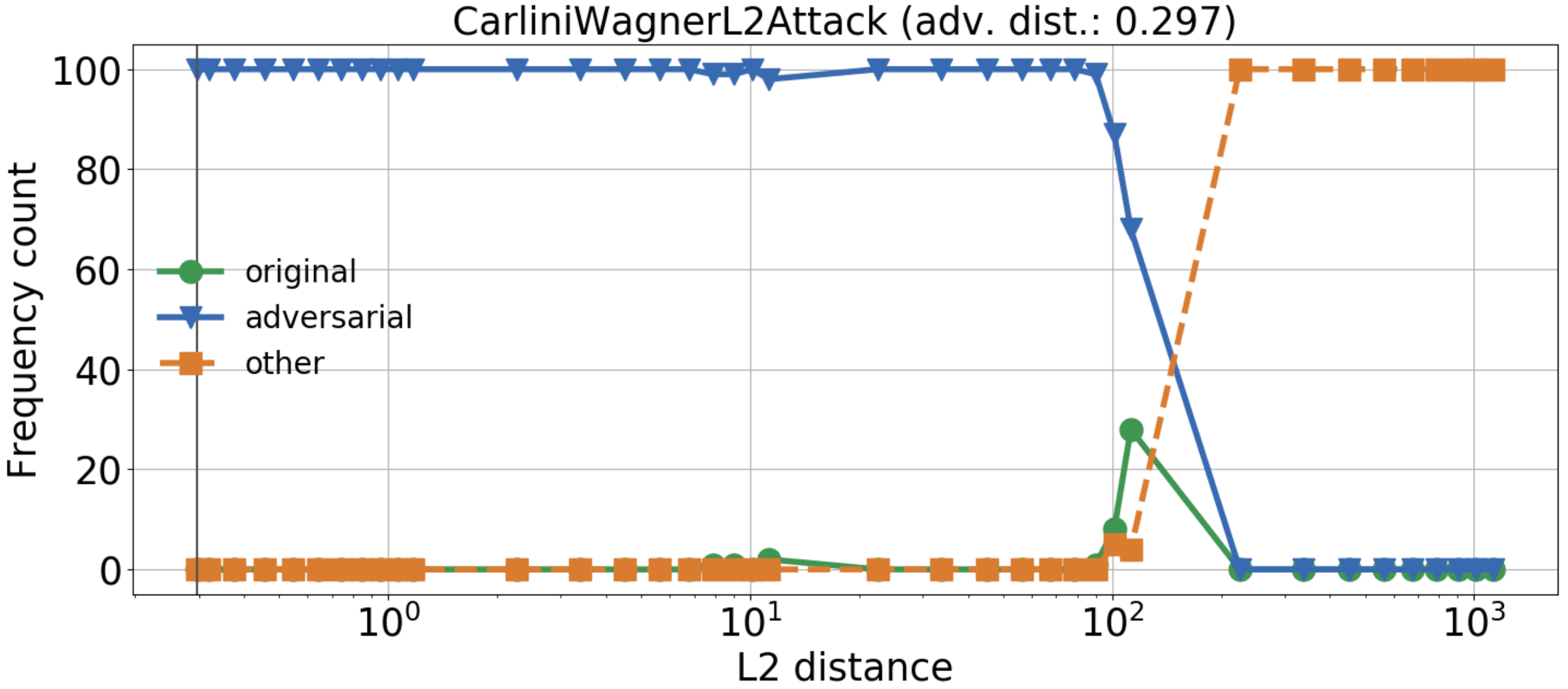
Recovery window shrinks with higher adversarial distortion



Recovery window shrinks with higher adversarial distortion



Recovery window shrinks with higher adversarial distortion



Defense with lossy channels

ImagNet trained on ResNet-50, ***baseline accuracy 83.5%,*** *attack on 1000 images.*

	FGSM	PGD	C&W
Empty channel	0.0	0.0	0.0

Defense with lossy channels

ImagNet trained on ResNet-50, ***baseline accuracy 83.5%,*** *attack on 1000 images.*

	FGSM	PGD	C&W
Empty channel	0.0	0.0	0.0
FFT compression	73.8	82.1	78.7

Defense with lossy channels

ImagNet trained on ResNet-50, ***baseline accuracy 83.5%,*** *attack on 1000 images.*

	FGSM	PGD	C&W
Empty channel	0.0	0.0	0.0
FFT compression	73.8	82.1	78.7
Feature squeezing	76.0	82.9	80.9

Defense with lossy channels

ImagNet trained on ResNet-50, ***baseline accuracy 83.5%,*** *attack on 1000 images.*

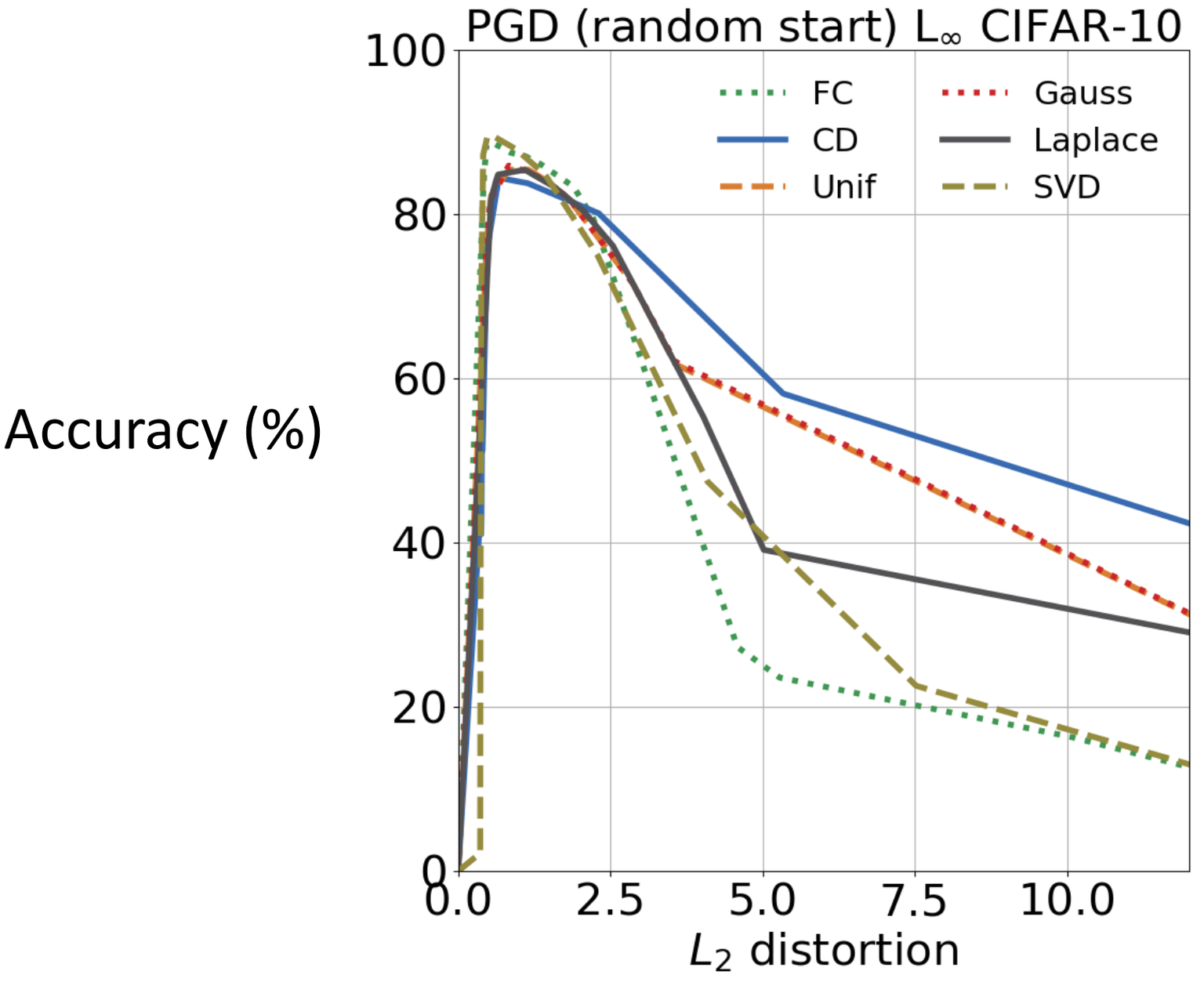
	FGSM	PGD	C&W
Empty channel	0.0	0.0	0.0
FFT compression	73.8	82.1	78.7
Feature squeezing	76.0	82.9	80.9
Gaussian noise	75.4	80.9	80.4

Defense with lossy channels

ImagNet trained on ResNet-50, ***baseline accuracy 83.5%,*** *attack on 1000 images.*

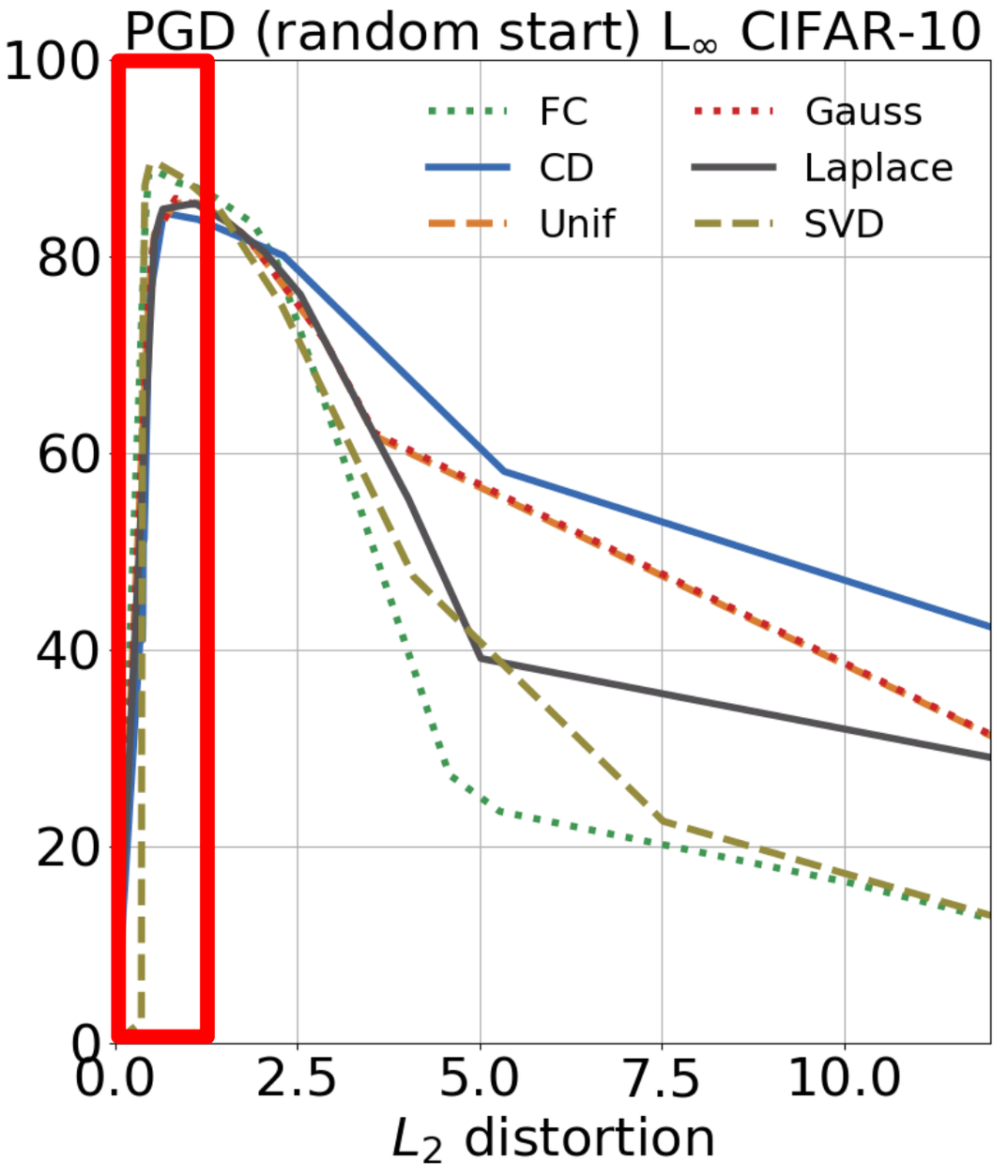
	FGSM	PGD	C&W
Empty channel	0.0	0.0	0.0
FFT compression	73.8	82.1	78.7
Feature squeezing	76.0	82.9	80.9
Gaussian noise	75.4	80.9	80.4

Defense with lossy channels against PGD

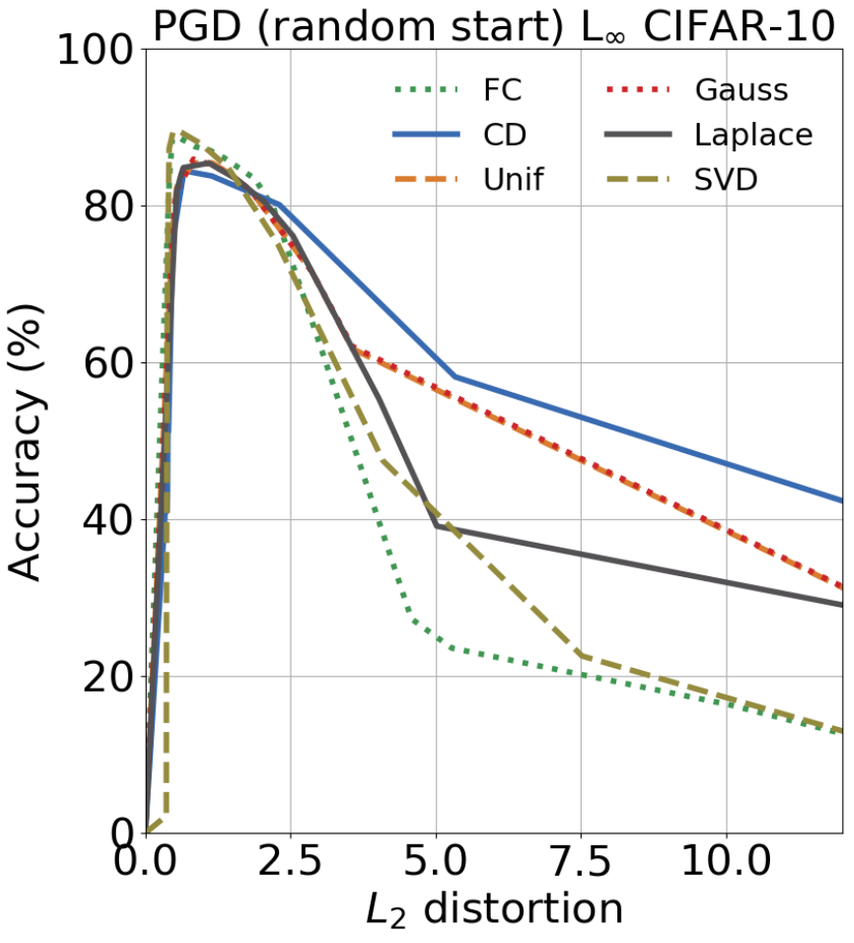


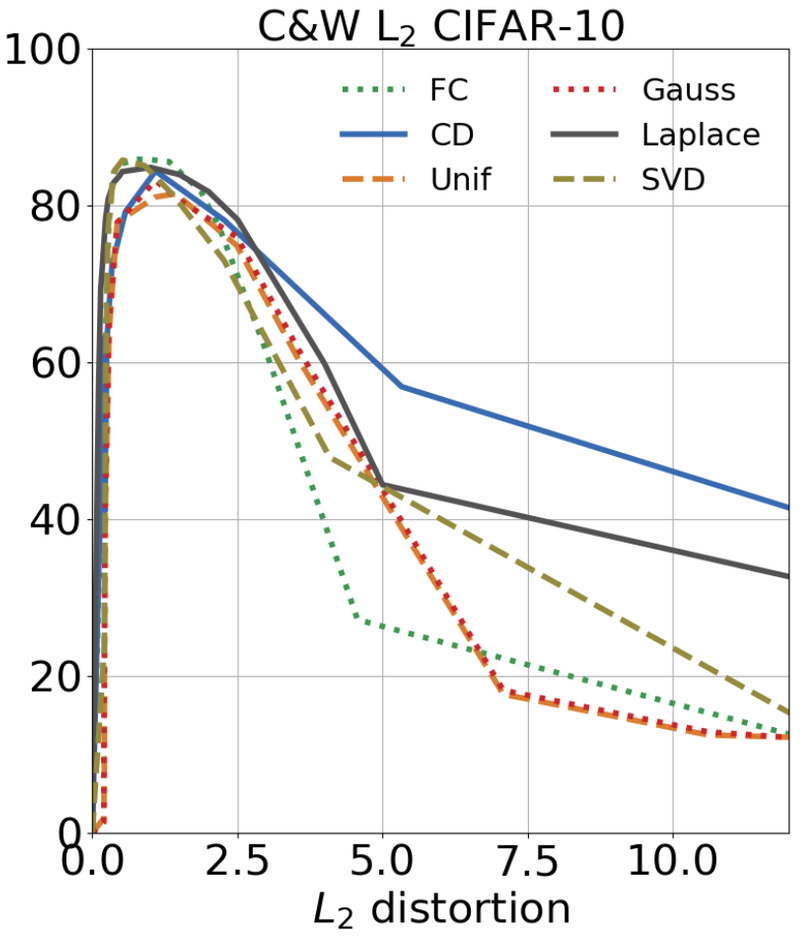
Defense with lossy channels against PGD

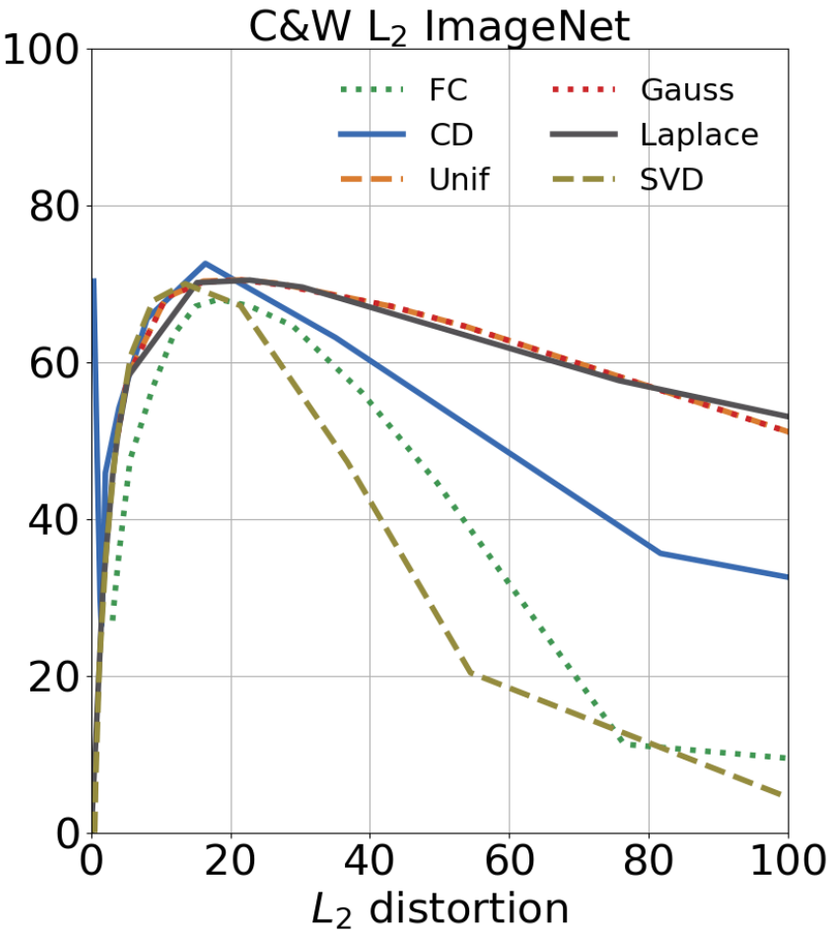
Accuracy (%)



