An automated approach for improving the inference latency and energy efficiency of pretrained CNNs by removing irrelevant pixels with focused convolutions

George K. Thiruvathukal, Professor and Chair, Loyola University Chicago, Computer Science Department

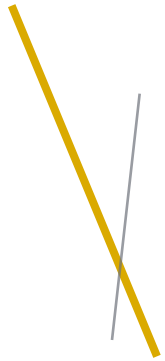
Caleb Tung, Nicholas Eliopoulos, Purvish Jajal, Gowri Ramshankar, Chen-Yun Yang (Purdue Univ., USA), Nicholas Synovic (Loyola Univ. Chicago, USA), Xuecen Zhang, Vipin Chaudhary (Case Western Reserve Univ., USA), George K. Thiruvathukal (Loyola Univ. Chicago, USA), Yung-Hsiang Lu (Purdue Univ., USA)

PURDUE UNIVERSITY
Elmore Family School of Electrical and Computer Engineering

LOYOLA UNIVERSITY CHICAGO
DEPARTMENT OF COMPUTER SCIENCE

CASE WESTERN RESERVE UNIVERSITY
CASE SCHOOL OF ENGINEERING

In computer vision, neural networks need lots of compute



Compute needed to run CNNs (already less intensive than Vision Transformers)

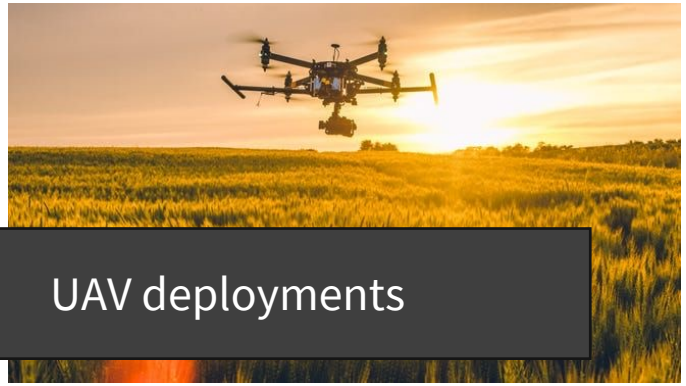
CNN	Measure of Computational Intensity	Value
VGG-16	Number of Parameters	138M
	Number of MACs	15.4G
	Model Size	553.4 MB
ResNet-18	Number of Parameters	11.7M
	Number of MACs	1.81G
	Model Size	46.7 MB
ConvNeXt-B	Number of Parameters	88.6M
	Number of MACs	15.4G
	Model Size	354.2 MB
Mask-RCNN	Number of Parameters	44.4M
	Number of MACs	134.4G
	Model Size	177.6 MB
RetinaNet	Number of Parameters	34.0M
	Number of MACs	151.6G
	Model Size	136.1 MB

Power-hungry (~1000 W) GPUs used to run these CNNs

Driver assist/collision avoidance

But what about when GPUs and network access is unavailable?

Trap cameras



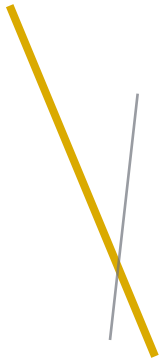
UAV deployments

Most methods of making CNNs more efficient require re-training

Millions of training images

Hours of GPU time

Costs lots of money



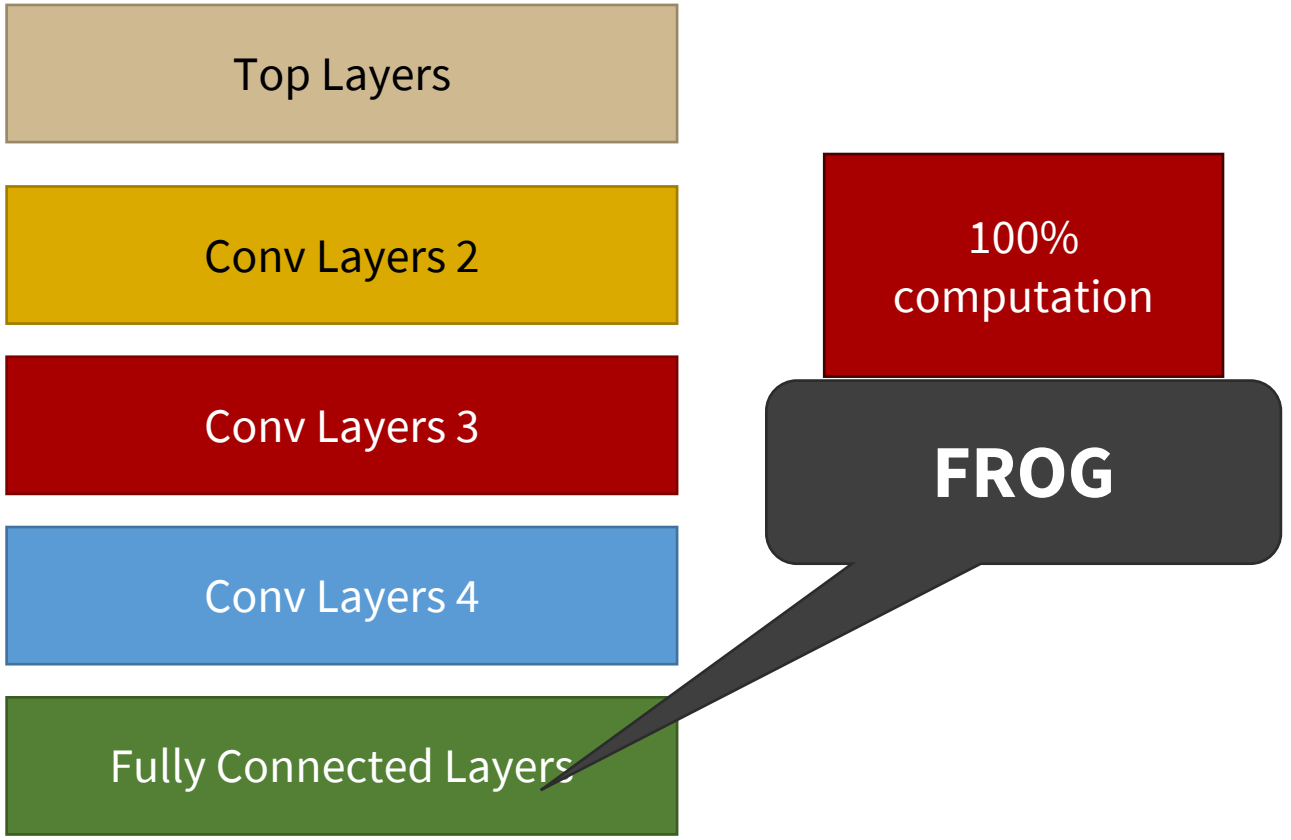
Focused convolutions for a *pretrained CNN*