Similarity of the lossy channels

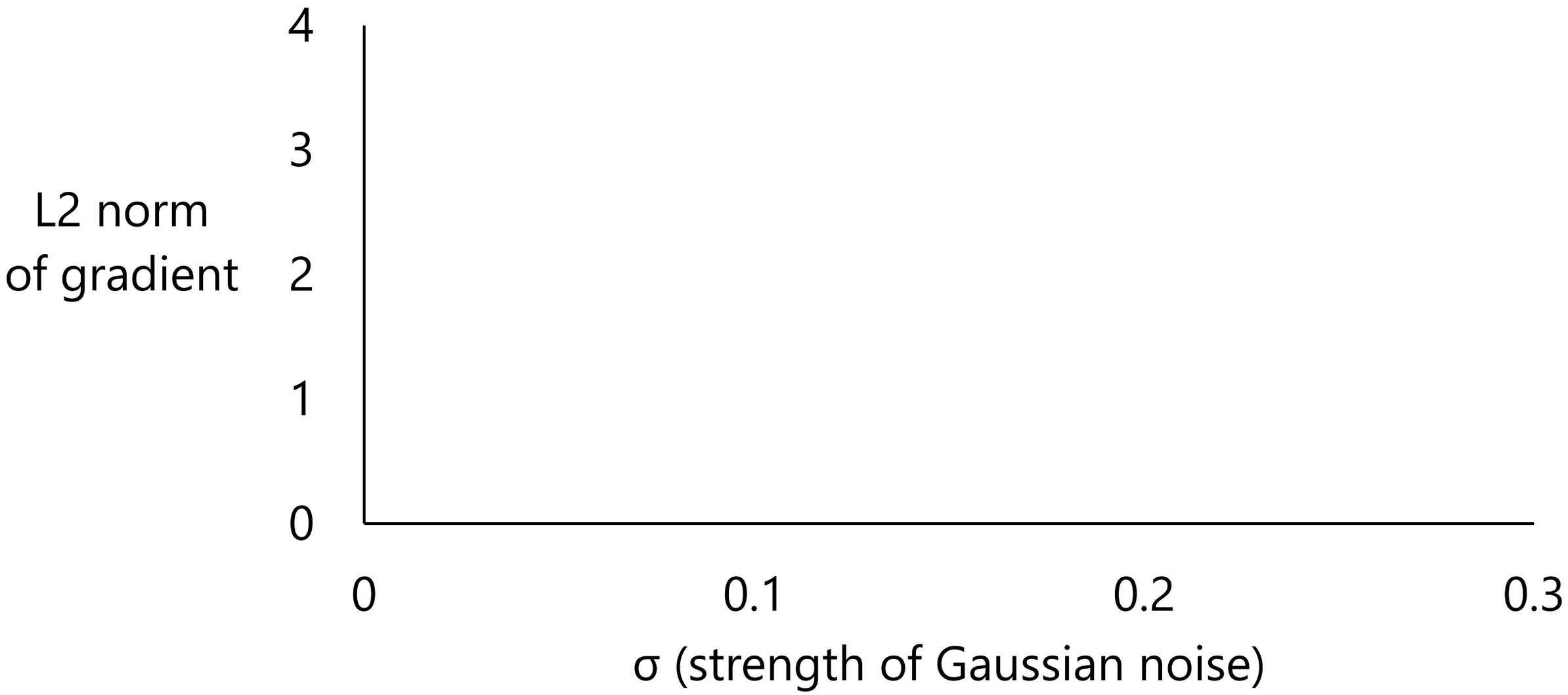




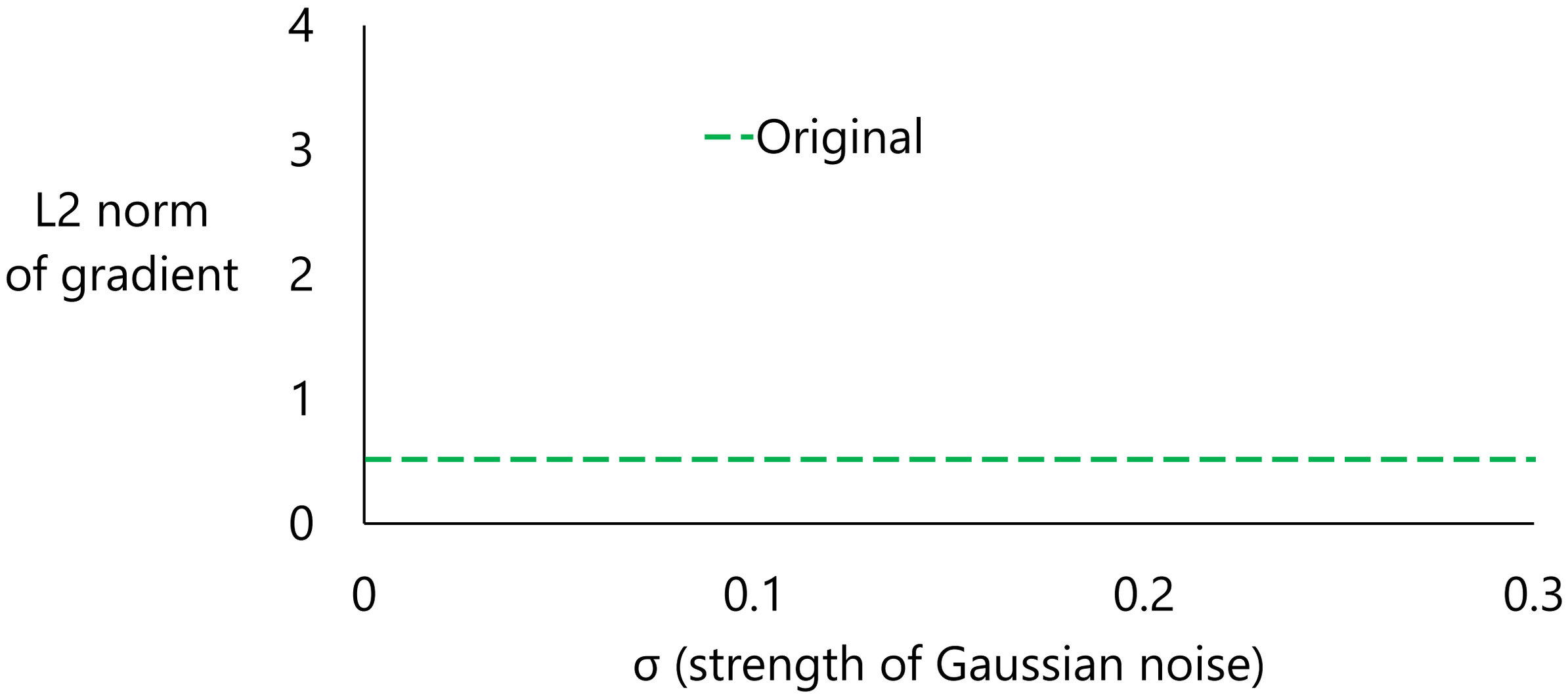


1st & 2nd order analysis

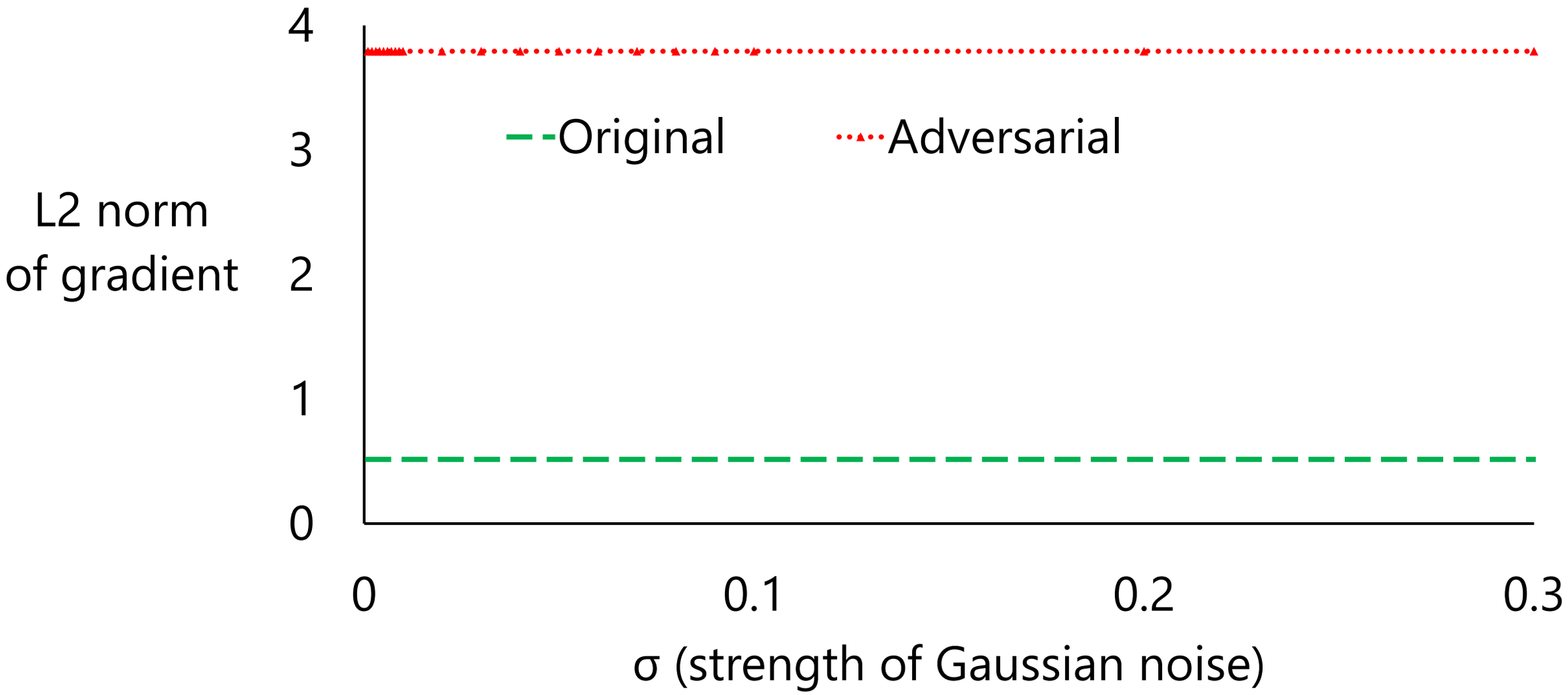
Gradient-based analysis



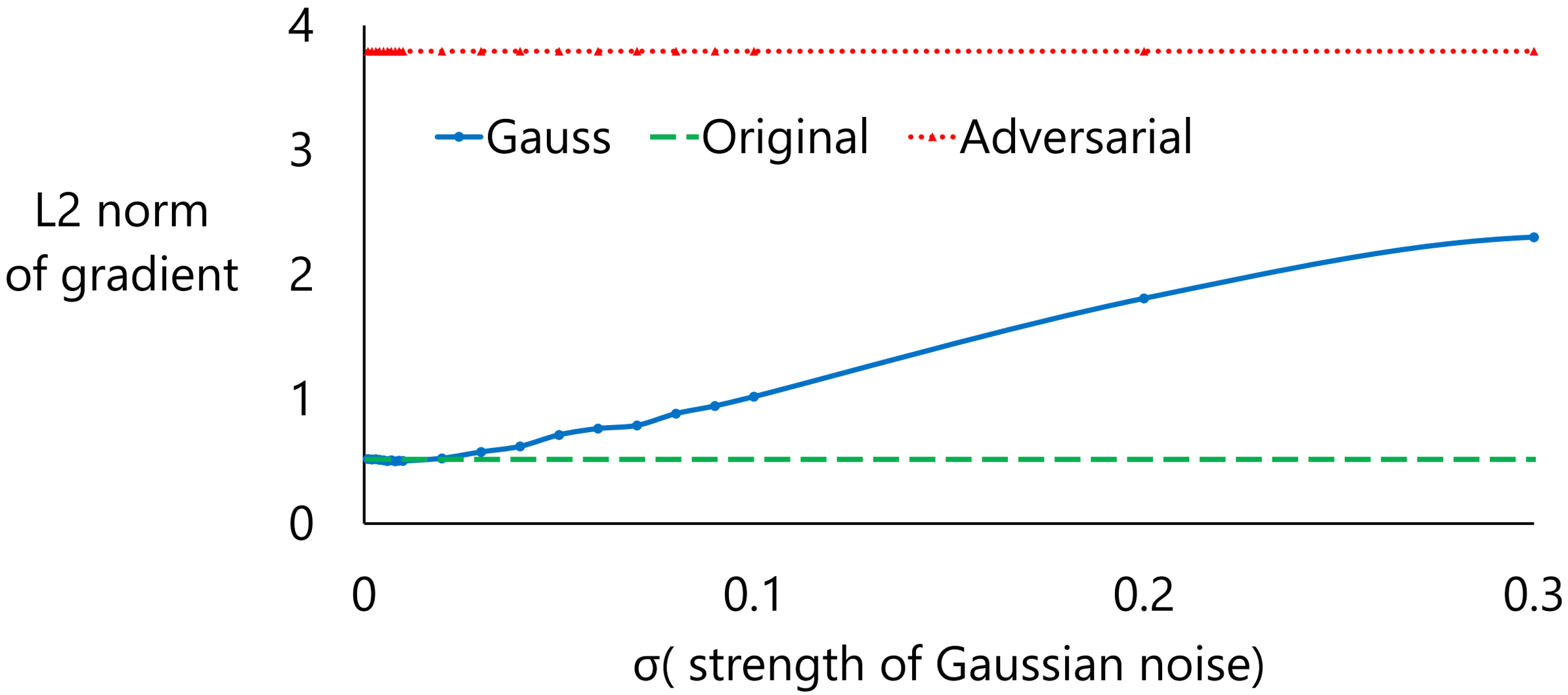
Gradient-based analysis



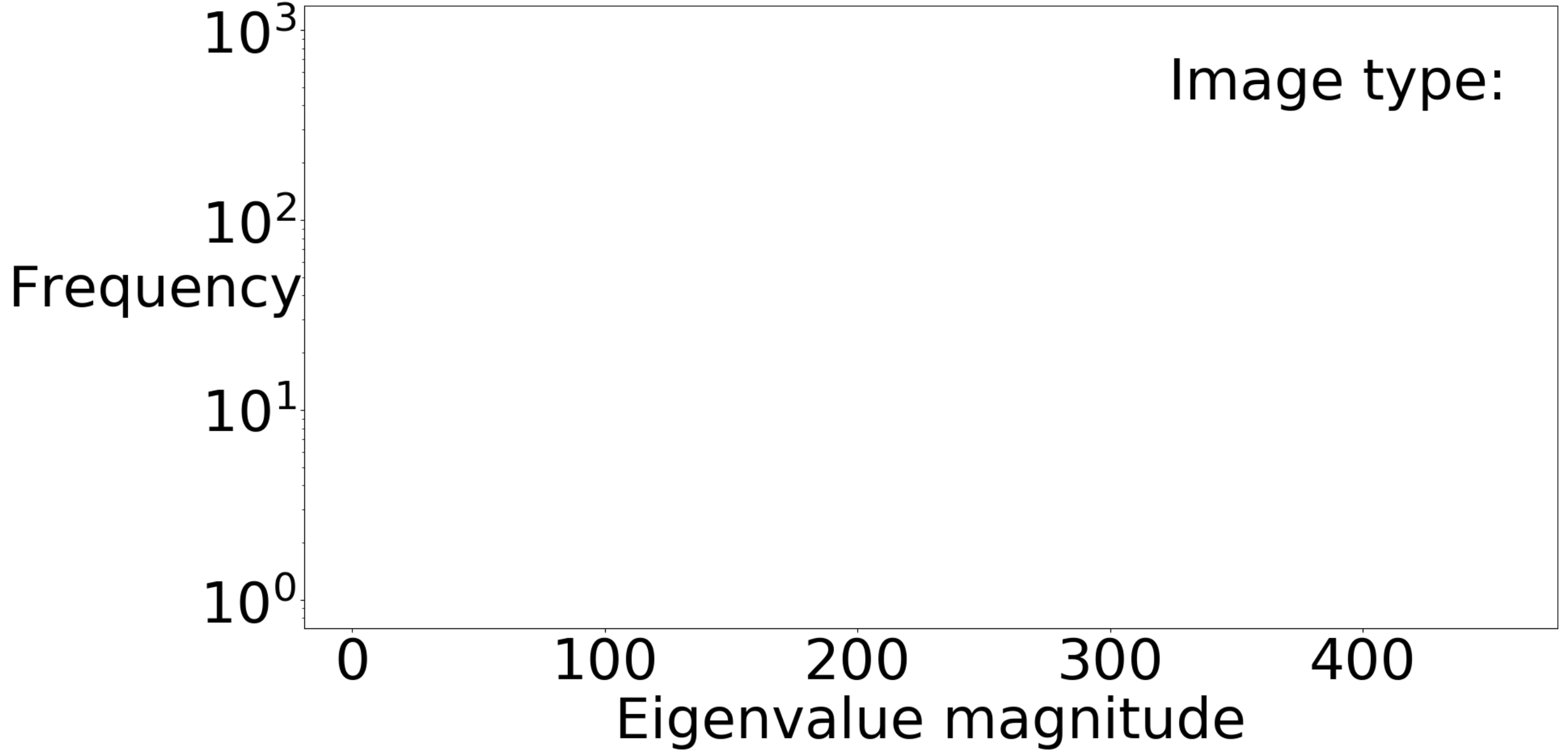
Gradient-based analysis



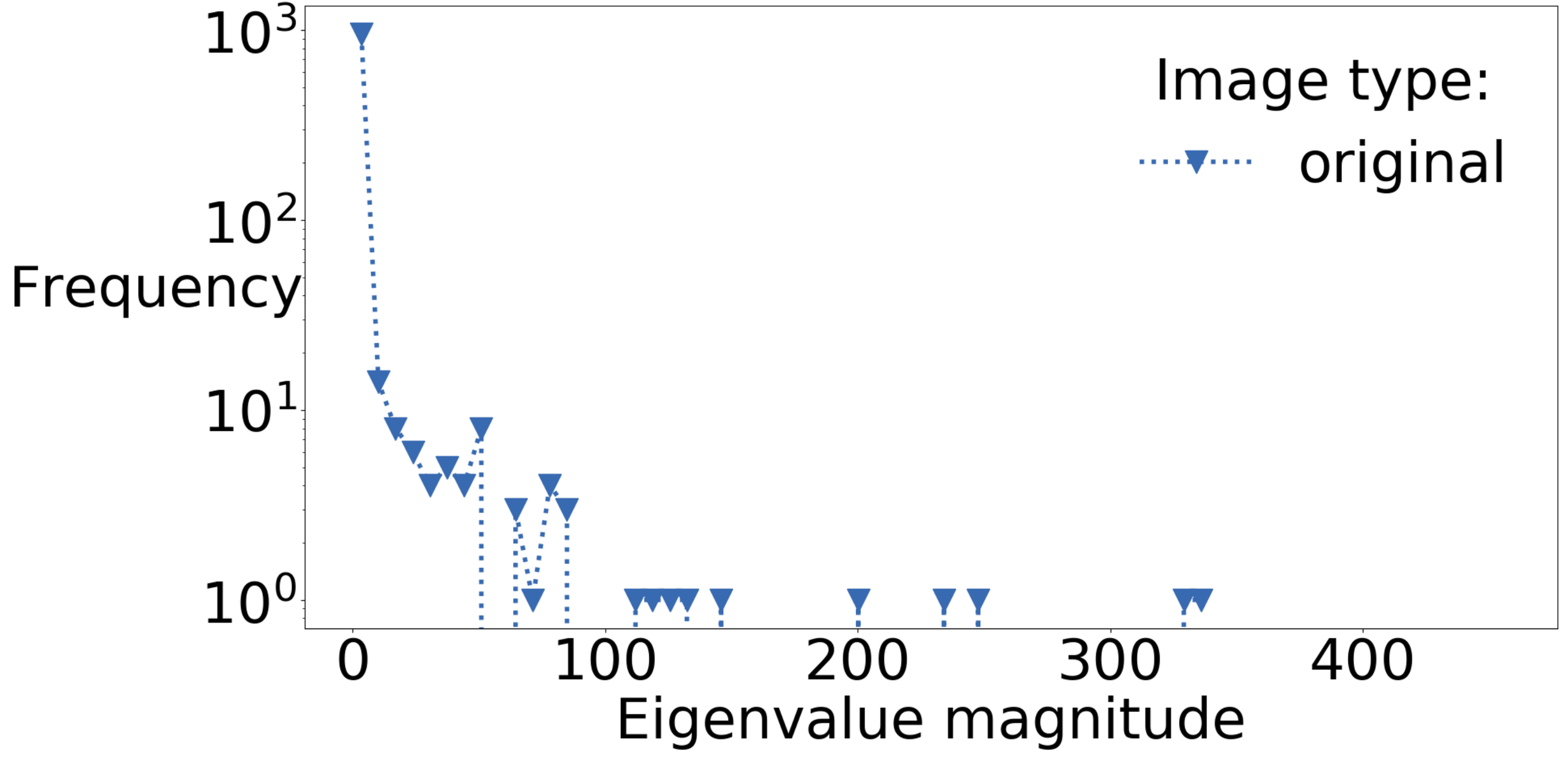
Gradient-based analysis



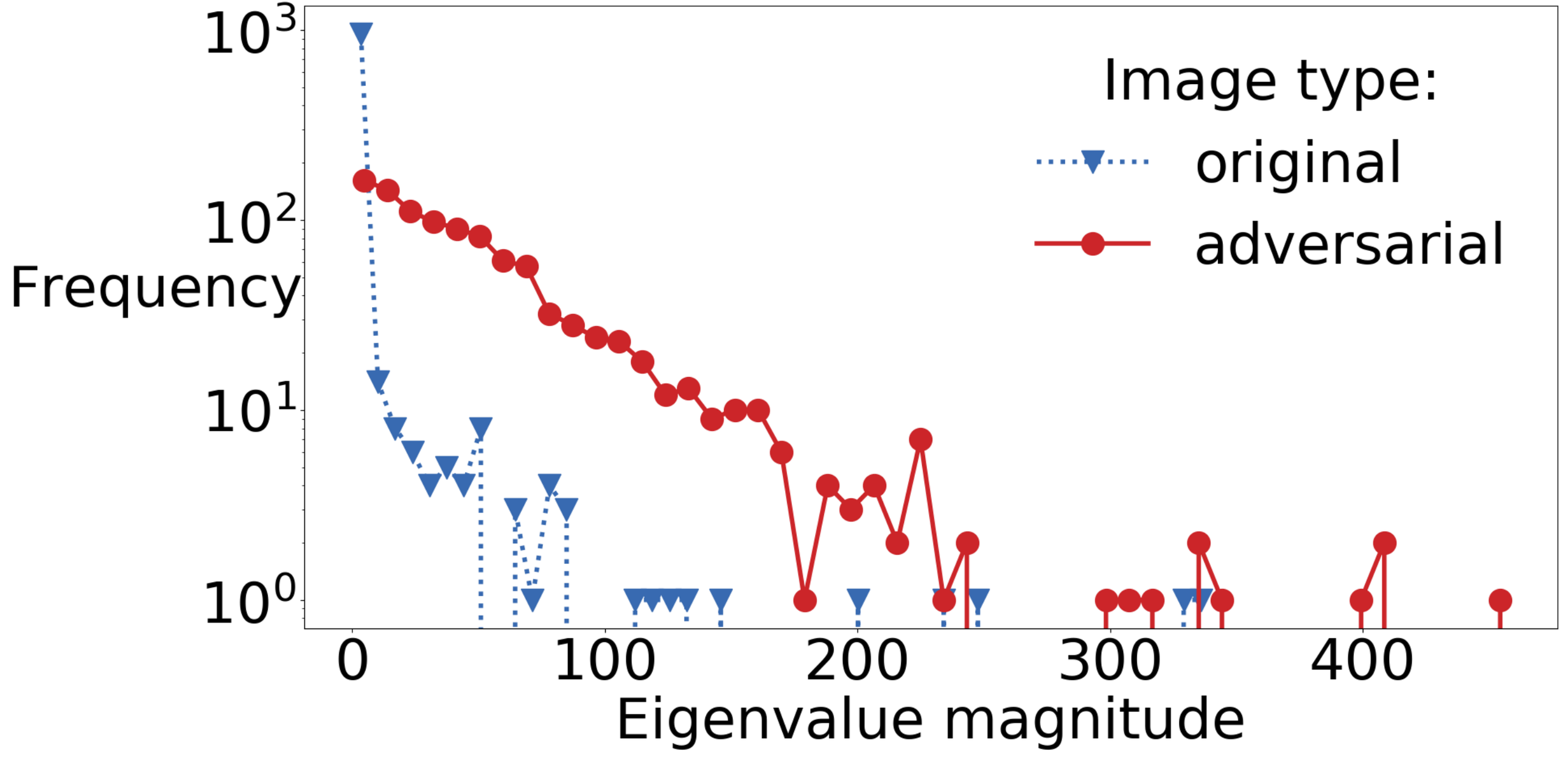
Hessian-based analysis



Hessian-based analysis



Hessian-based analysis



Main take-aways

Adaptive & Robust ConvNets

Adaptive & Robust ConvNets

- FFT-based convolution:
  - Fastest for **big filters**.
  - Apply compression to effectively **control resource usage** (GPU/CPU and memory).

Adaptive & Robust ConvNets

- FFT-based convolution:
  - Fastest for **big filters**.
  - Apply compression to effectively **control resource usage** (GPU/CPU and memory).
- Low frequency coefficients learned first during training.

Adaptive & Robust ConvNets

- FFT-based convolution:
  - Fastest for **big filters**.
  - Apply compression to effectively **control resource usage** (GPU/CPU and memory).
- Low frequency coefficients learned first during training.
- The more band-limited model, the more robust to attacks.

Adaptive & Robust ConvNets

- FFT-based convolution:
  - Fastest for **big filters**.
  - Apply compression to effectively **control resource usage** (GPU/CPU and memory).
- Low frequency coefficients learned first during training.
- The more band-limited model, the more robust to attacks.
- Adversarial examples are not robust to perturbations.

Adaptive & Robust ConvNets

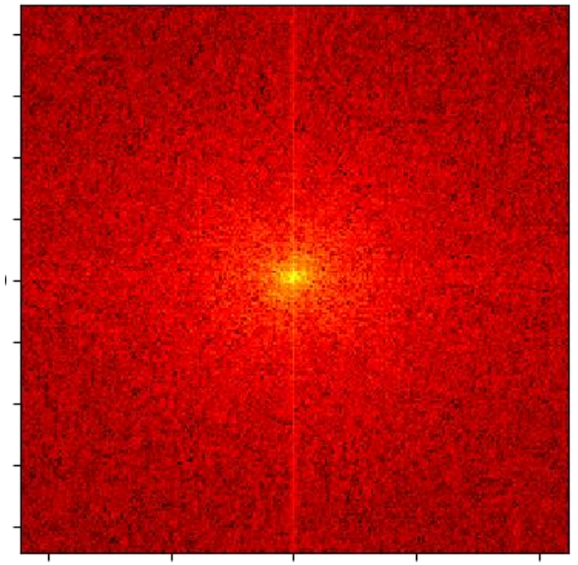
- FFT-based convolution:
  - Fastest for **big filters**.
  - Apply compression to effectively **control resource usage** (GPU/CPU and memory).
- Low frequency coefficients learned first during training.
- The more band-limited model, the more robust to attacks.
- Adversarial examples are not robust to perturbations.
- Different perturbation defenses provide similar gains in robustness.

1. Introduction & FFT-based convolution

2. Attacks and defenses

3. Research plan

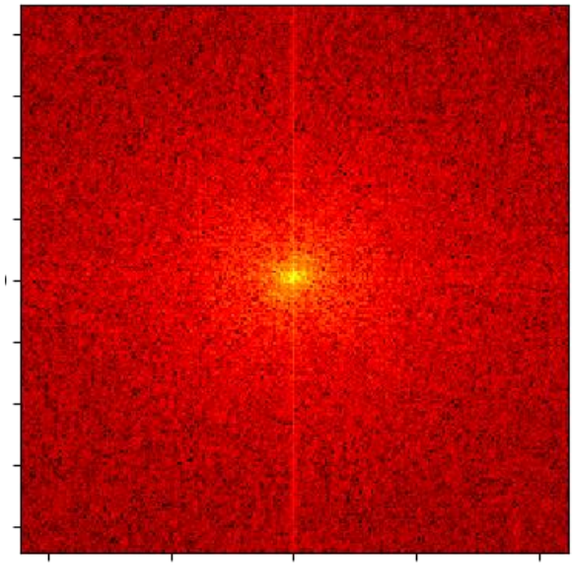
Different ways of encoding CNNs



Fourier domain:

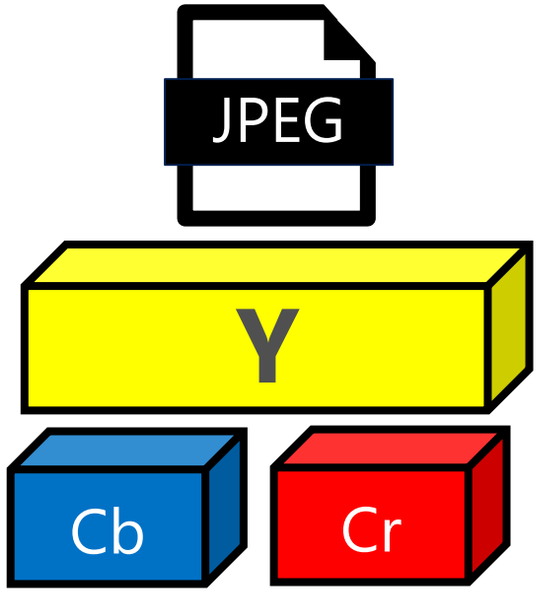
- ✚ Interpretable
- ✚ Compress: training & inference
- ━ Only for big filters

Different ways of encoding CNNs



Fourier domain:

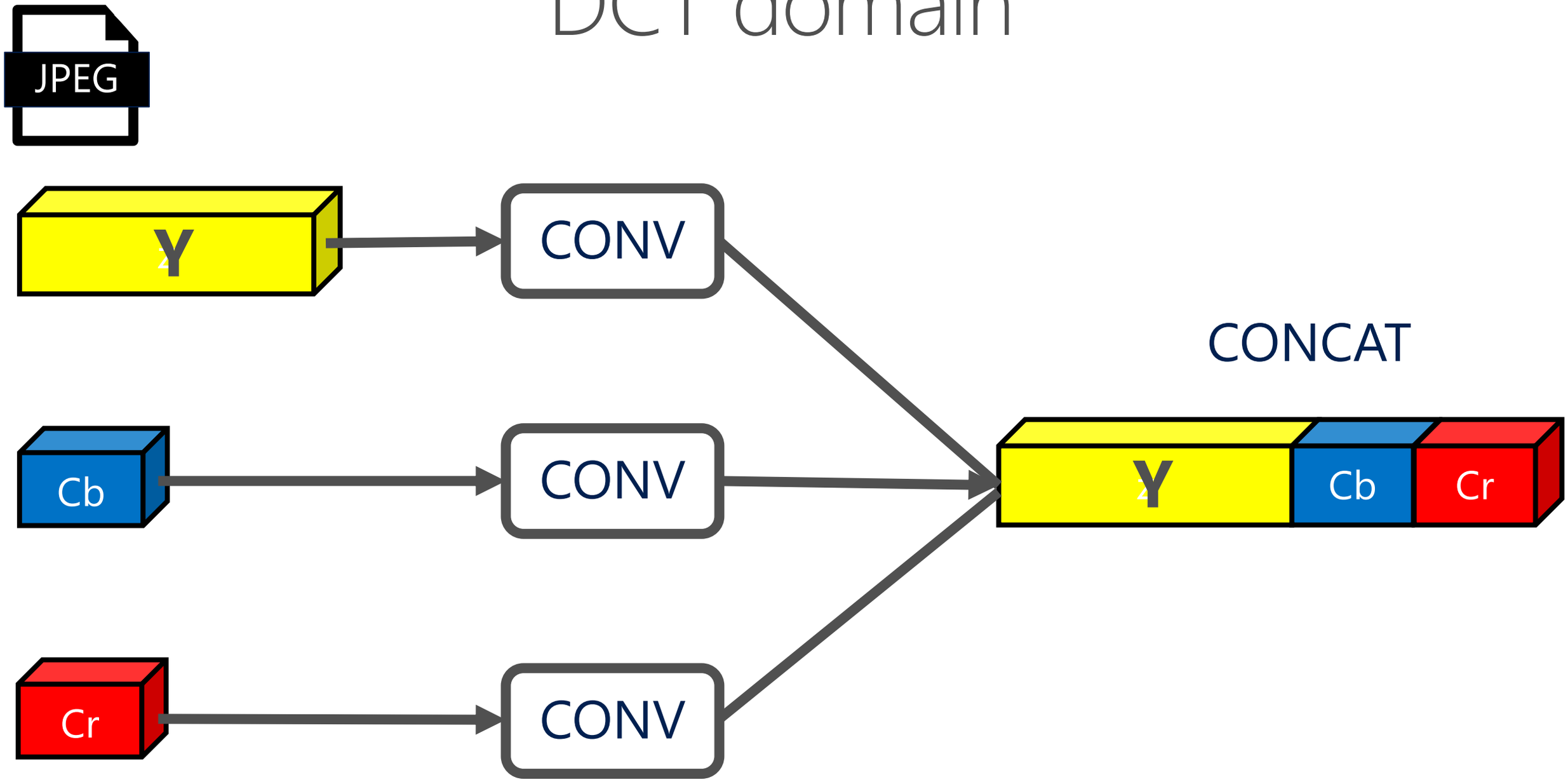
- ✚ Interpretable
- ✚ Compress: training & inference
- ▬ Only for big filters