No re-training

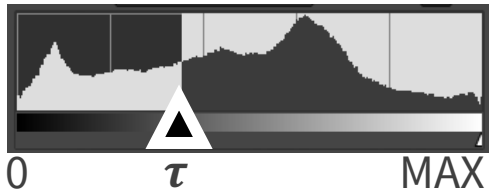
Drop-in, more efficient replacement for Conv2D

Maintains accuracy

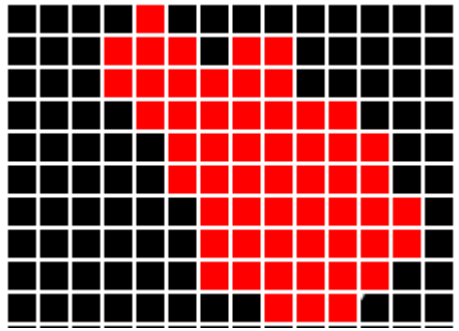
A pretrained CNN



Top Layers



Apply threshold



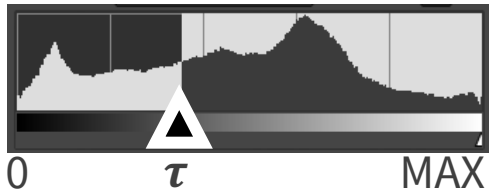
Conv Layers 2

Conv Layers 3

Conv Layers 4

Fully Connected Layers