Cosine domain:

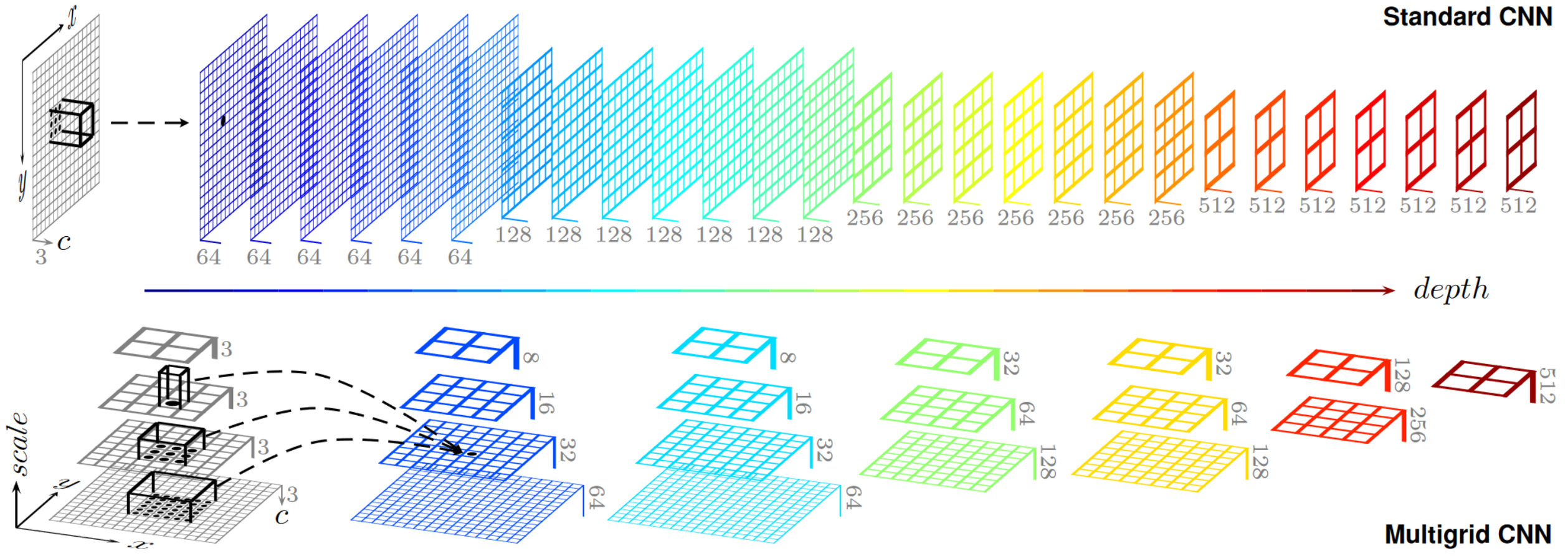
- ✚ Learned in the first conv layers
- ✚ Extract from JPEG
- ▬ Poor approximate convolution

DCT domain



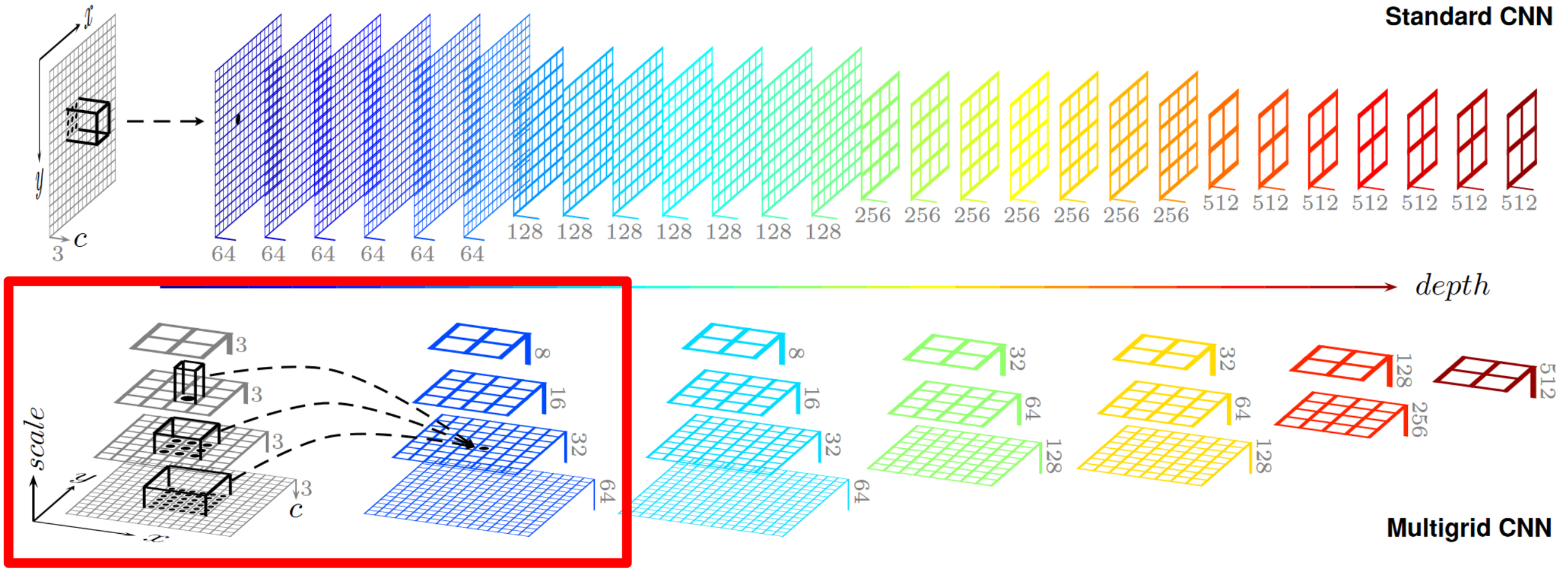
[8] Lionel G. et al. *Faster neural networks straight from jpeg*. NIPS, 2018.

Multi-grid CNNs



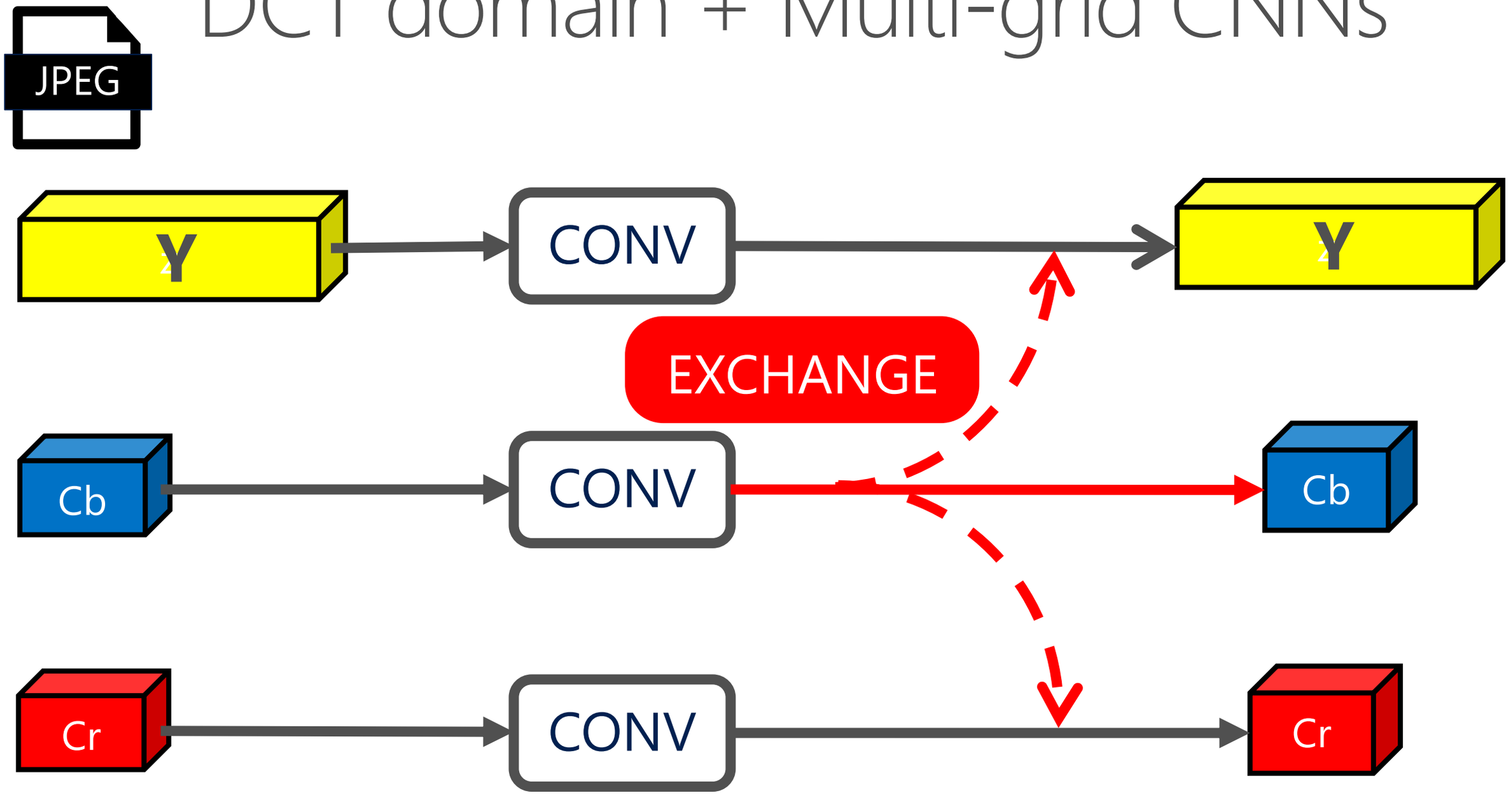
[9] Ke T. et al. Multigrid Neural Architectures. CVPR, 2017.

Multi-grid CNNs

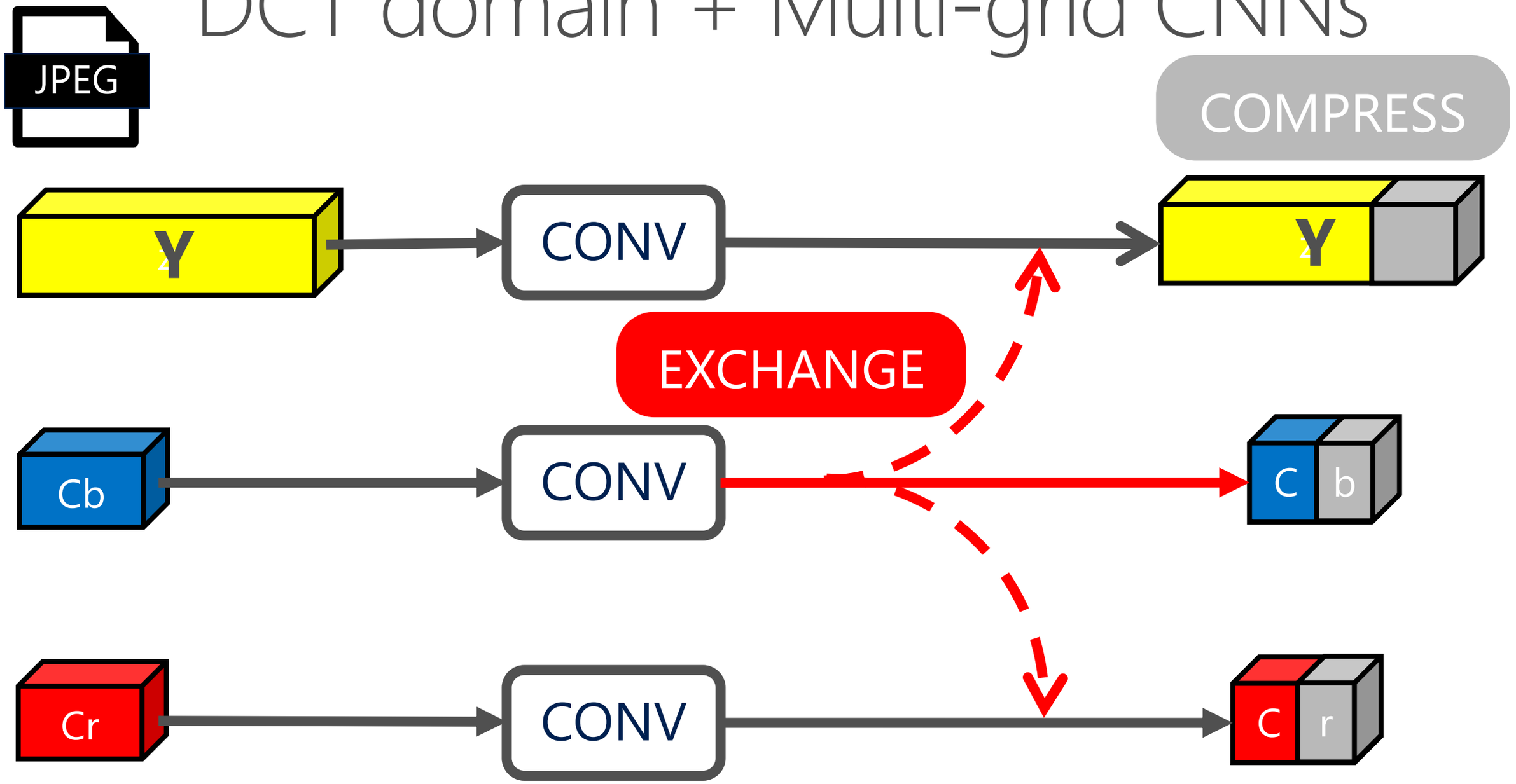


[9] Ke T. et al. Multigrid Neural Architectures. CVPR, 2017.

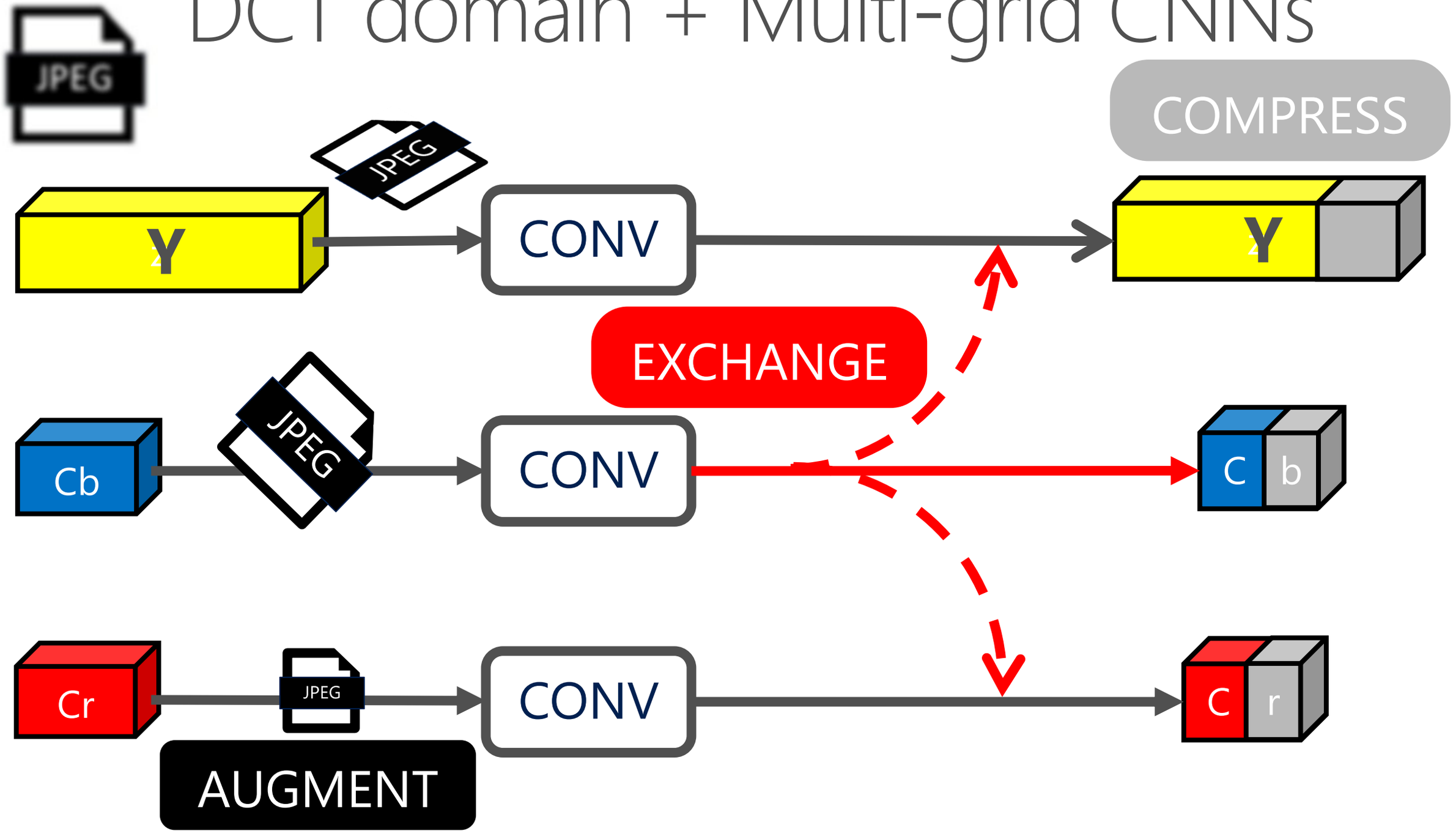
DCT domain + Multi-grid CNNs



DCT domain + Multi-grid CNNs



DCT domain + Multi-grid CNNs



Research plan

- Design and implement:
  - **Information exchange** between the Y, Cb, and Cr DCT blocks
  - **Compression** of internal representations in the DCT domain
  - **Data augmentation** (rotate, resize, crop, gaussian noise) in the DCT domain
- Find faster and/or more accurate:
  - Processing paths (conv blocks, pooling, etc.) of the Y, Cb, and Cr channels
  - **Size of the DCT blocks** (default is 8 x 8 pixels)
- Datasets & Base Architectures
  - CIFAR-10 on ResNet-18 & ImageNet on ResNet-50
- Ablation study: identify from where improvements arise

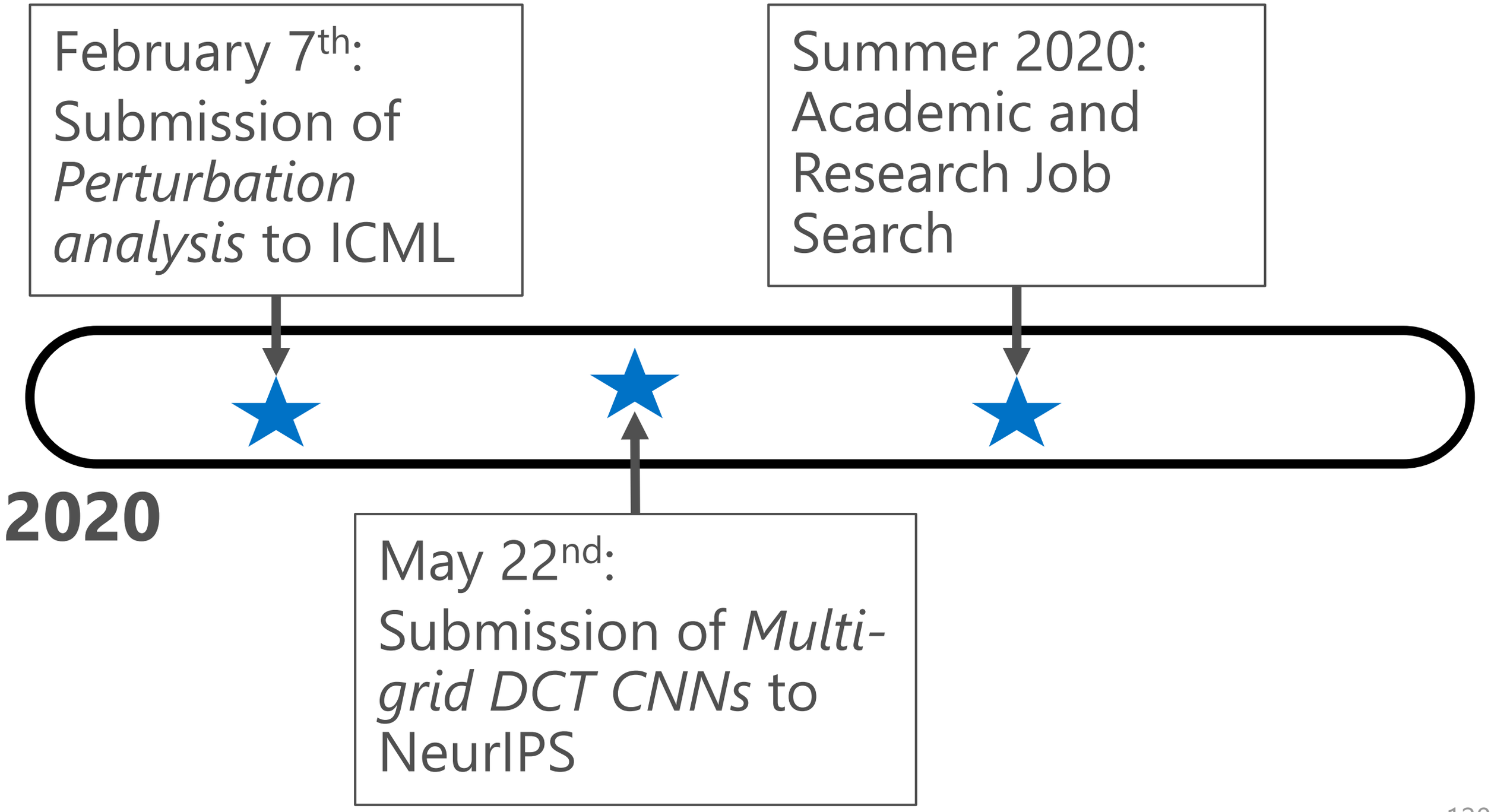
February 7th: Submission of *Perturbation analysis* to ICML

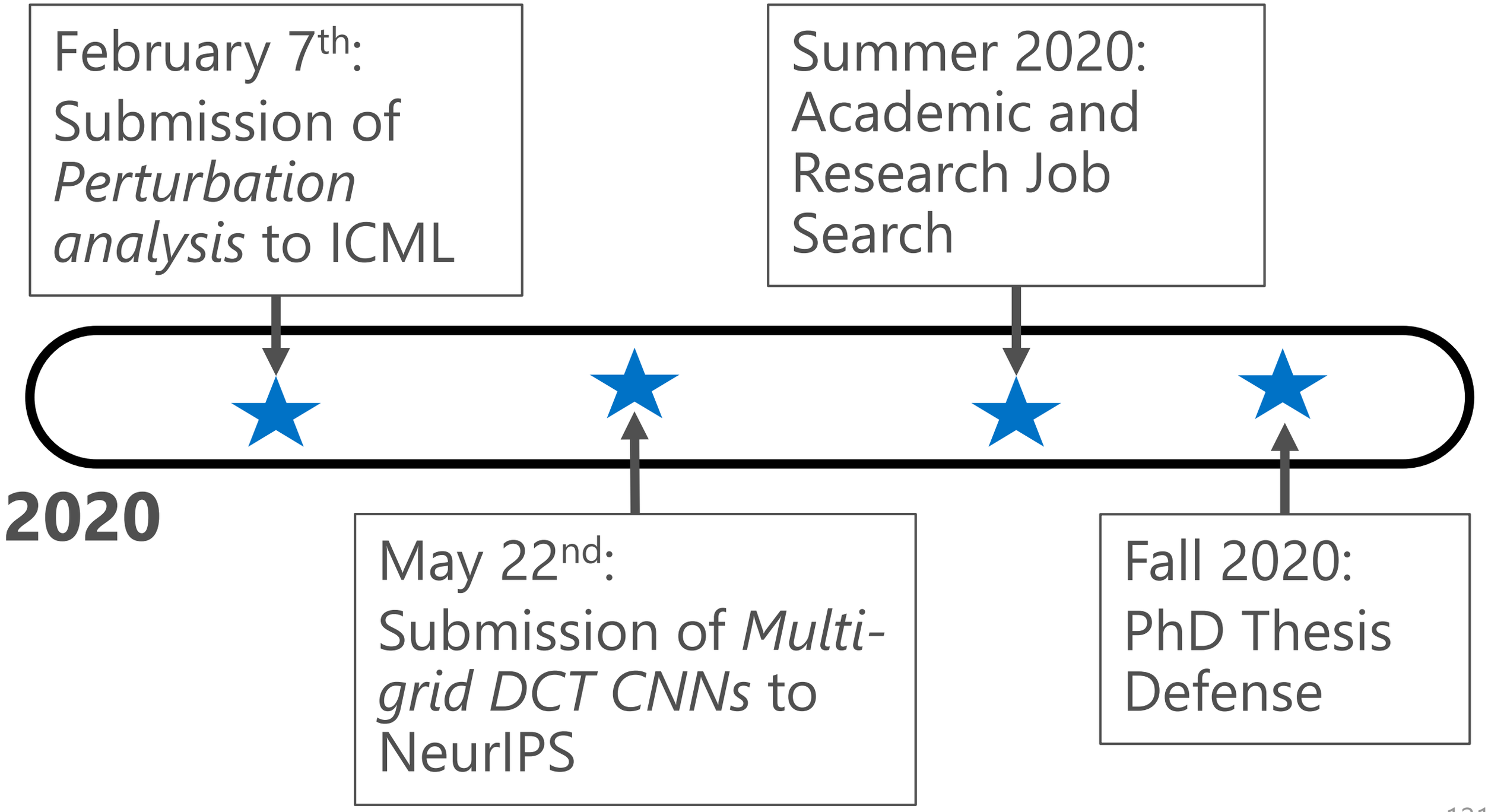
2020

February 7th: Submission of *Perturbation analysis* to ICML

2020

May 22nd: Submission of *Multi-grid DCT CNNs* to NeurIPS


February 7th: Submission of *Perturbation analysis* to ICML
Summer 2020: Academic and Research Job Search
2020
May 22nd: Submission of *Multi-grid DCT CNNs* to NeurIPS


February 7th: Submission of *Perturbation analysis* to ICML
Summer 2020: Academic and Research Job Search
2020
May 22nd: Submission of *Multi-grid DCT CNNs* to NeurIPS
Fall 2020: PhD Thesis Defense

Thank you

github.com/adam-dziedzic/bandlimited-cnns

ady@uchicago.edu

Bibliography

[1] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. ICLR, 2016.
[2] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. NIPS, 2019
[3] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. arXiv:1710.03740, 2017
[4]Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. NIPS, 2018.
[5] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. arXiv:1608.00853, 2016.
[6] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. NDSS, 2018.
[7] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. NIPS, 2019.

Bibliography

[8] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. NIPS, 2018.
[9] Tsung-Wei Ke, Michael Maire, Stella X. Yu. Multigrid Neural Architectures. CVPR, 2017.

Why is FFT based convolution important?

- The theoretical properties of the Fourier domain are well understood. No such properties in other domains (Winograd).
- ResNet and DenseNet architectures use 7x7 filters in first layers.
- FFT based convolution can be combined with spectral pooling.
- Band-limiting minimize aliasing & serves as a simple defense.
- A standard algorithm included in popular frameworks (cuDNN).
- Gradient acts as a large filter in the backward pass.
- Zlateski et al. suggest using FFT based convolution on CPUs.
- The 1D FFT convolution for DSP where large filters are used.

Band-limited FFT based convolution *formally*

Cross-correlate input data and filter: $x *_c y$

$$F_x[\omega] = F(x[n]) \qquad F_y[\omega] = F(y[n])$$

$$x *_c y = F^{-1}(F_x[\omega] \odot F_y[\omega])$$

Spectrum of convolution: $S[\omega] = F_x[\omega] \odot F_y[\omega]$

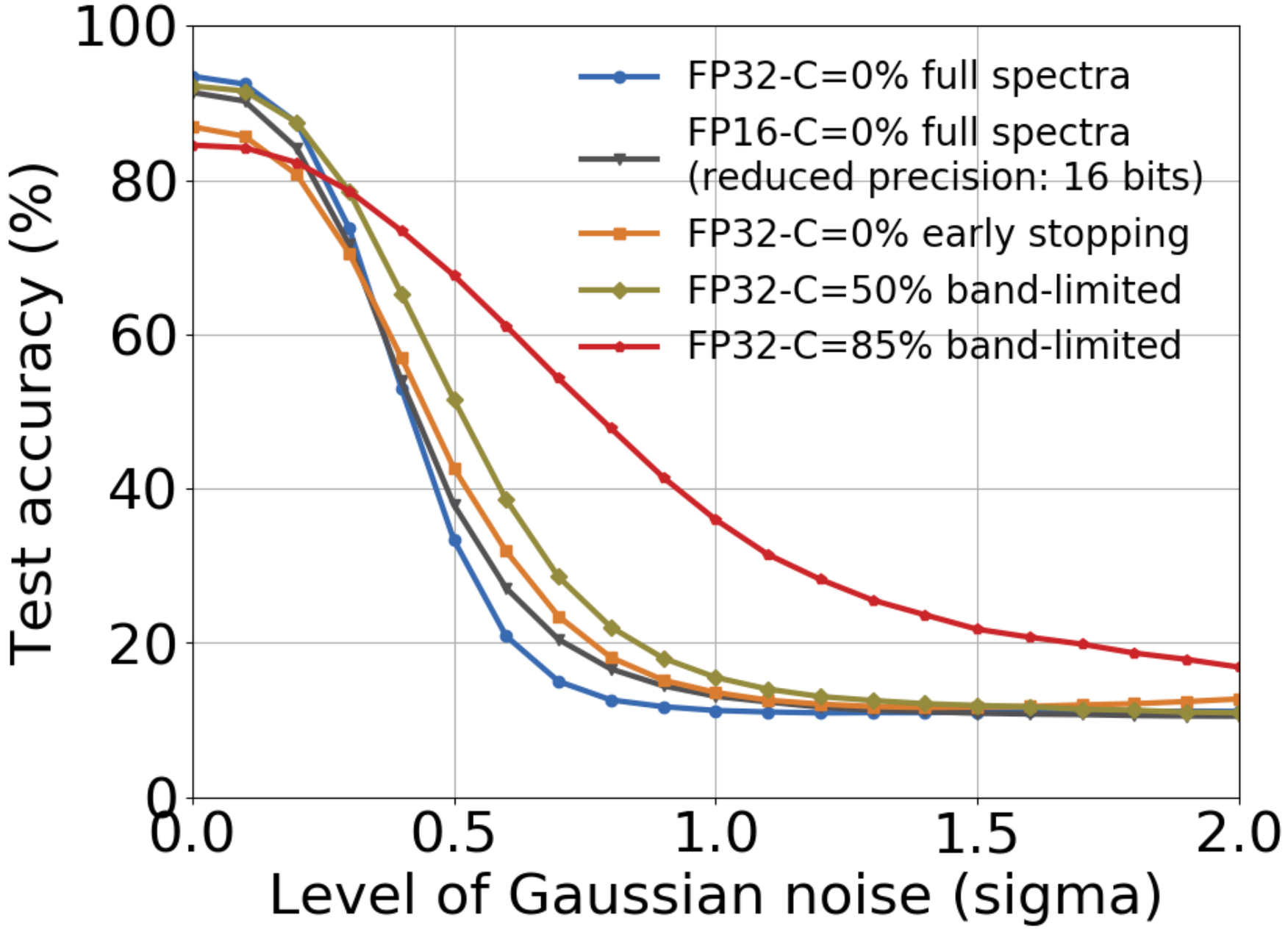
$$\mathbf{M_c}[\boldsymbol{\omega}] = \begin{cases} \mathbf{1}, \boldsymbol{\omega} \leq \mathbf{c} \\ \mathbf{0}, \boldsymbol{\omega} > \mathbf{c} \end{cases}$$

$$x *_c y = F^{-1}[(F_x[\omega] \odot M_c[\omega]) \odot (F_y[\omega] \odot M_c[\omega])]$$

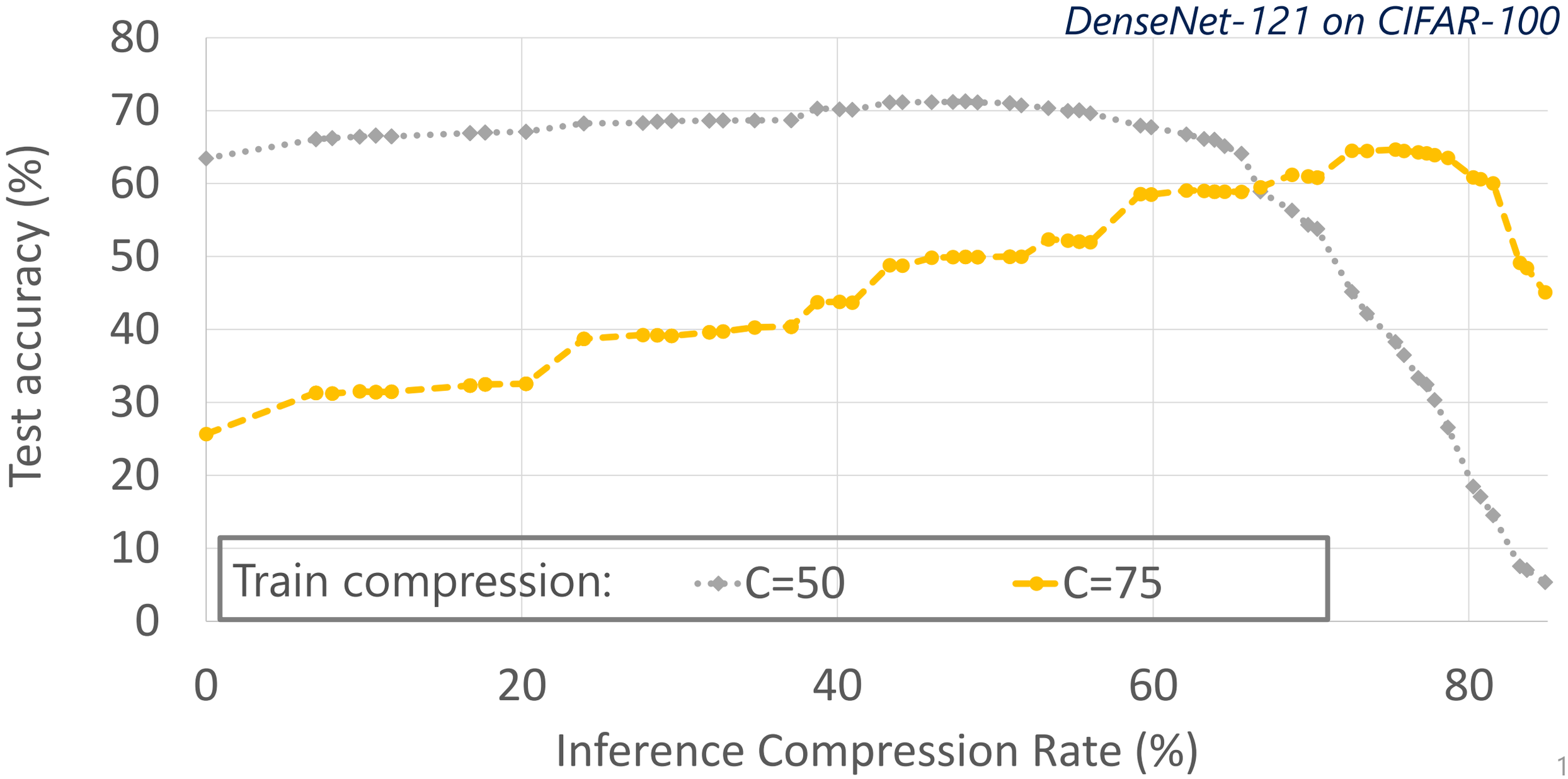
$$x *_c y = F^{-1}(S[\omega] \odot M_c[\omega])$$

Energy (Parseval's theorem): $\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{\omega=0}^{2\pi} |F_x(\omega)|^2$

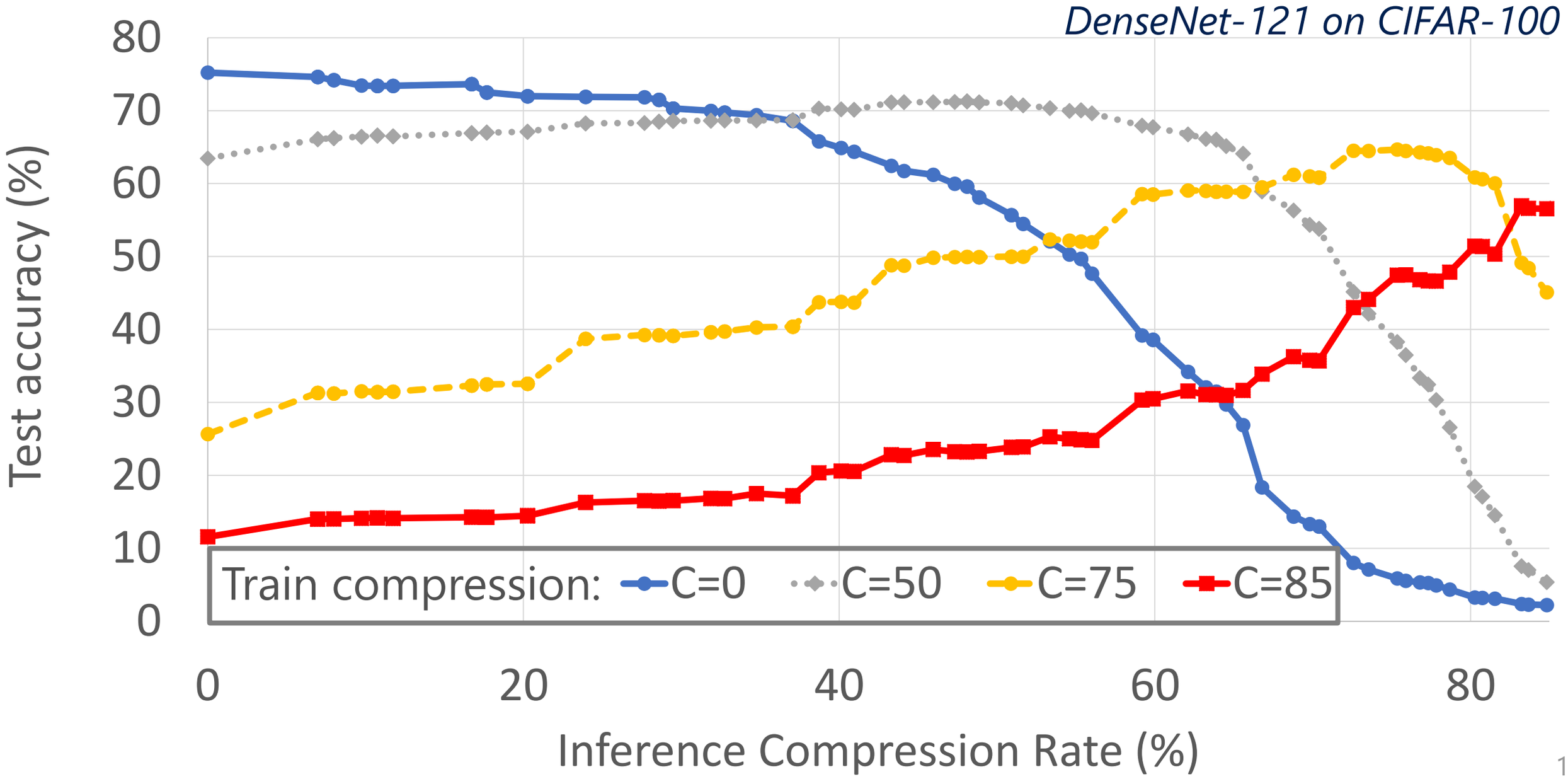
Robustness to noise



Compression Rate for Training vs Inference

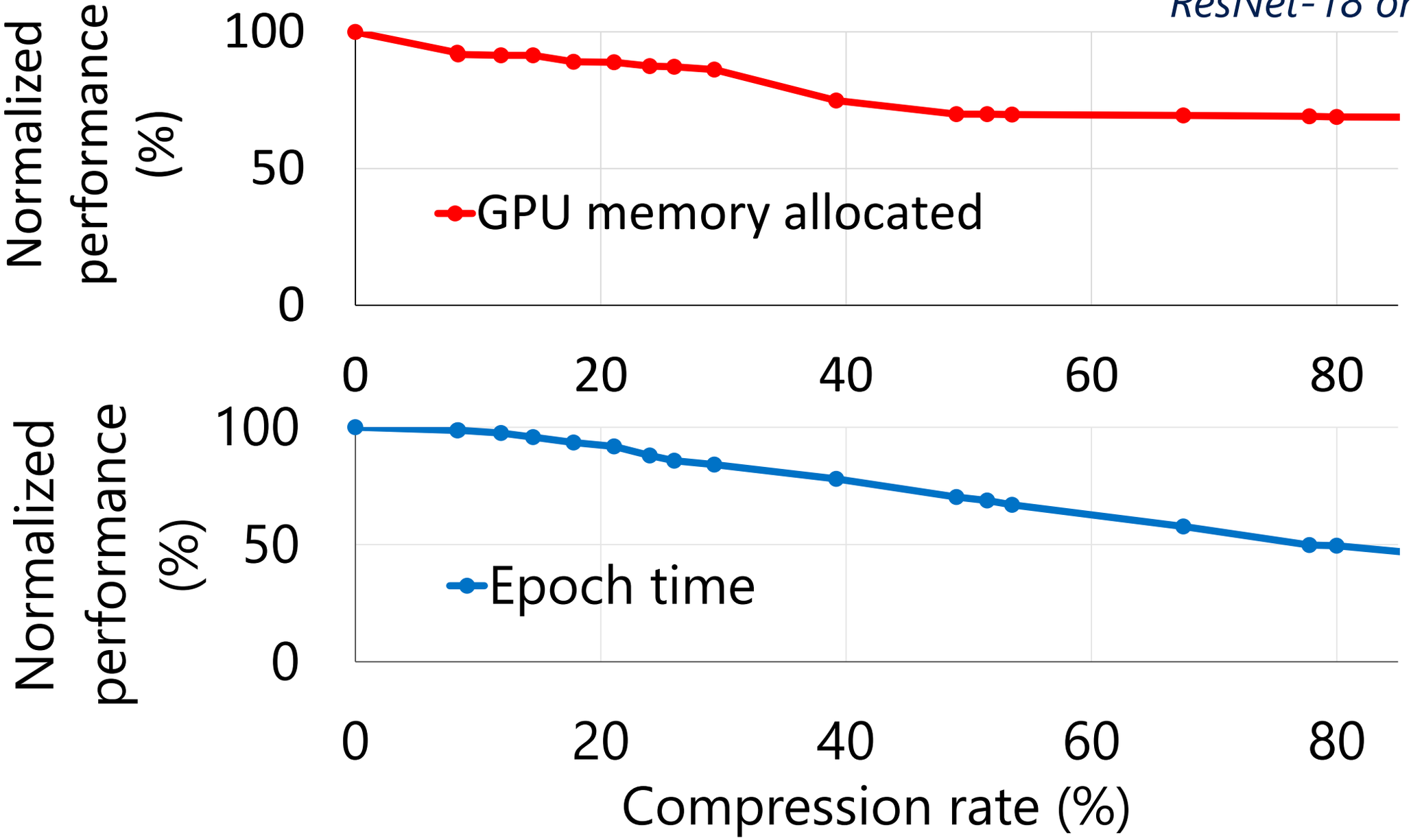


Compression Rate for Training vs Inference

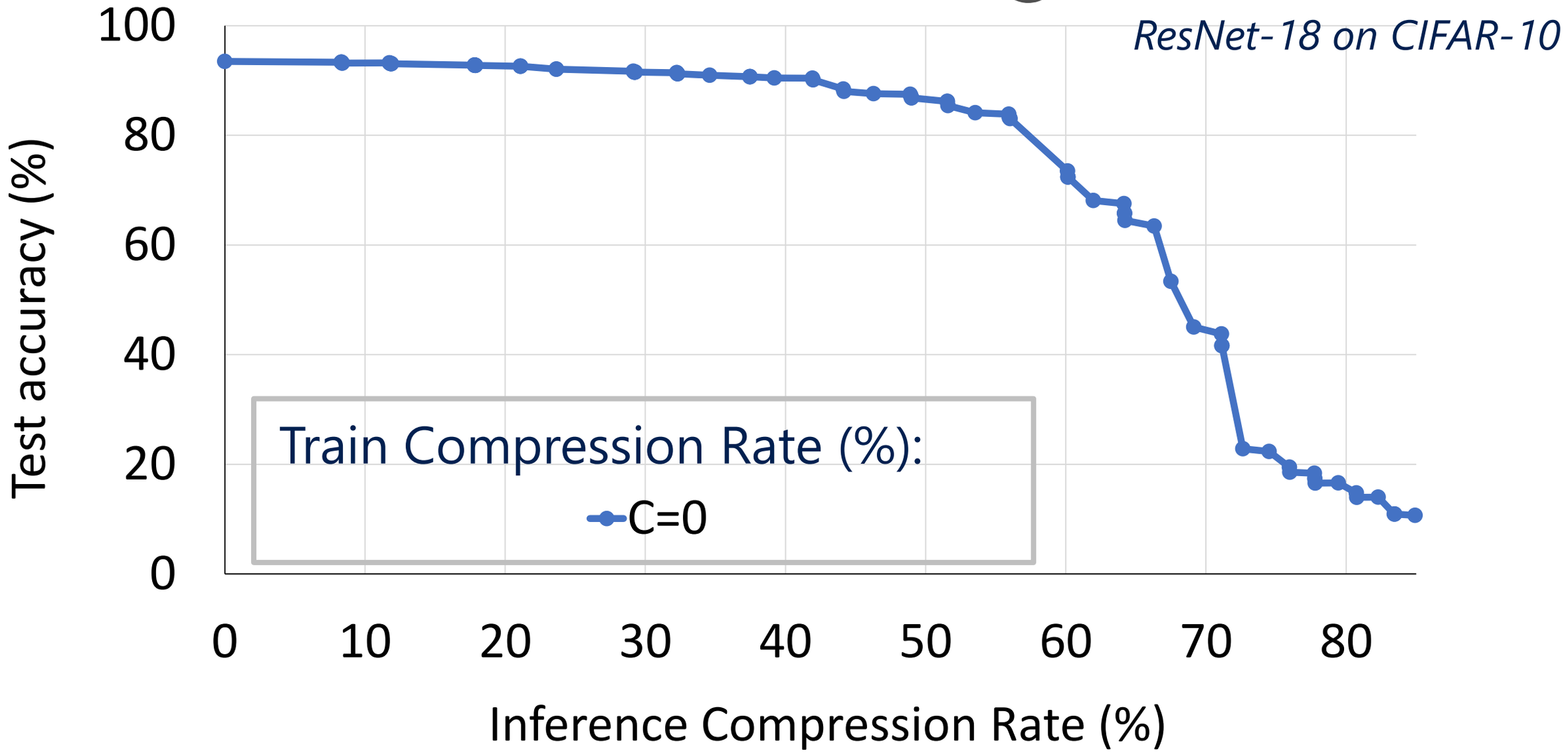


Effectively control resource usage

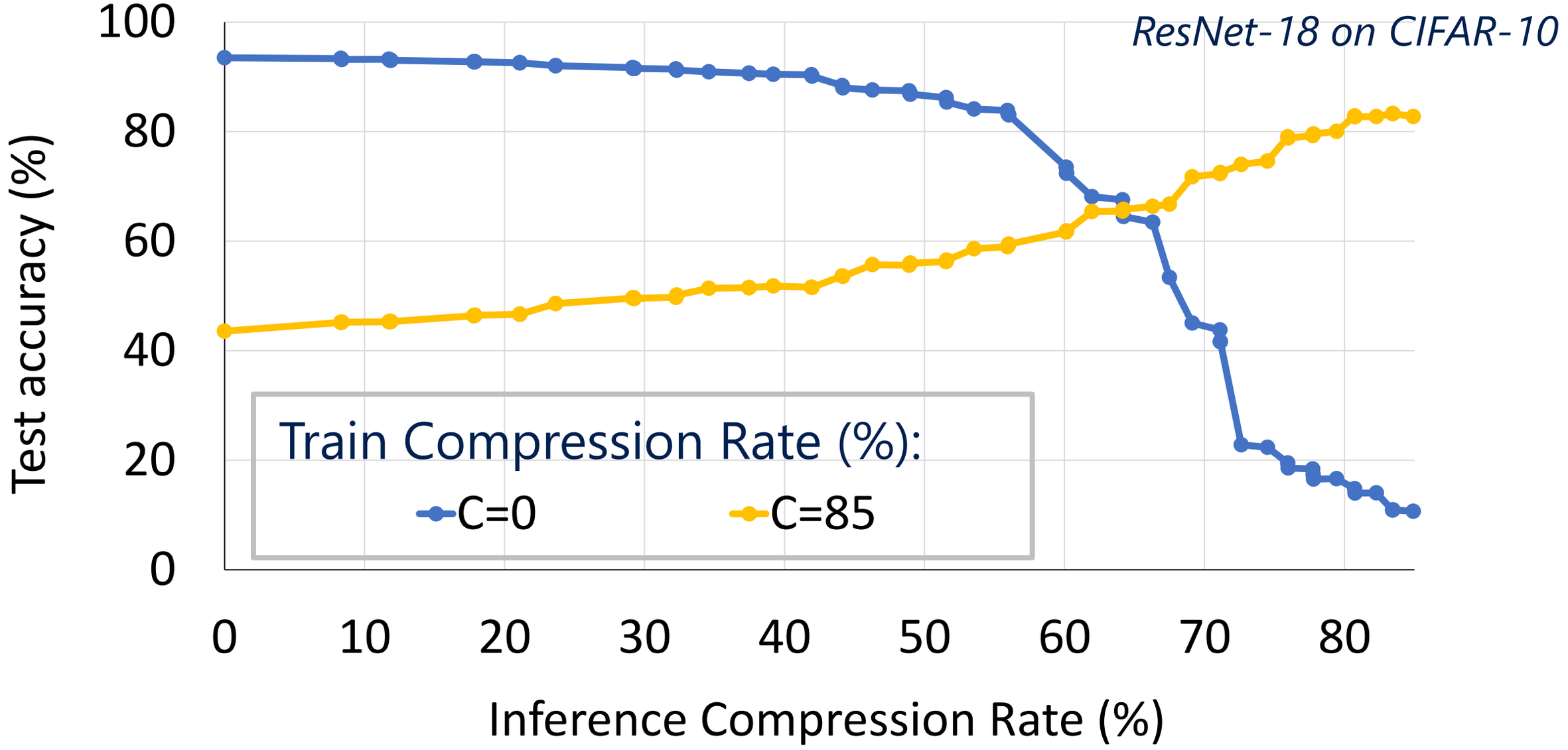
ResNet-18 on CIFAR-10



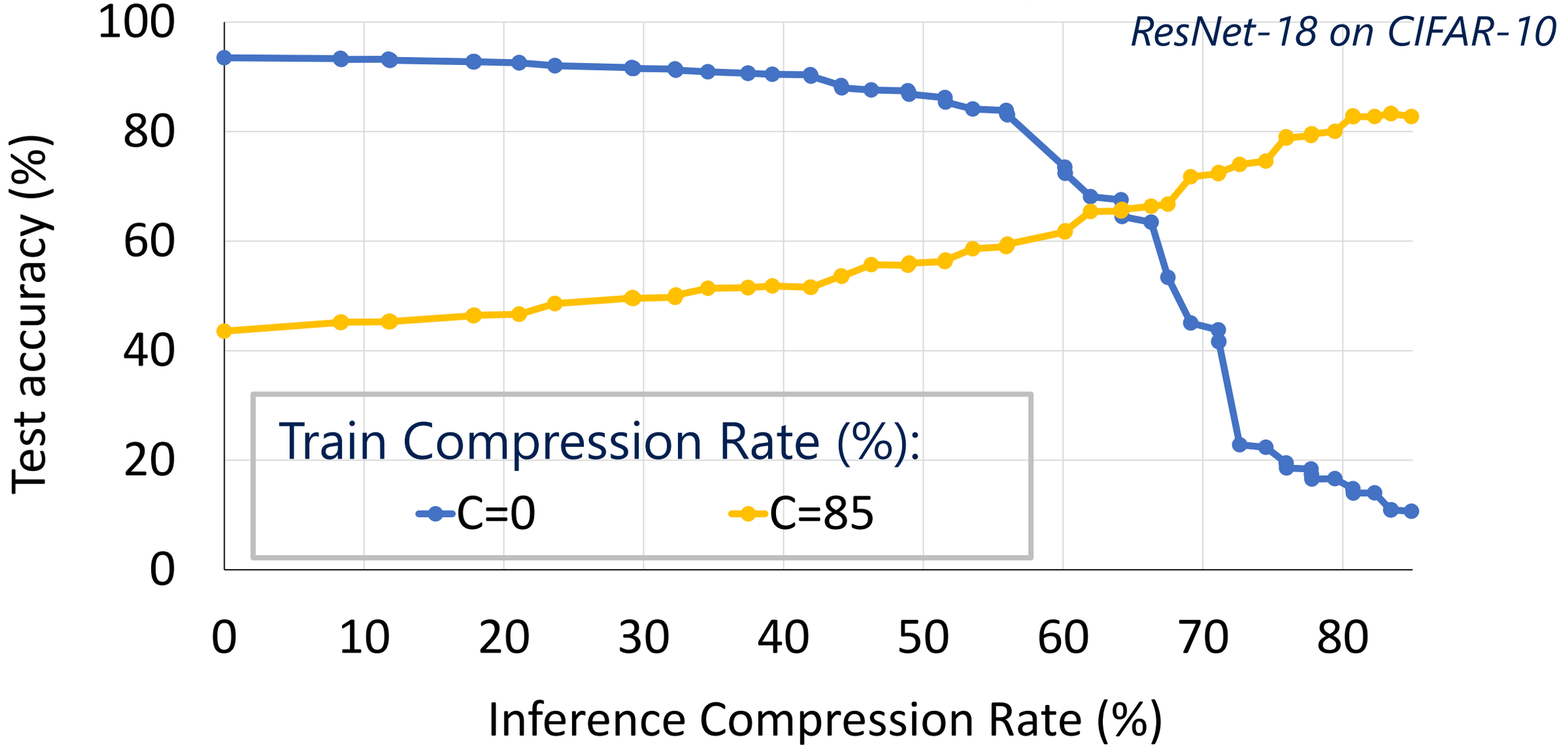
Compression Rate for Training vs Inference



Compression Rate for Training vs Inference

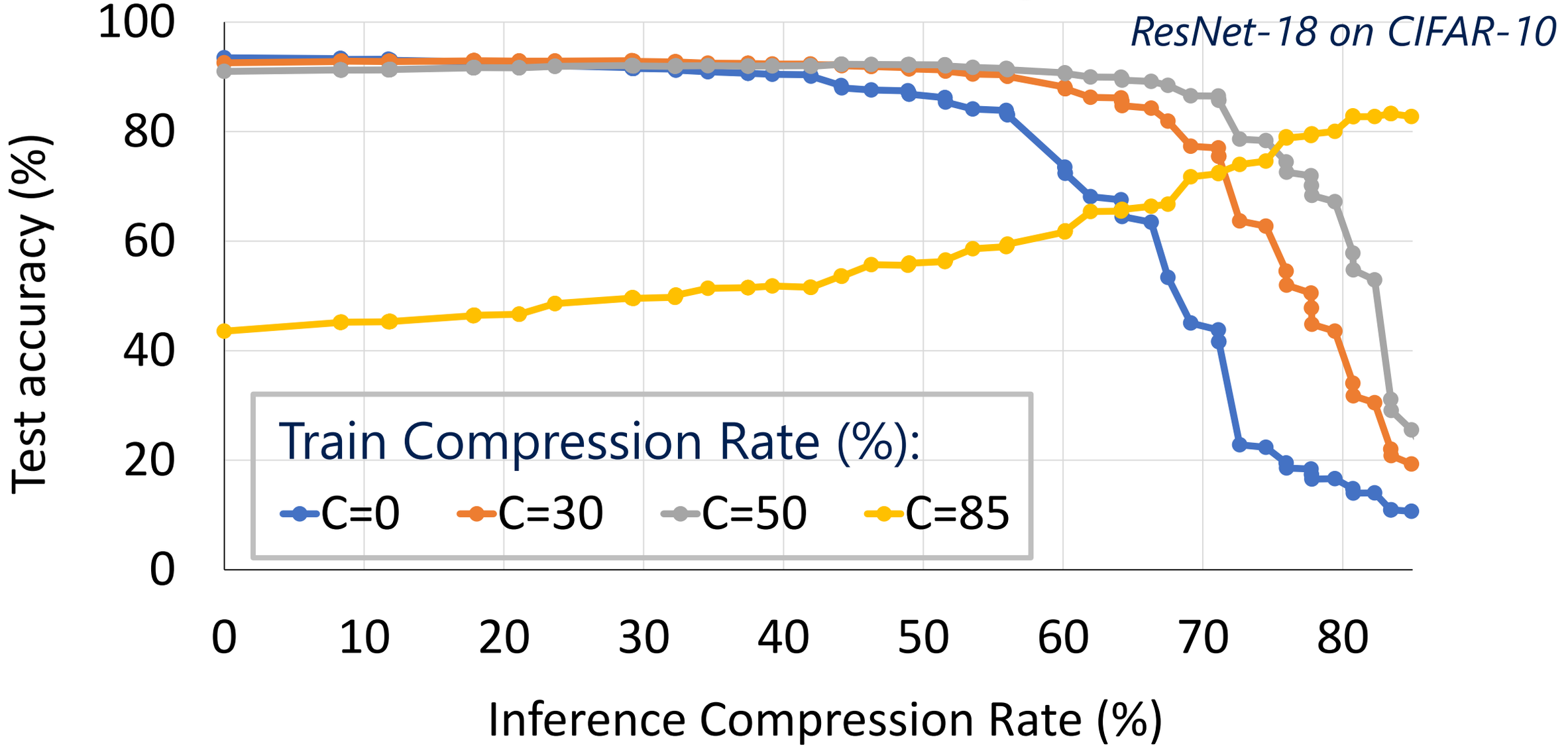


Compression Rate for Training vs Inference



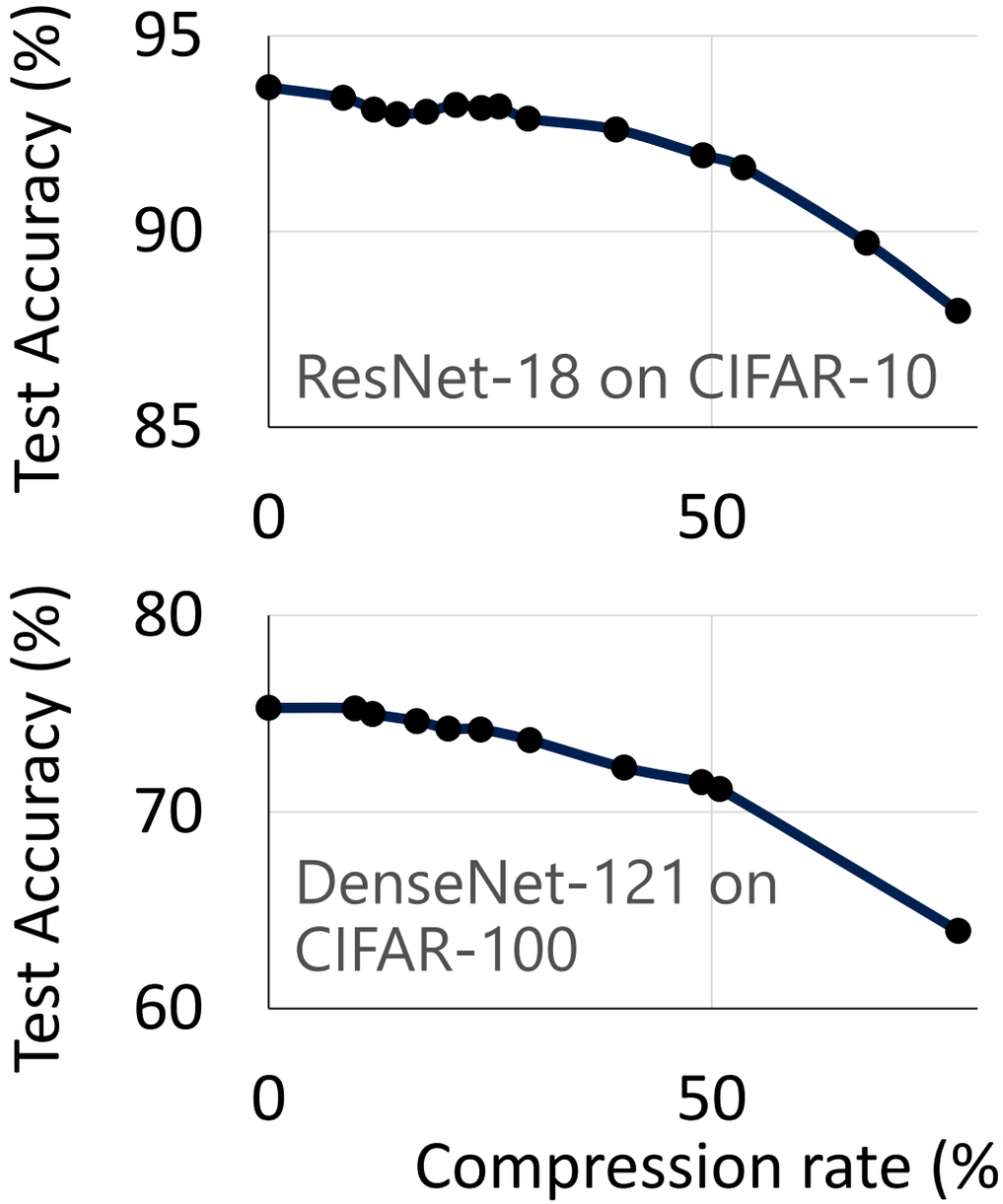
Smooth degradation of accuracy during inference

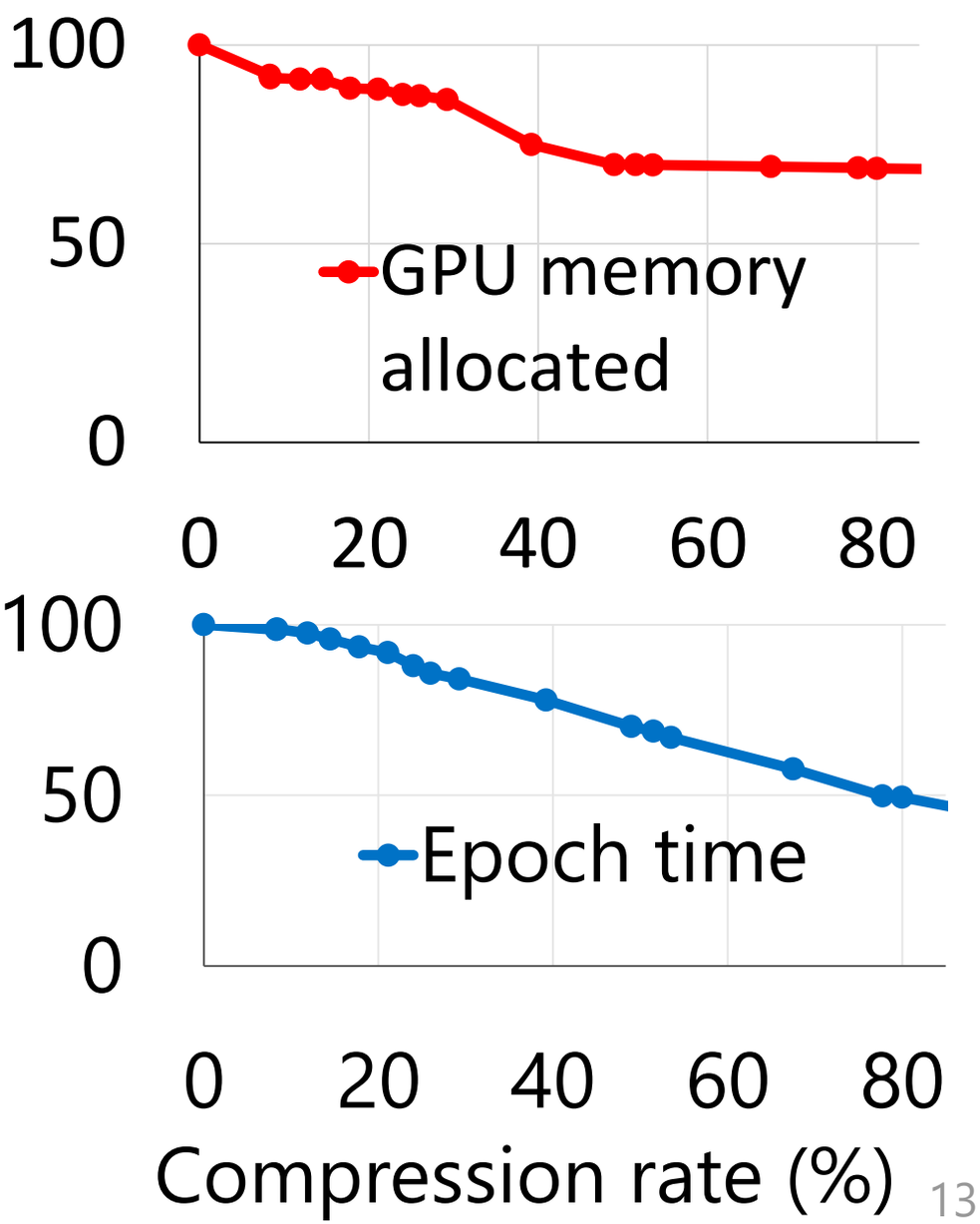
Compression Rate for Training vs Inference

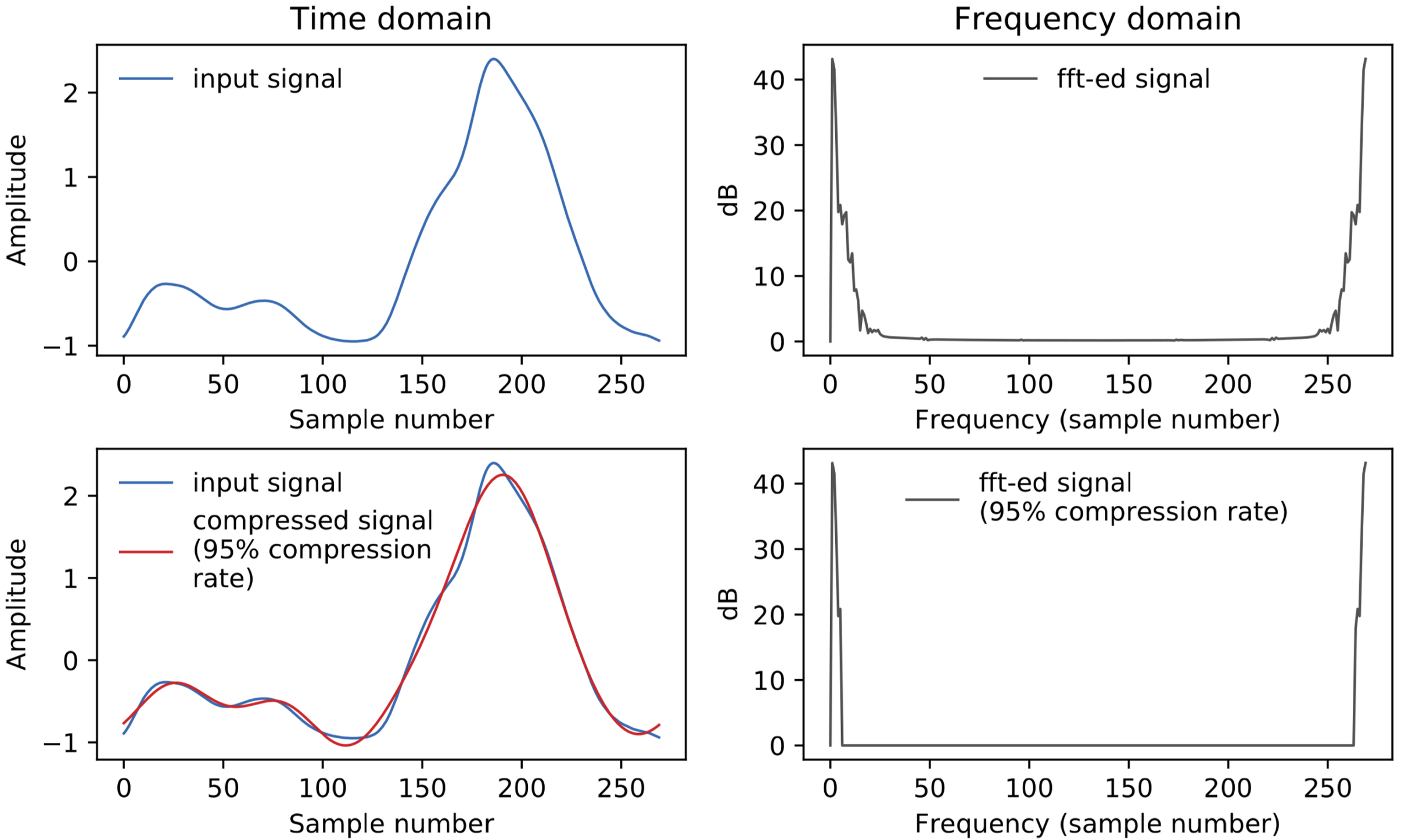


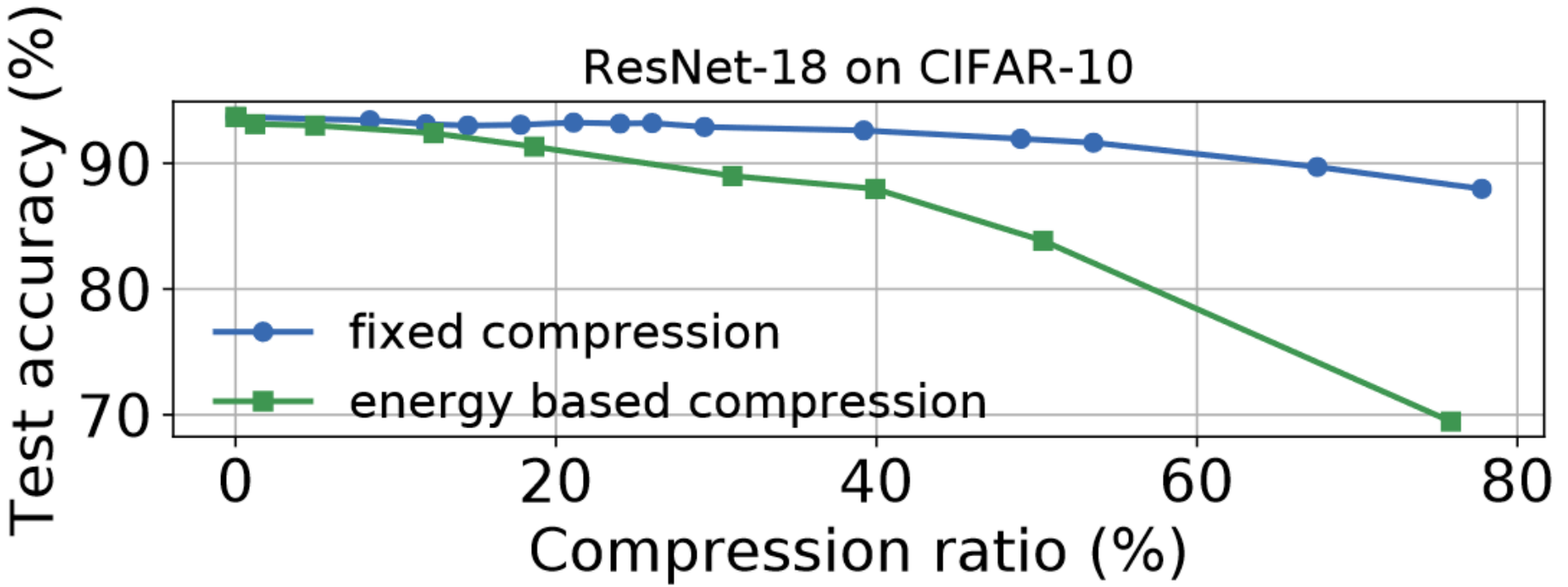
Apply the same compression rate to training and inference

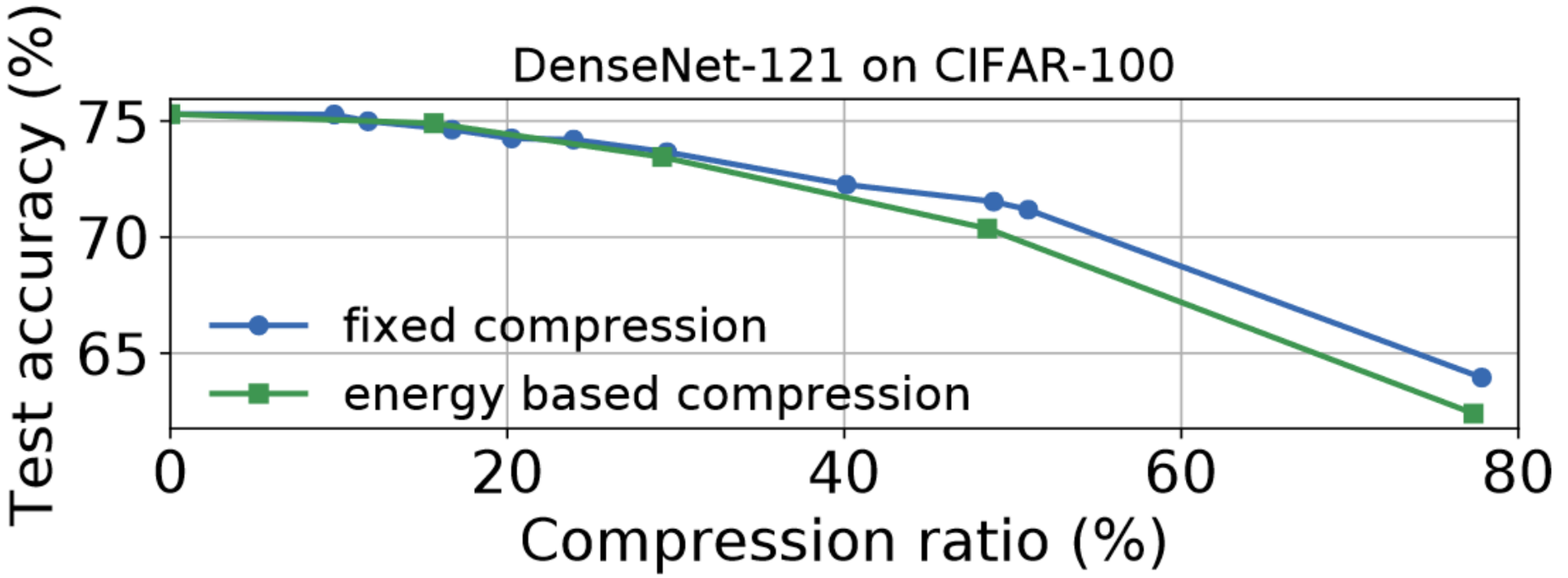
Tuning: Accuracy vs Higher Performance

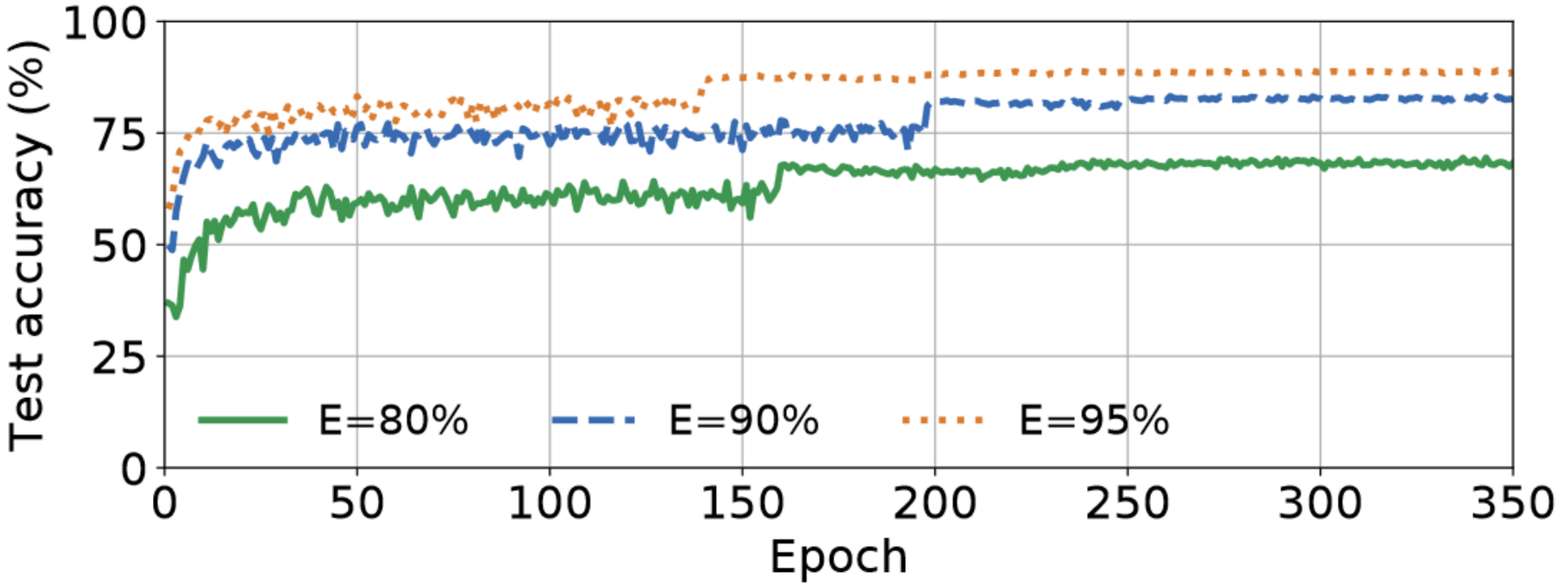


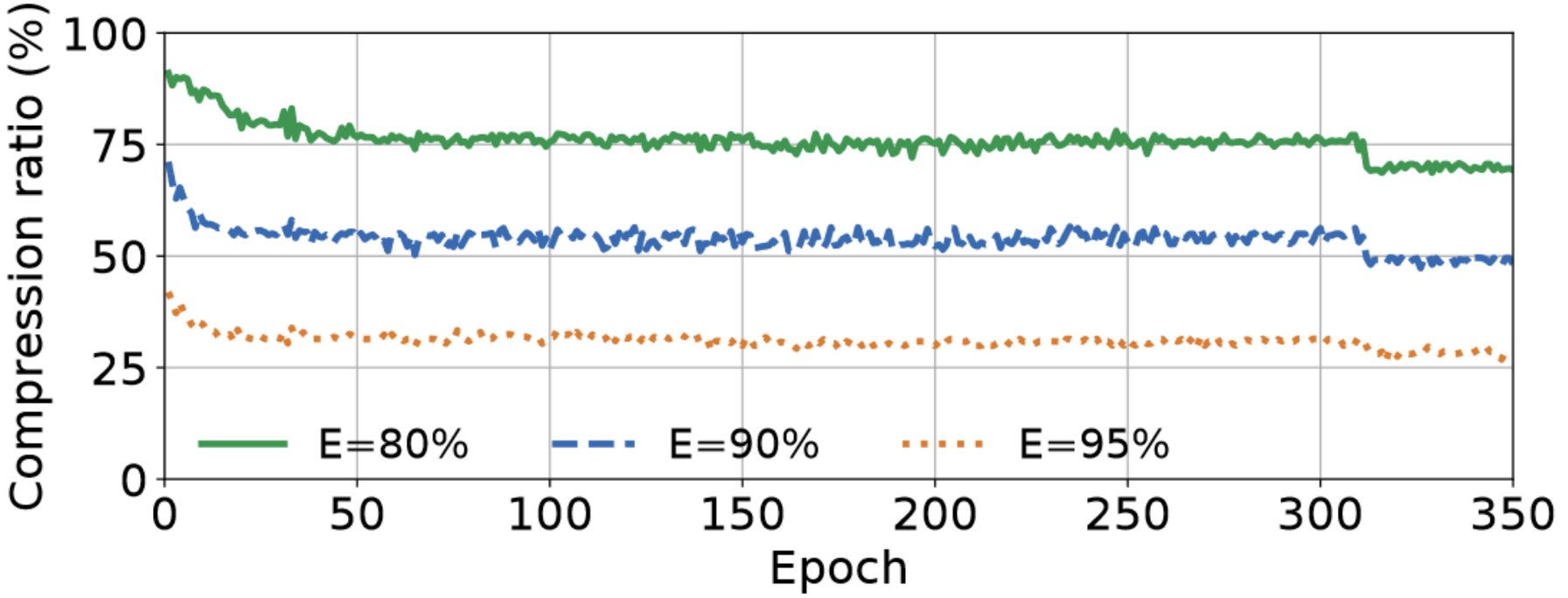


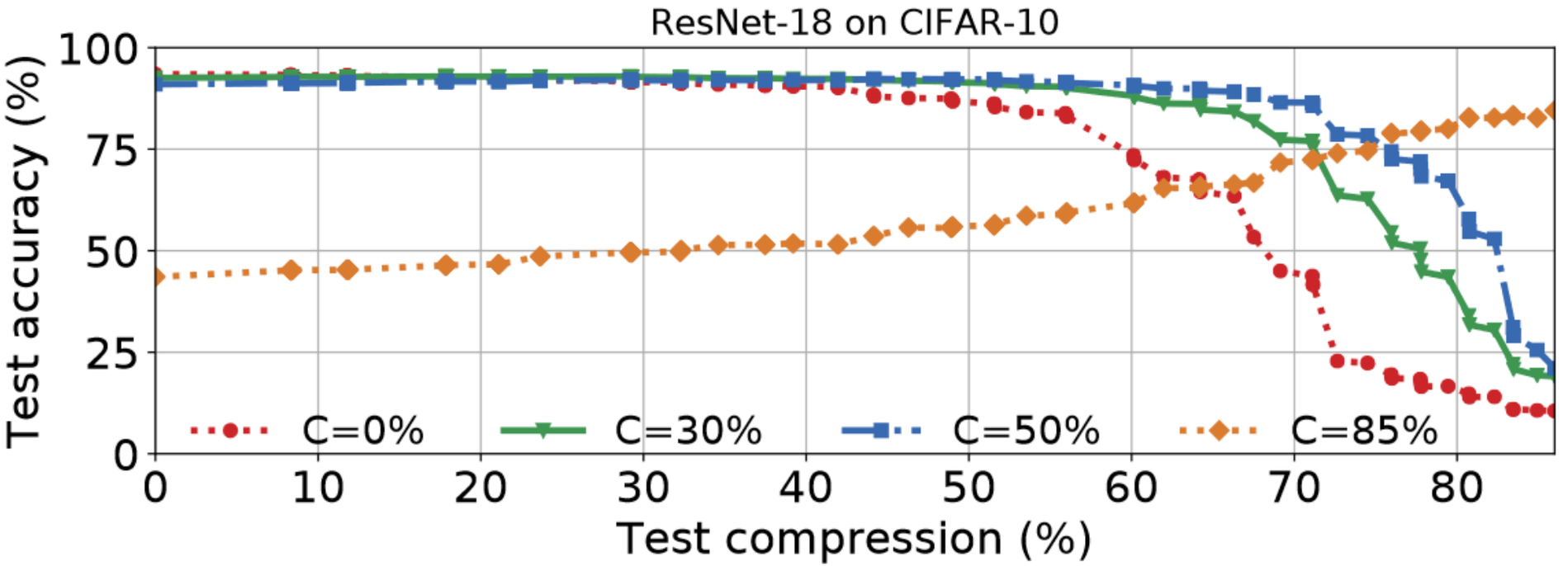

Time domain
Frequency domain
input signal
fft-ed signal
Amplitude
dB
Sample number
Frequency (sample number)
input signal
compressed signal
(95% compression
rate)
fft-ed signal
(95% compression rate)

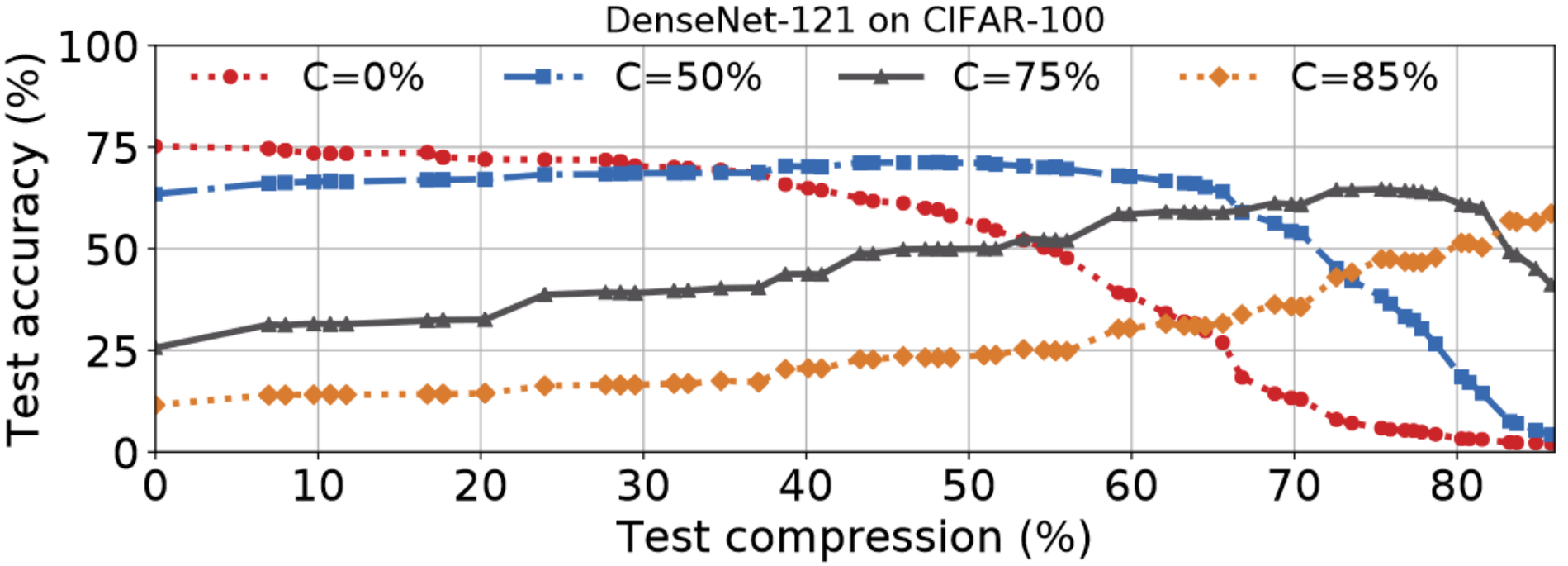

ResNet-18 on CIFAR-10
Test accuracy (%)
90
80
70
0
20
40
60
80
Compression ratio (%)
fixed compression
energy based compression

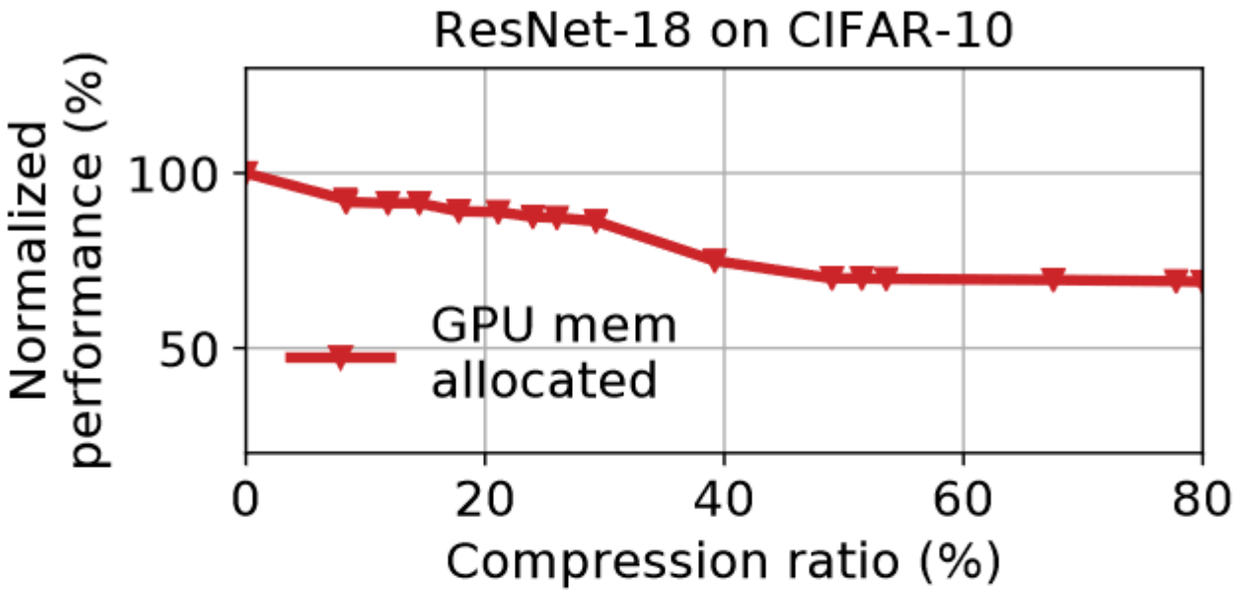

DenseNet-121 on CIFAR-100
Test accuracy (%)
75
70
65
0
20
40
60
80
Compression ratio (%)
fixed compression
energy based compression

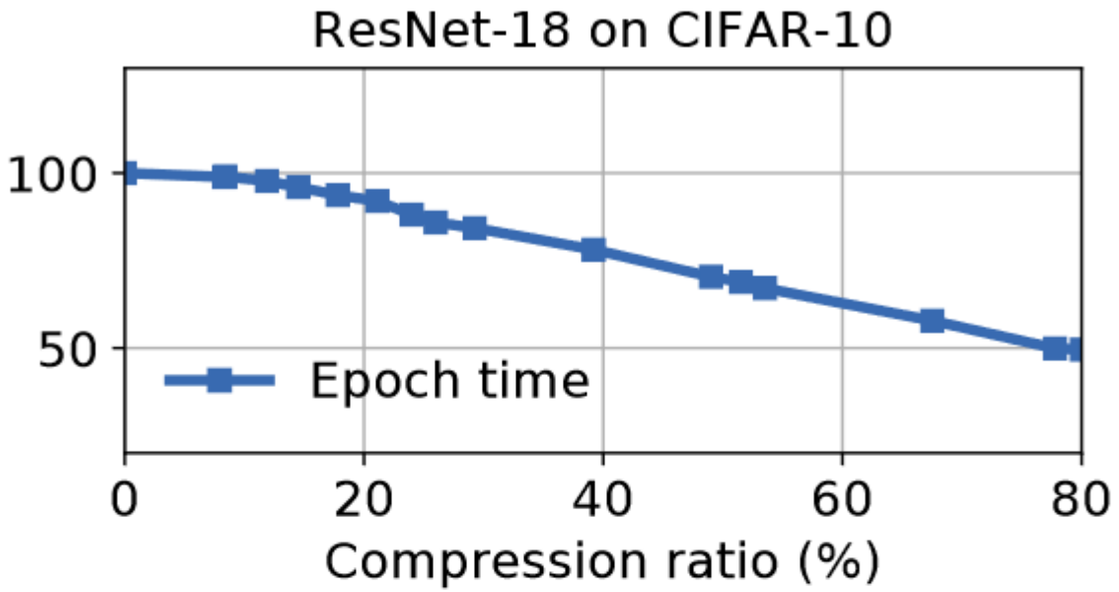

Test accuracy (%)
100
75
50
25
0
0
50
100
150
200
250
300
350
Epoch
E=80%
E=90%
E=95%

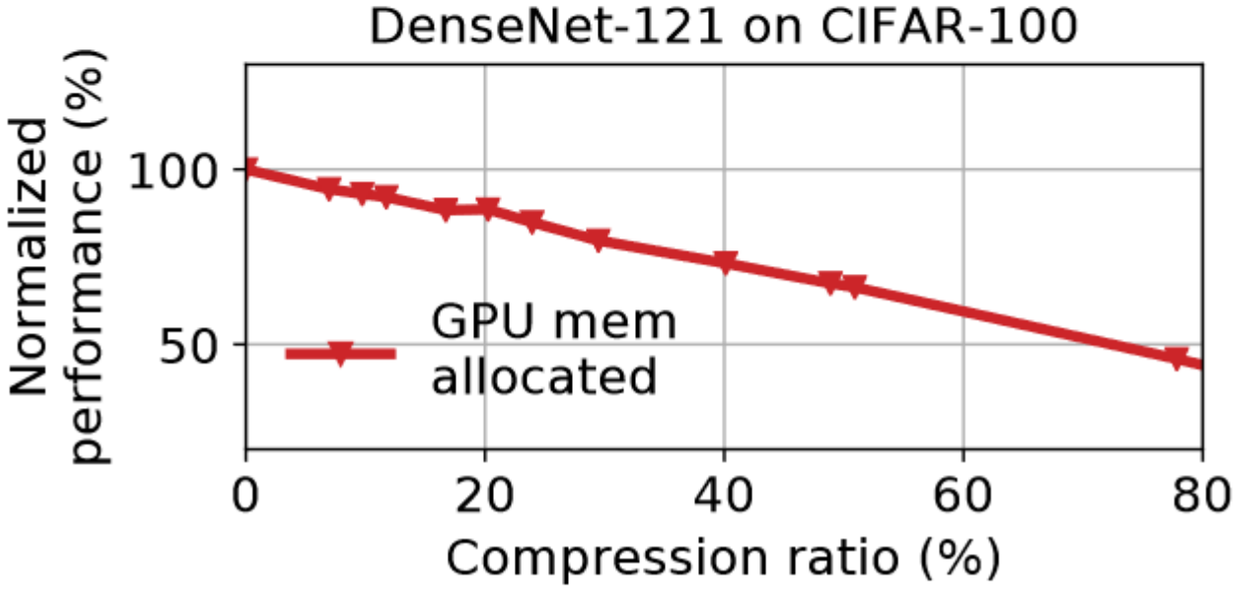

Compression ratio (%)
100
75
50
25
0
0
50
100
150
200
250
300
350
Epoch
E=80%
E=90%
E=95%

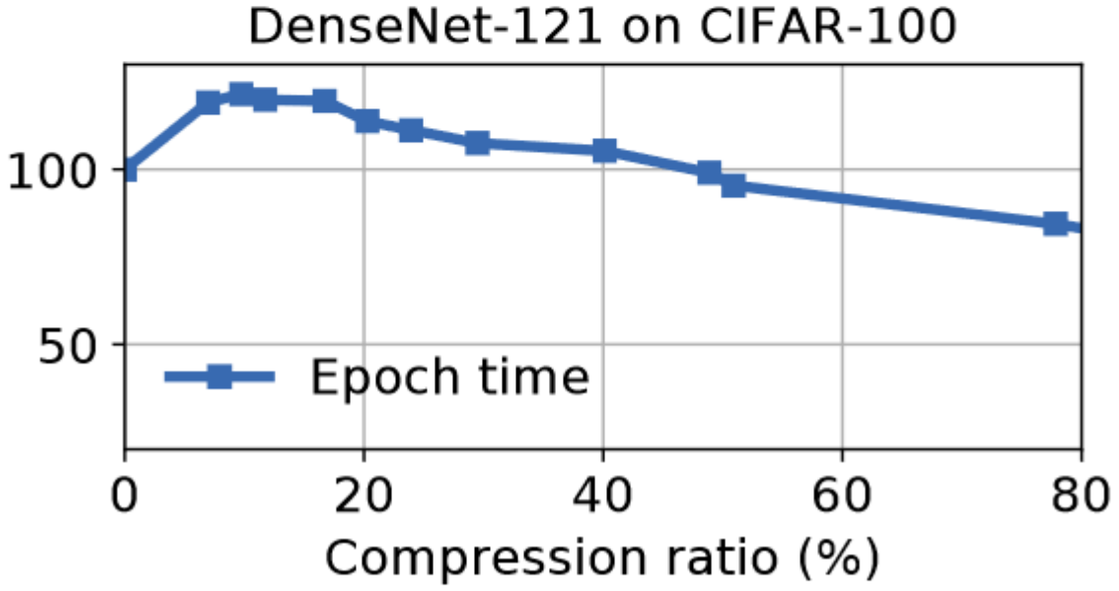

ResNet-18 on CIFAR-10
Test accuracy (%)
Test compression (%)
C=0%
C=30%
C=50%
C=85%
0
25
50
75
100
0
10
20
30
40
50
60
70
80


DenseNet-121 on CIFAR-100
Test accuracy (%)
Test compression (%)
C=0%
C=50%
C=75%
C=85%
0
25
50
75
100
0
10
20
30
40
50
60
70
80


ResNet-18 on CIFAR-10
Normalized performance (%)
100
50
GPU mem allocated
0
20
40
60
80
Compression ratio (%)


ResNet-18 on CIFAR-10
100
50
Epoch time
0
20
40
60
80
Compression ratio (%)


DenseNet-121 on CIFAR-100
Normalized performance (%)
100
50
GPU mem allocated
0
20
40
60
80
Compression ratio (%)


DenseNet-121 on CIFAR-100
100
50
Epoch time
0
20
40
60
80
Compression ratio (%)

"Speaking of longer term, it would be nice if the community migrated to a fully open sourced implementation for all of this [convolution operations, etc.]. This stuff is just too important to the progress of the field for it to be **locked away in proprietary implementations**. The more people working together on this the better for everyone. There's plenty of room to compete on the hardware implementation side."

Scott Gray

https://github.com/soumith/convnet-benchmarks/issues/93

Winograd

Direct Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

Direct Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

$$\boldsymbol{O_0} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}] \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

Direct Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

$$\boldsymbol{O_0} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}] \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

Direct Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

$$\boldsymbol{O_0} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}] \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d_0} \cdot \boldsymbol{g_0} + \boldsymbol{d_1} \cdot \boldsymbol{g_1} + \boldsymbol{d_2} \cdot \boldsymbol{g_2} \\ \boldsymbol{d_1} \cdot \boldsymbol{g_0} + \boldsymbol{d_2} \cdot \boldsymbol{g_1} + \boldsymbol{d_3} \cdot \boldsymbol{g_2} \end{bmatrix}$$

6 multiplications & 4 additions

Winograd Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

$$\boldsymbol{O_0} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}] \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1 + m_2 + m_3} \\ \boldsymbol{m_2 - m_3 - m_4} \end{bmatrix}$$

Winograd Convolution

$$\boldsymbol{I} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}\ \boldsymbol{d_3}] \quad \boldsymbol{F} = [\boldsymbol{g_0}\ \boldsymbol{g_1}\ \boldsymbol{g_2}]$$

$$\boldsymbol{O_0} = [\boldsymbol{d_0}\ \boldsymbol{d_1}\ \boldsymbol{d_2}] \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix}$$

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1 + m_2 + m_3} \\ \boldsymbol{m_2 - m_3 - m_4} \end{bmatrix}$$

Winograd Convolution

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1 + m_2 + m_3} \\ \boldsymbol{m_2 - m_3 - m_4} \end{bmatrix}$$

Winograd Convolution

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1 + m_2 + m_3} \\ \boldsymbol{m_2 - m_3 - m_4} \end{bmatrix}$$

$$\boldsymbol{m_1 = (d_0 - d_2) \times g_0} \quad \boldsymbol{m_2 = (d_1 + d_2) \times \frac{g_0 + (g_1 + g_2)}{2}}$$

$$\boldsymbol{m_4 = (d_1 - d_3) \times g_2} \quad \boldsymbol{m_3 = (d_2 - d_1) \times \frac{g_0 - (g_1 + g_2)}{2}}$$

Winograd Convolution

$$\boldsymbol{O} = \begin{bmatrix} \boldsymbol{d_0} & \boldsymbol{d_1} & \boldsymbol{d_2} \\ \boldsymbol{d_1} & \boldsymbol{d_2} & \boldsymbol{d_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{g_0} \\ \boldsymbol{g_1} \\ \boldsymbol{g_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_1 + m_2 + m_3} \\ \boldsymbol{m_2 - m_3 - m_4} \end{bmatrix}$$

$$\boldsymbol{m_1 = (d_0 - d_2) \times g_0} \quad \boldsymbol{m_2 = (d_1 + d_2) \times \frac{g_0 + (g_1 + g_2)}{2}}$$

$$\boldsymbol{m_4 = (d_1 - d_3) \times g_2} \quad \boldsymbol{m_3 = (d_2 - d_1) \times \frac{g_0 - (g_1 + g_2)}{2}}$$

4 multiplications & 7 additions & 2 bit shifts

Matrix form for 1D: $\mathbf{Y} = \boldsymbol{A^T}[(\boldsymbol{Gg}) \odot (\boldsymbol{B^T d})]$

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$

$$g = \begin{bmatrix} g_0 & g_1 & g_2 \end{bmatrix}^T$$

$$d = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \end{bmatrix}^T$$

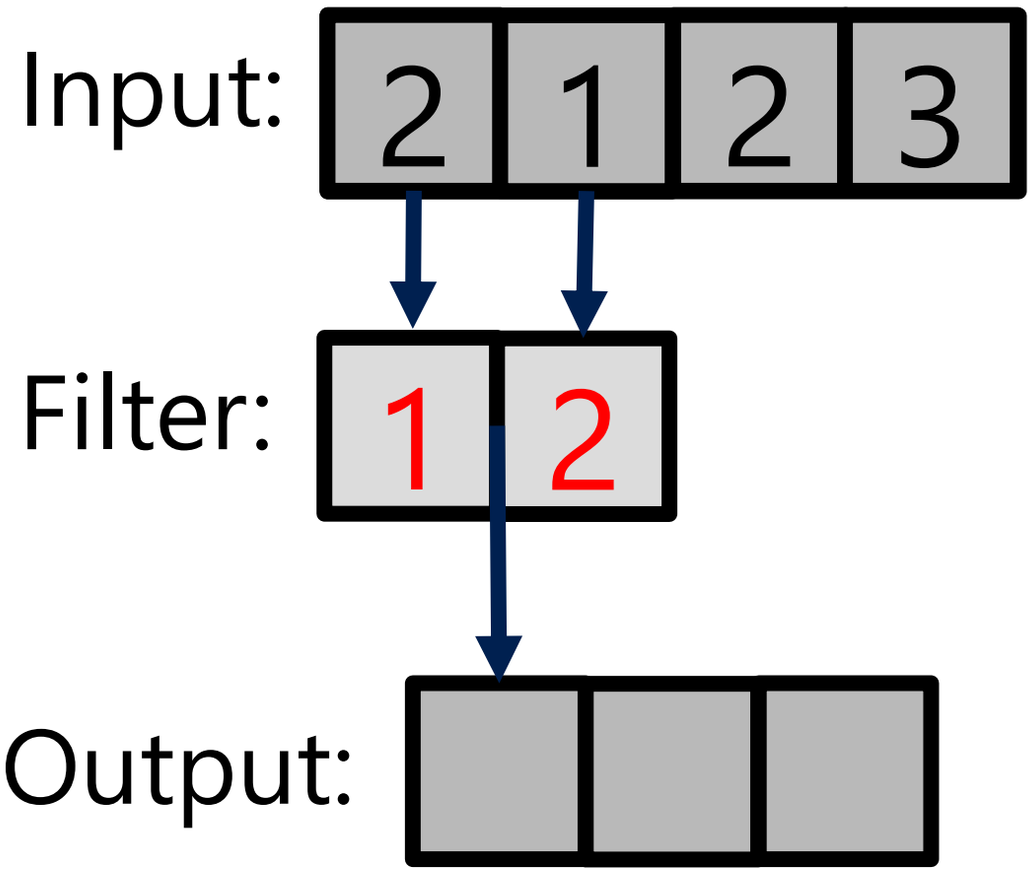
2D Winograd Convolution

- Nested 1D Winograd Convolution gives the 2D algorithm: $\mathbf{Y} = \boldsymbol{A^T}\left[\left(\boldsymbol{GgG^T}\right)\odot\left(\boldsymbol{B^TdB}\right)\right]\boldsymbol{A}$
- $\boldsymbol{g}$ is an $\boldsymbol{r}\times\boldsymbol{r}$ filter and $\boldsymbol{d}$ is an $(\boldsymbol{m}+\boldsymbol{r}-\mathbf{1})\times(\boldsymbol{m}+\boldsymbol{r}-\mathbf{1})$ image tile
- Winograd $\boldsymbol{F}(\mathbf{2}\times\mathbf{2},\mathbf{3}\times\mathbf{3})$ uses $\mathbf{4}\times\mathbf{4}=\mathbf{16}$ multiplications, while standard convolution uses $\mathbf{2}\times\mathbf{2}\times\mathbf{3}\times\mathbf{3}=\mathbf{36}$ multiplications, reduction: 2.25

Direct & FFT convolution

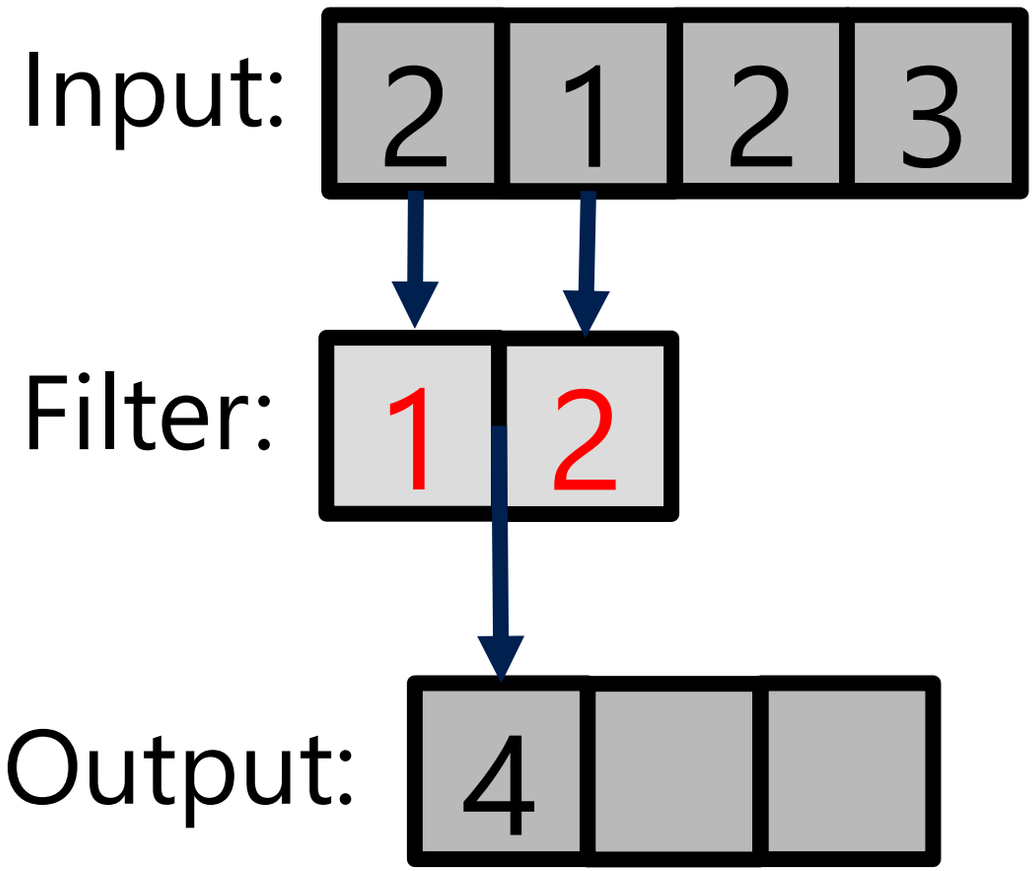
Convolutions in Deep Neural Networks

Direct



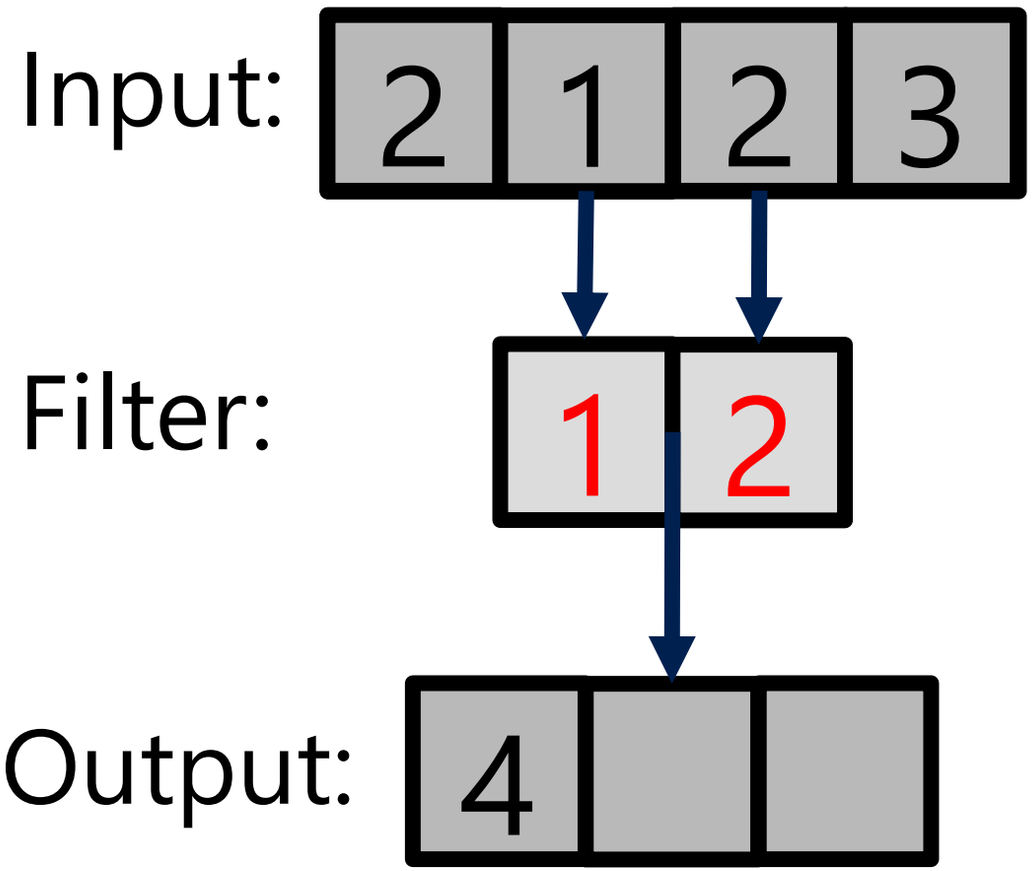
Convolutions in Deep Neural Networks

Direct



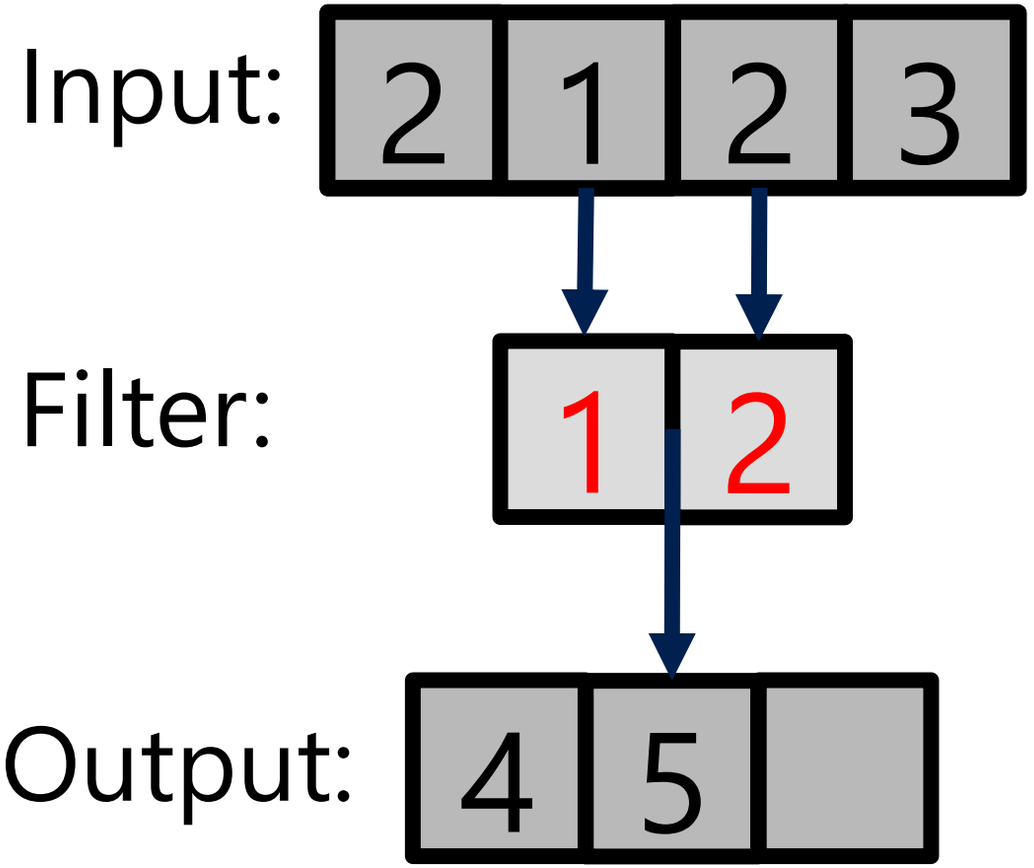
Convolutions in Deep Neural Networks

Direct



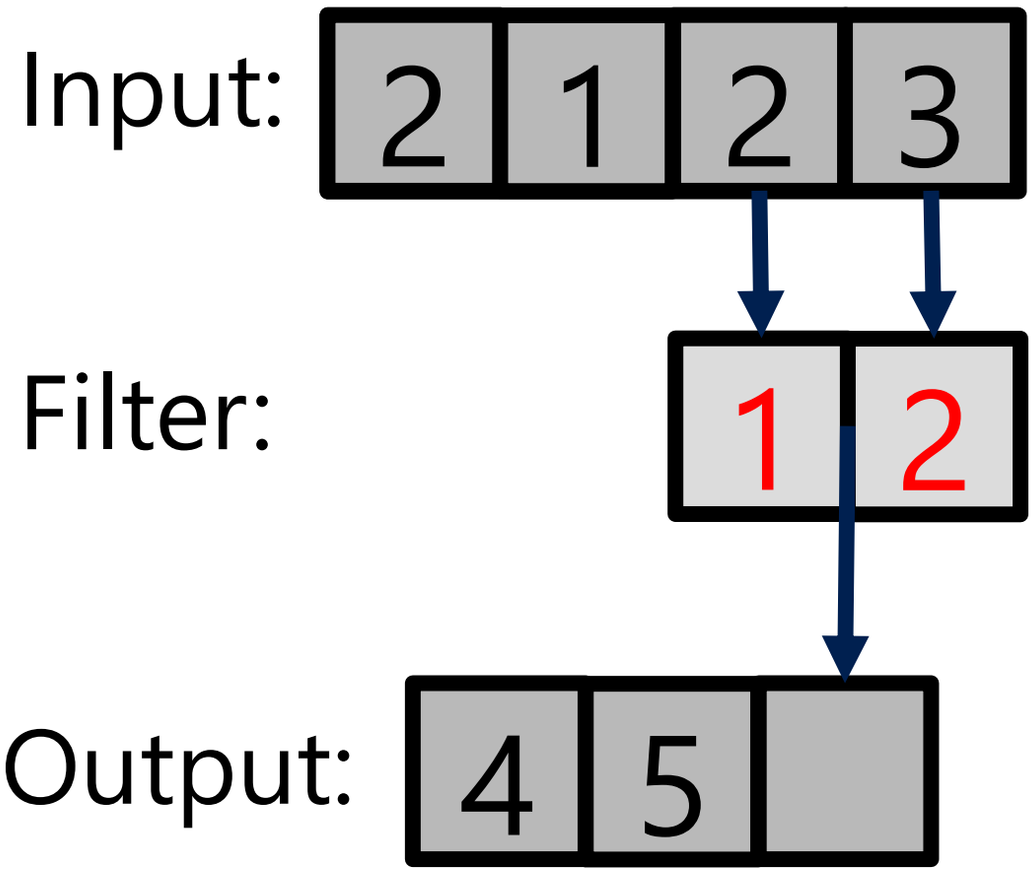
Convolutions in Deep Neural Networks

Direct



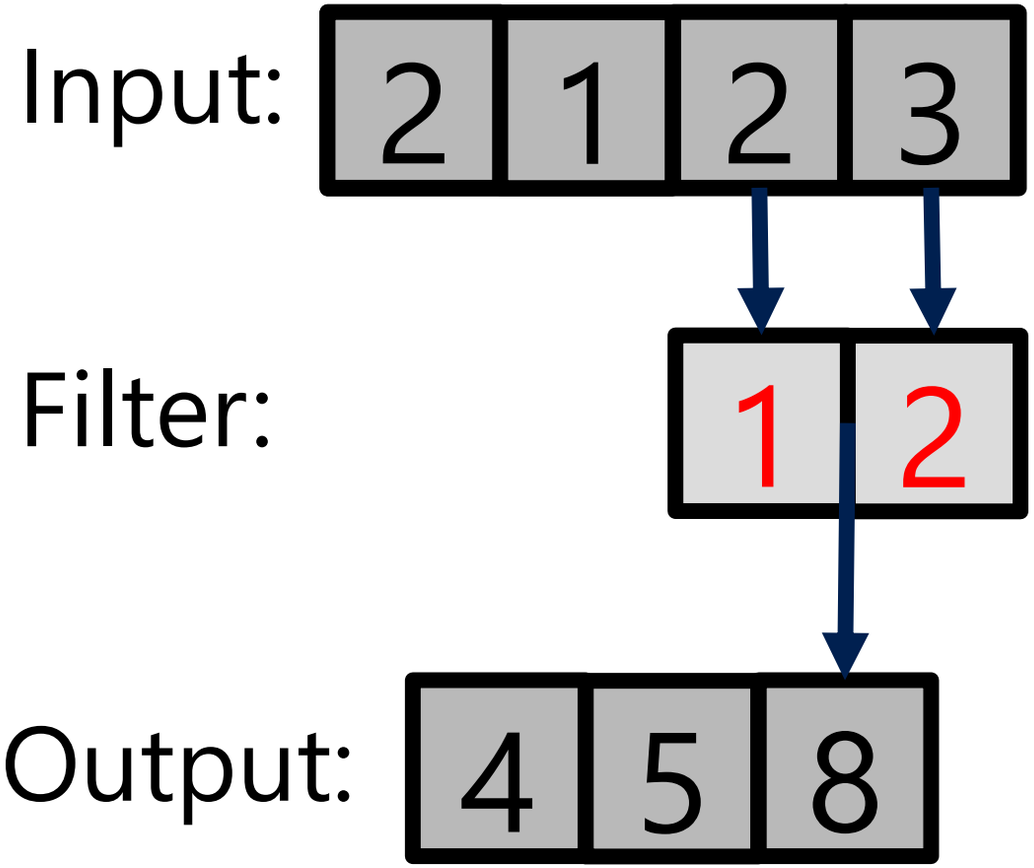
Convolutions in Deep Neural Networks

Direct



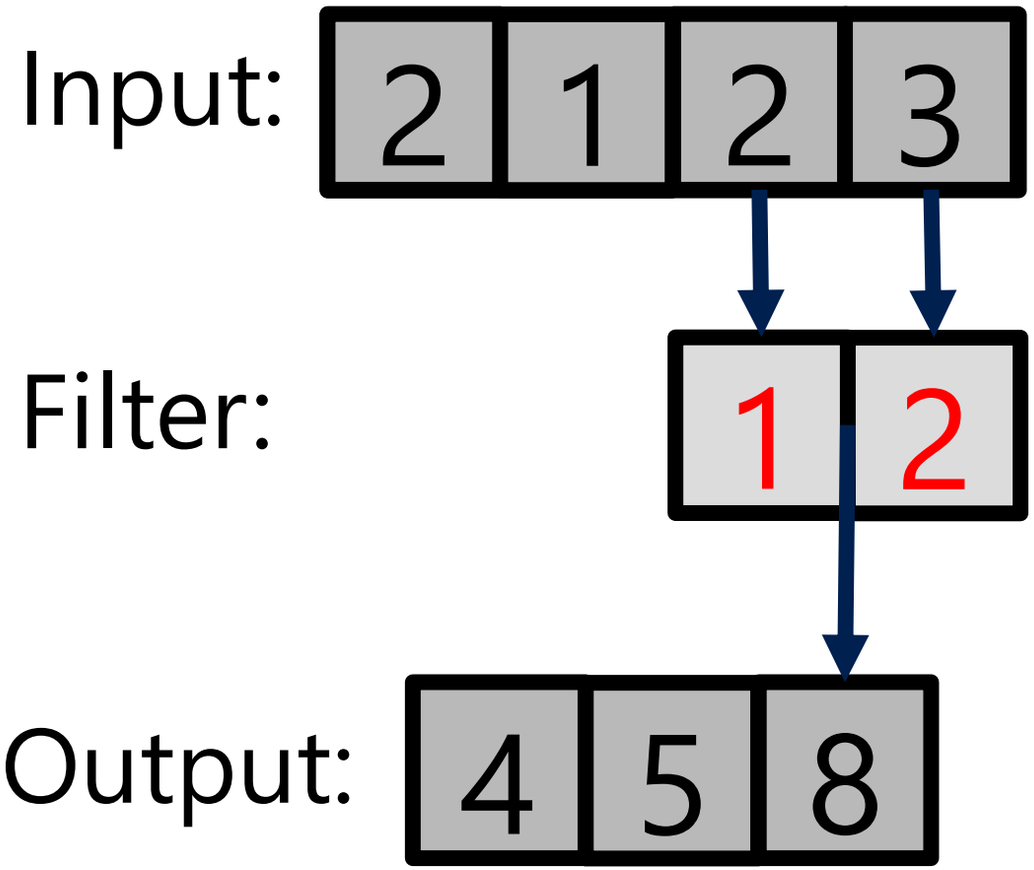
Convolutions in Deep Neural Networks

Direct



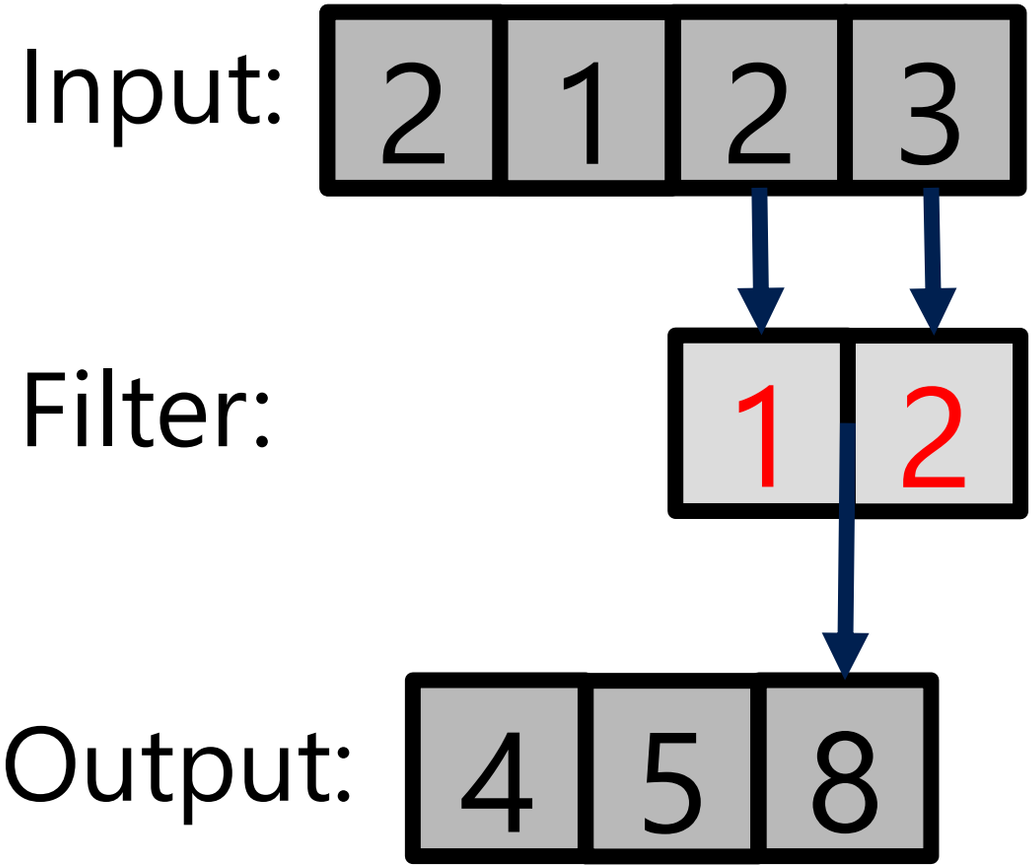
Convolutions in Deep Neural Networks

Direct **via FFT**



Convolutions in Deep Neural Networks

Direct

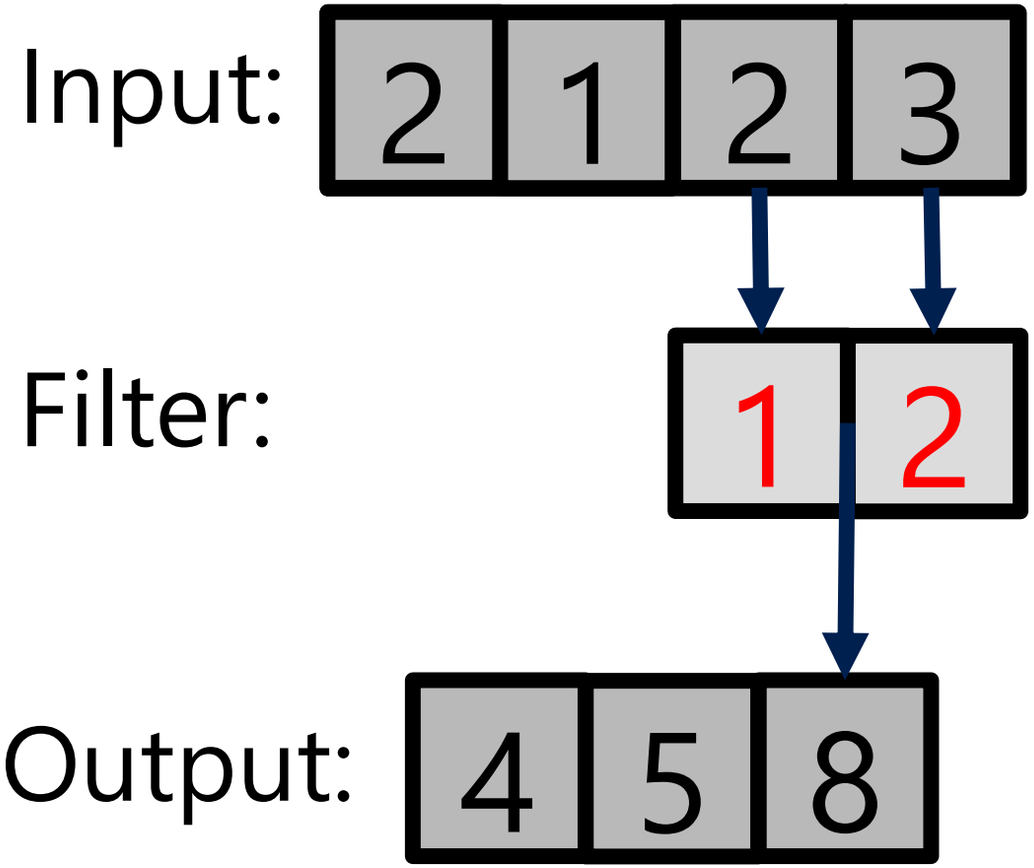


via FFT

I = FFT(Input)

Convolutions in Deep Neural Networks

Direct



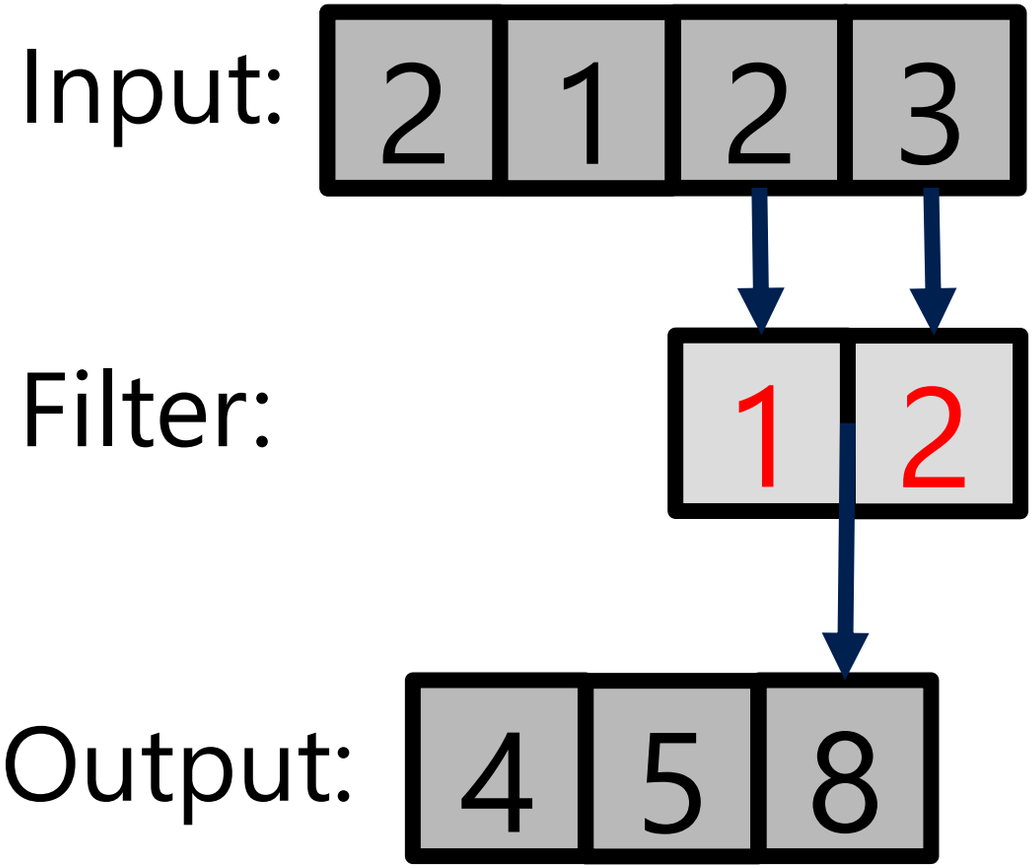
via FFT

I = FFT(Input)

F = cj(FFT(Filter))

Convolutions in Deep Neural Networks

Direct



via FFT

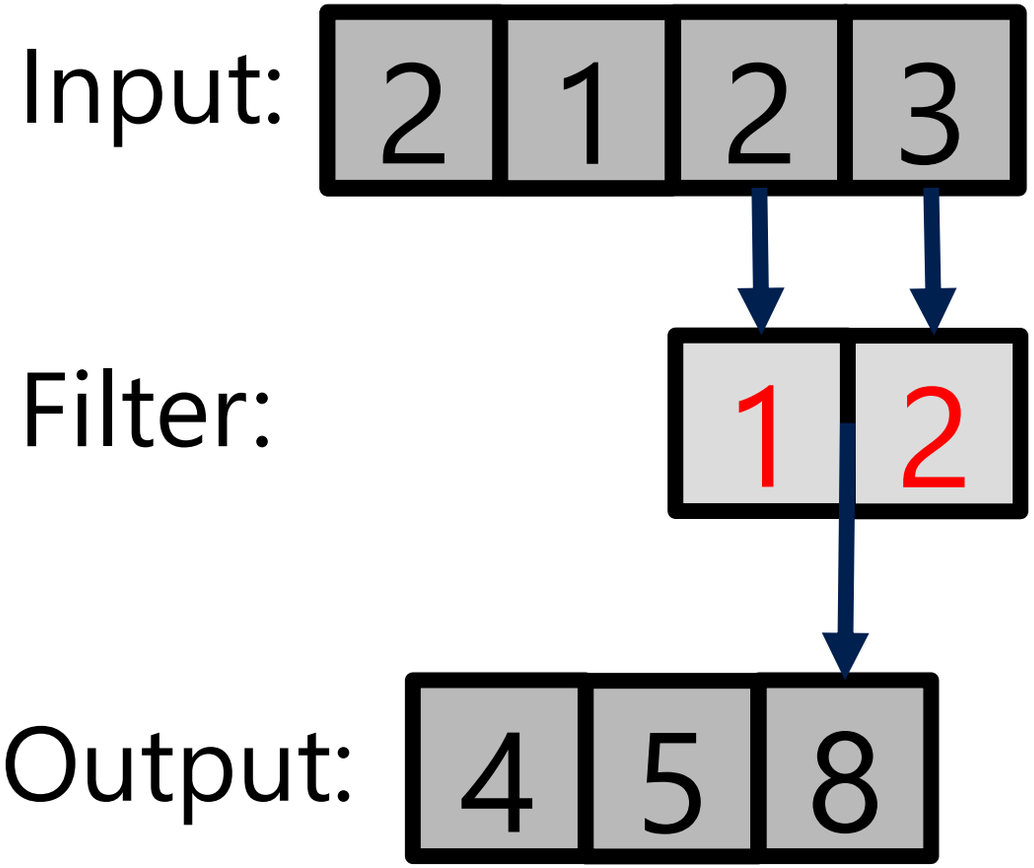
$$I = \text{FFT}(\text{Input})$$

$$F = cj(\text{FFT}(\text{Filter}))$$

$$O = I \odot F$$

Convolutions in Deep Neural Networks

Direct



via FFT

$$I = \text{FFT}(\text{Input})$$

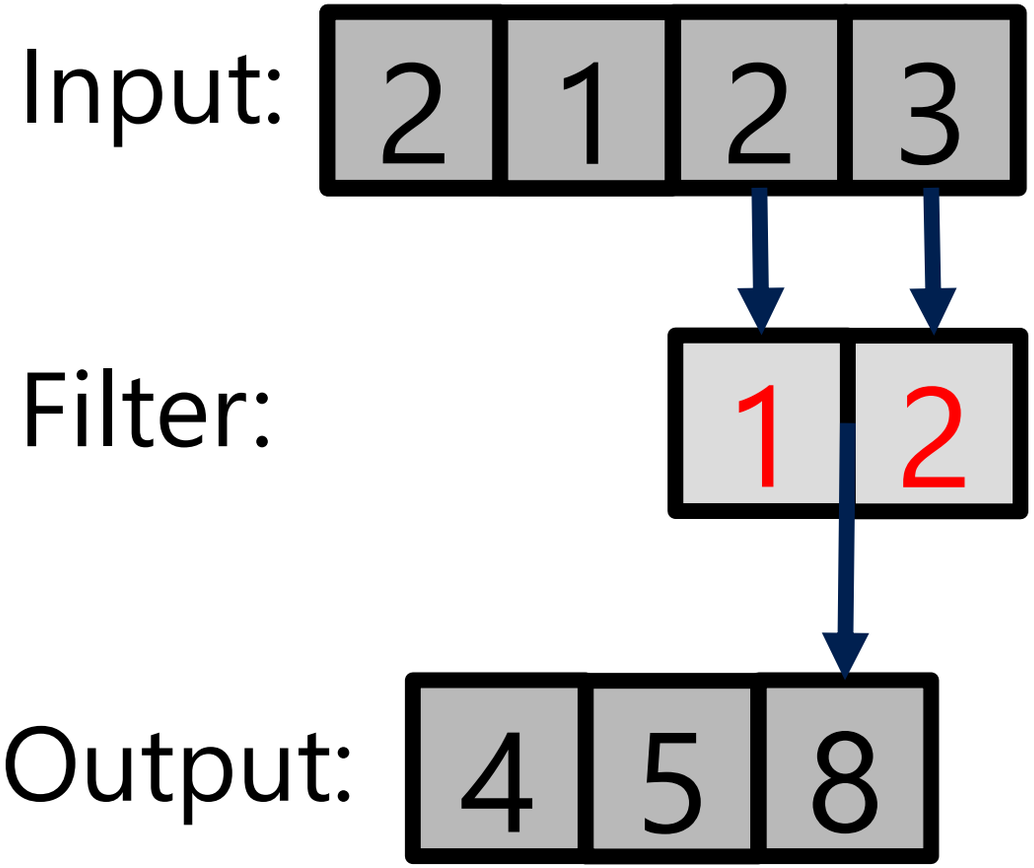
$$F = cj(\text{FFT}(\text{Filter}))$$

$$O = I \odot F$$

$$\text{Output} = \text{IFFT}(O)$$

Convolutions in Deep Neural Networks

Direct O(kn)



via FFT O($n \log n$)

I = FFT(Input)

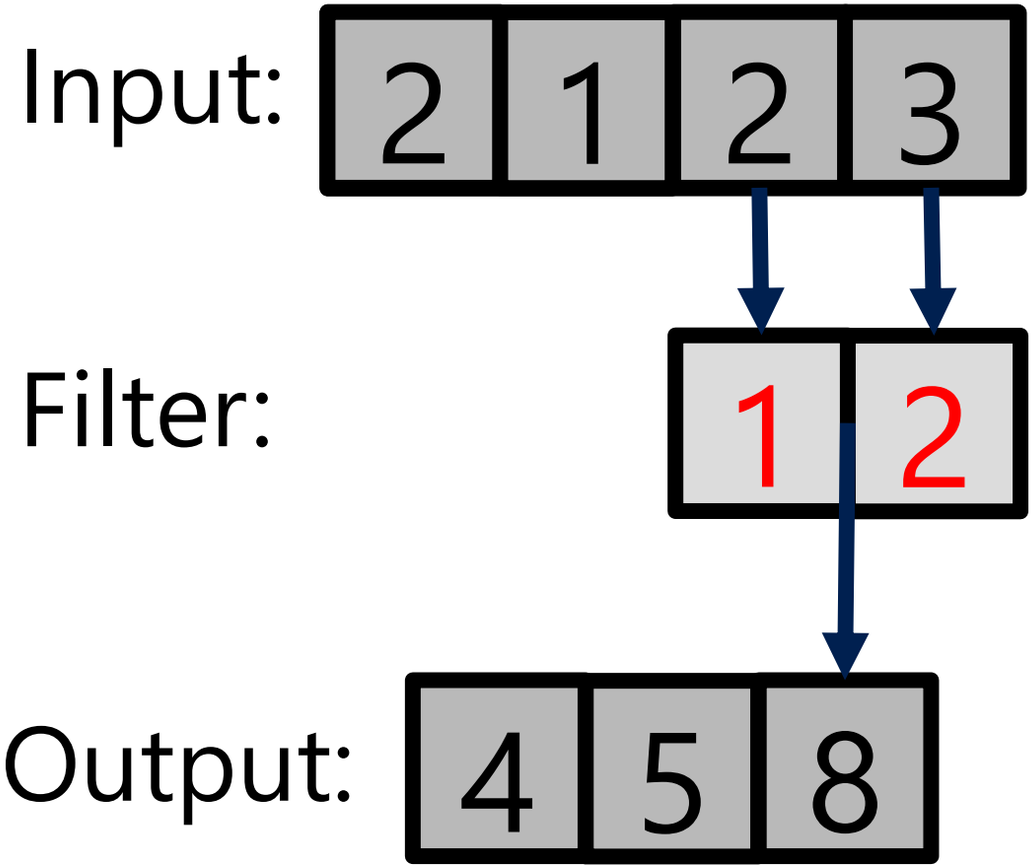
F = cj(FFT(Filter))

O = I ⊙ F

Output = IFFT(O)

Convolutions in Deep Neural Networks

Direct O(kn) $\xrightarrow[k \to n]{}$ **O(n^2)**



via FFT O($n \log n$)

I = FFT(Input)

F = cj(FFT(Filter))

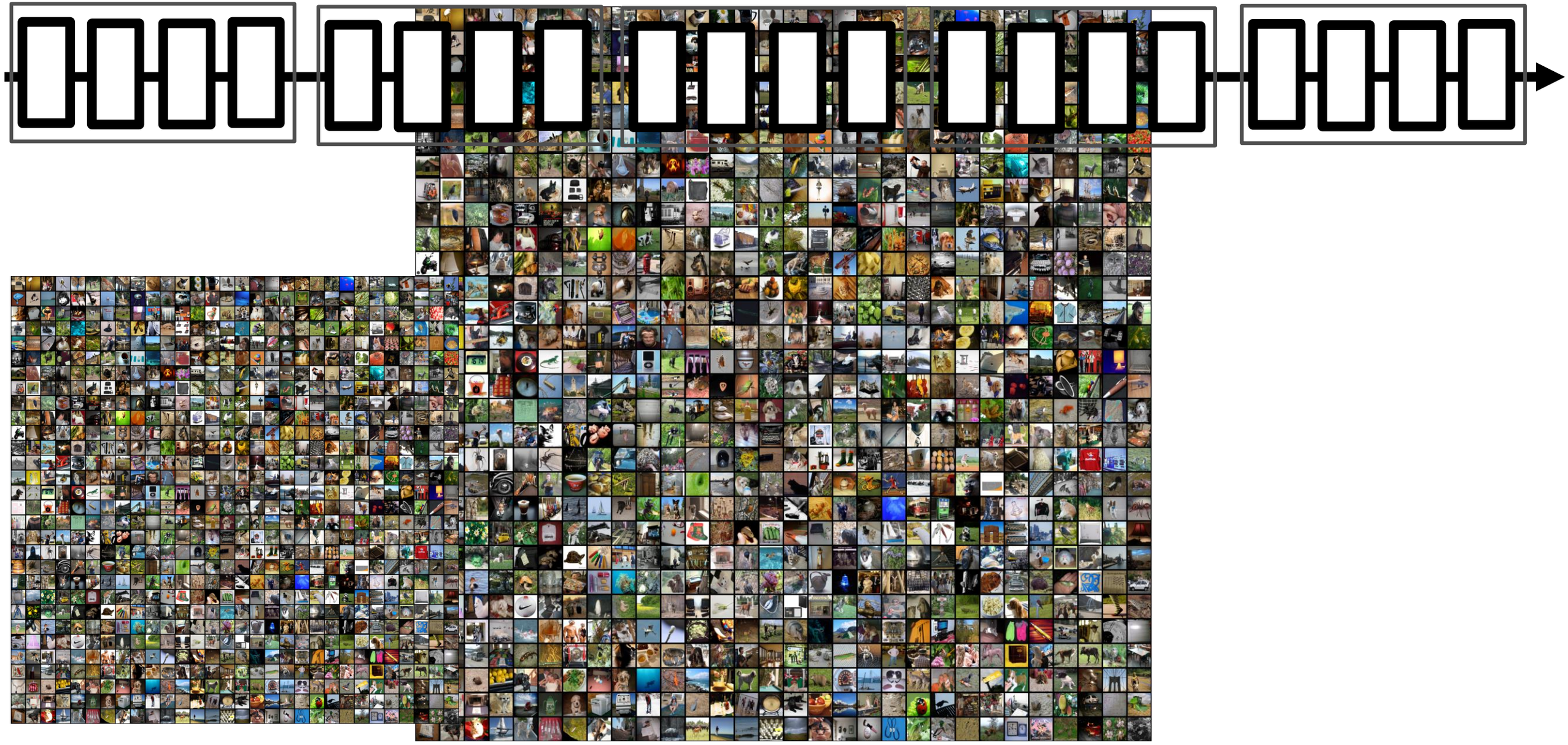
O = I ⊙ F

Output = IFFT(O)

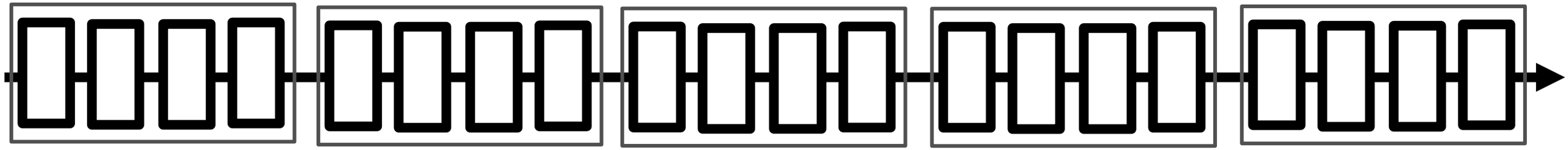
Long Training & Adversarial Vulnerability

Training Convolutional Neural Networks

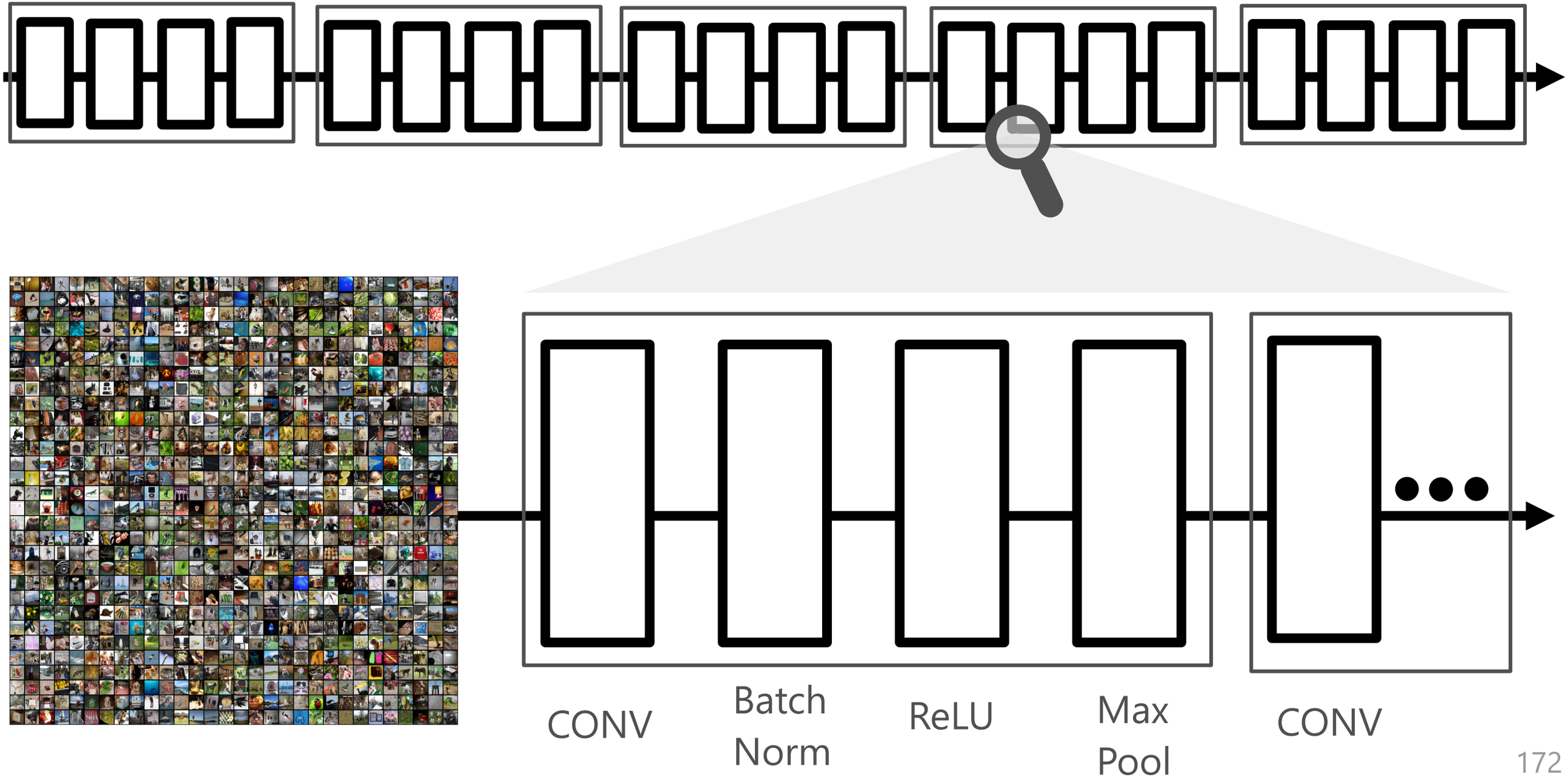
Training Convolutional Neural Networks



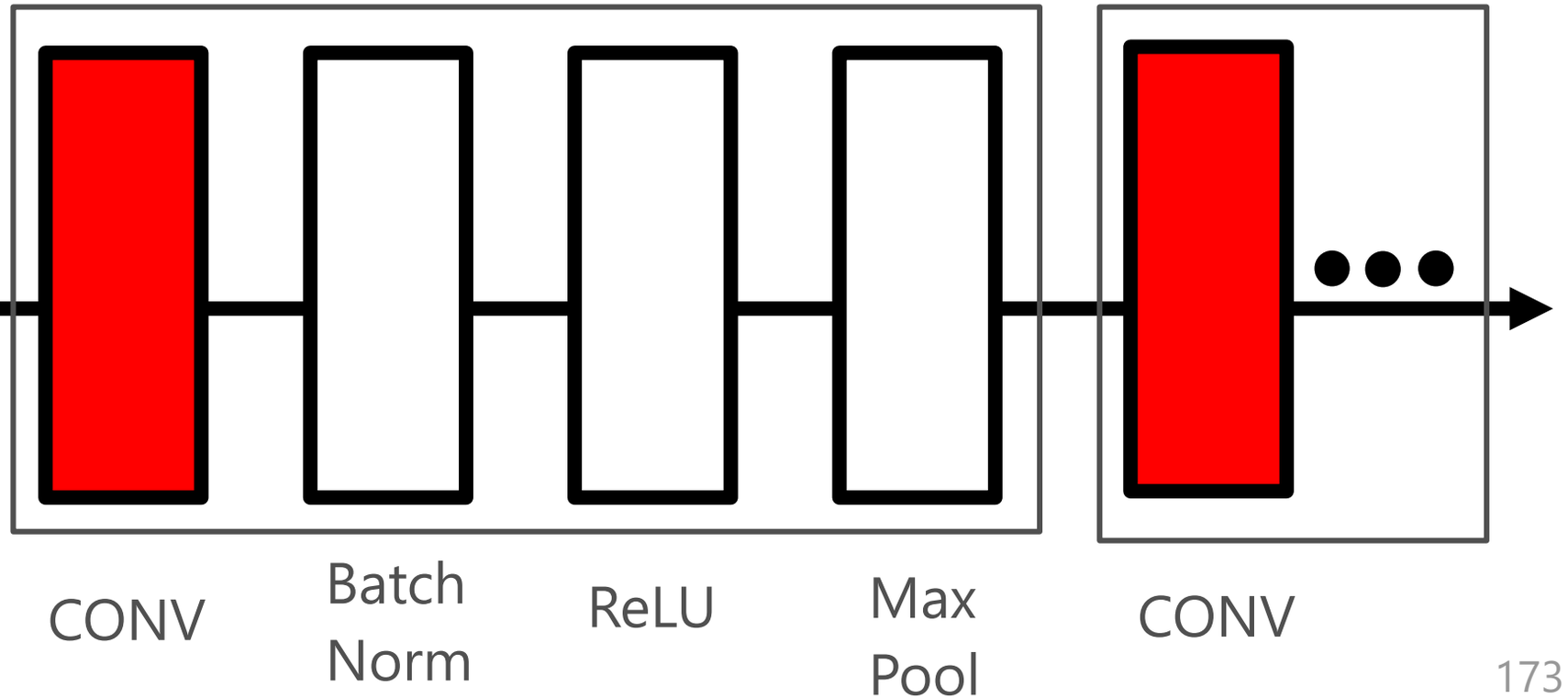
Training Convolutional Neural Networks

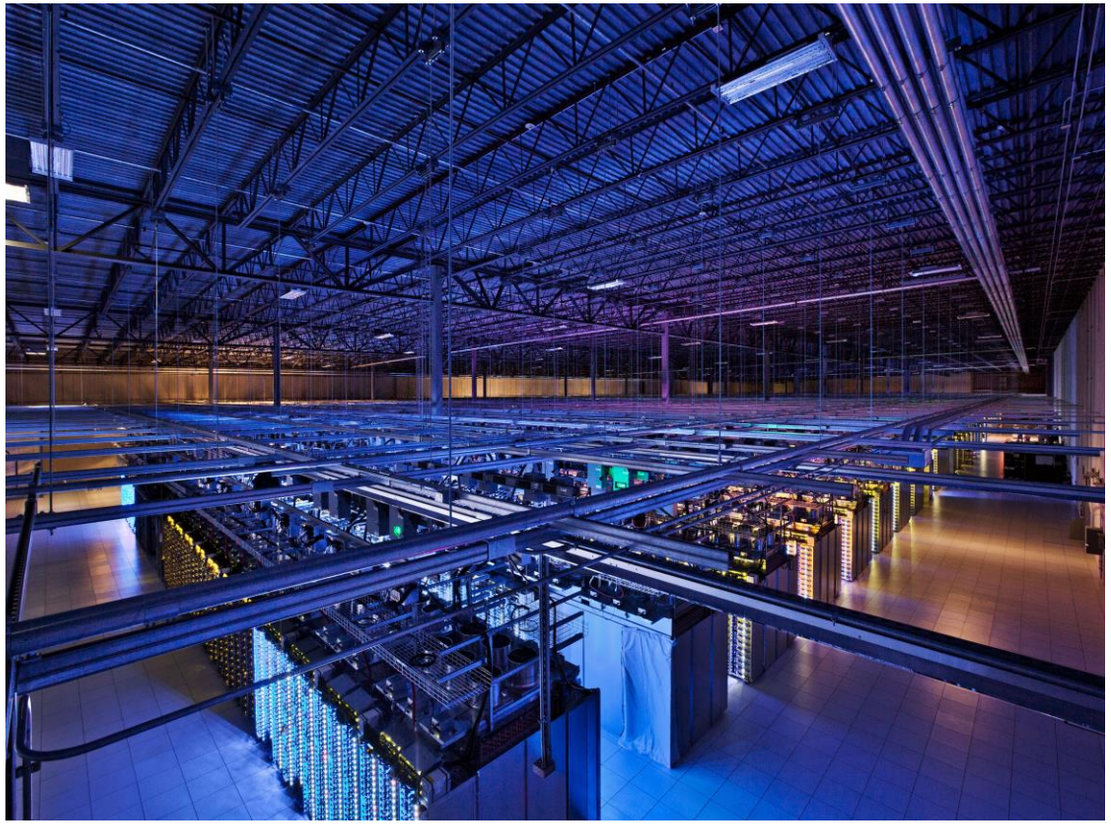


Training Convolutional Neural Networks



Training Convolutional Neural Networks





Training VS Inference

Dog

Dog

Dog

Hummingbird

Quiz

Quiz

Quiz

Hummingbird

Hummingbird