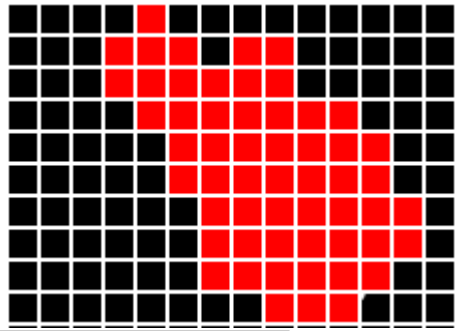
Top Layers



Apply threshold

SAVE ENERGY. ONLY COMPUTE ON RED SQUARES

Focused Convolutions



Focused

Conv Layers 2

Focused

Conv Layers 3

Focused

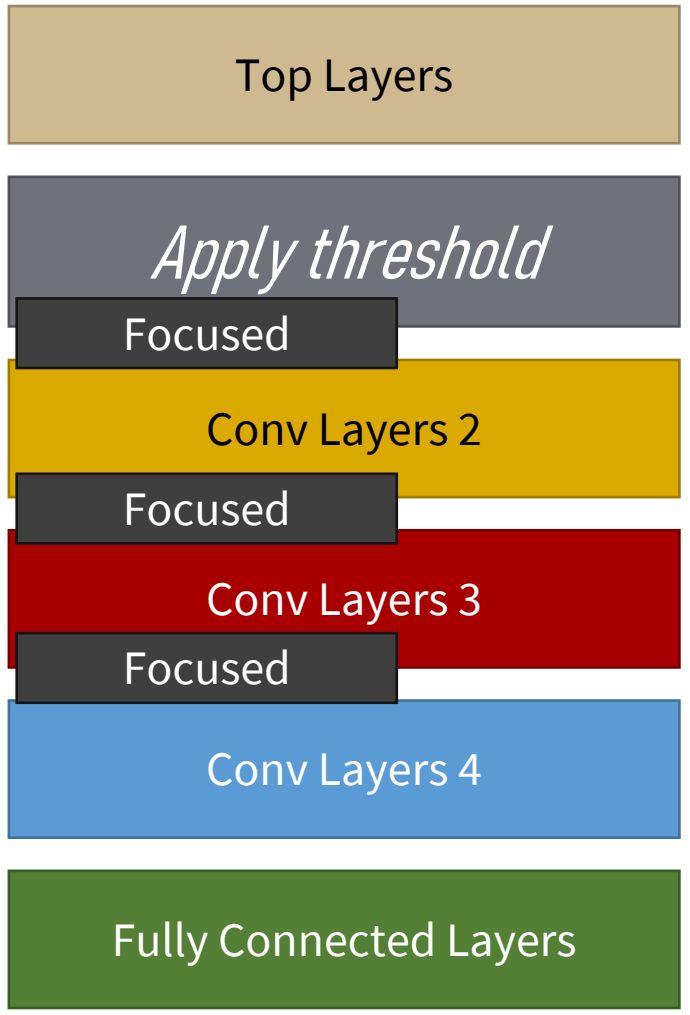
Conv Layers 4

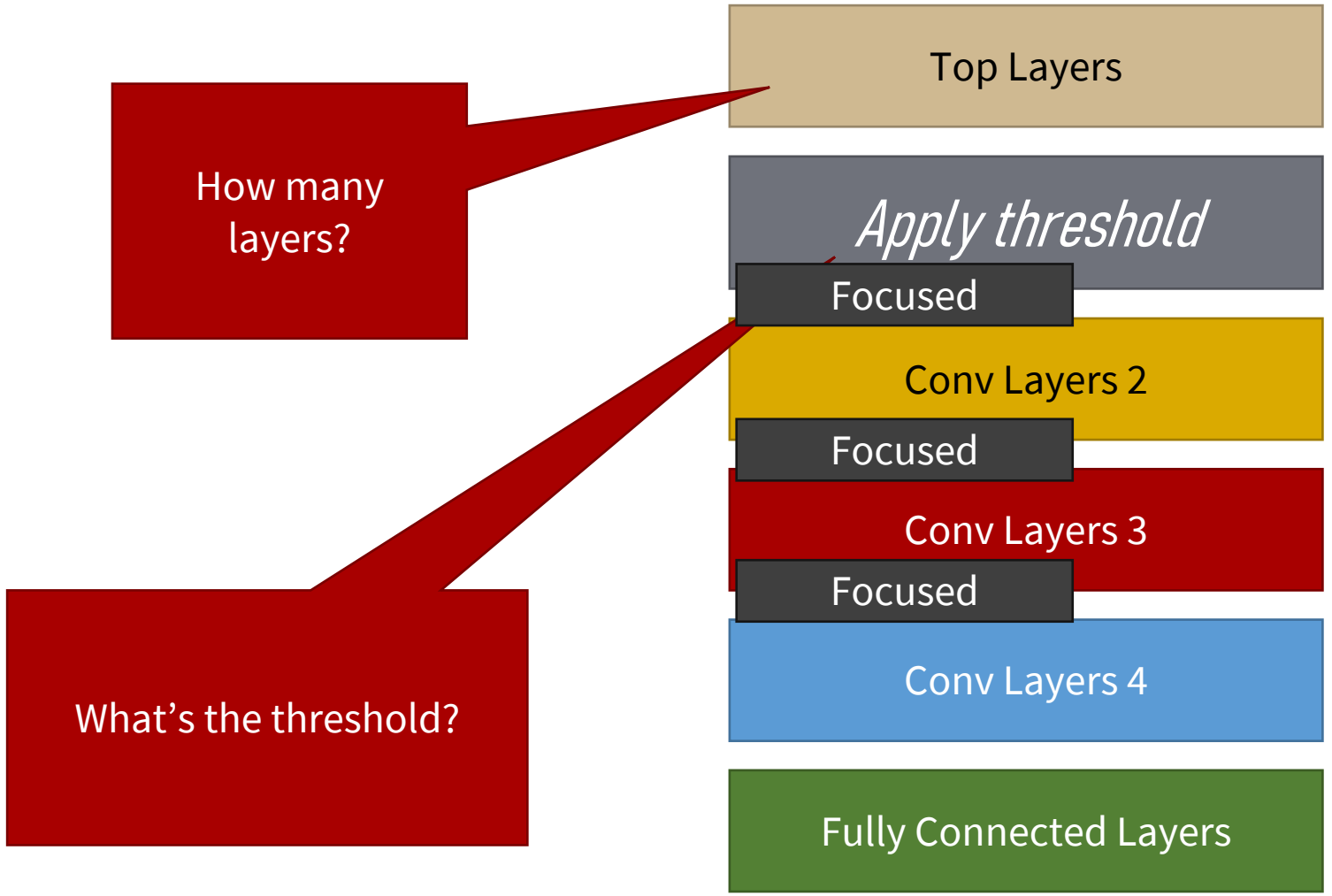
< 100% computation

FROG

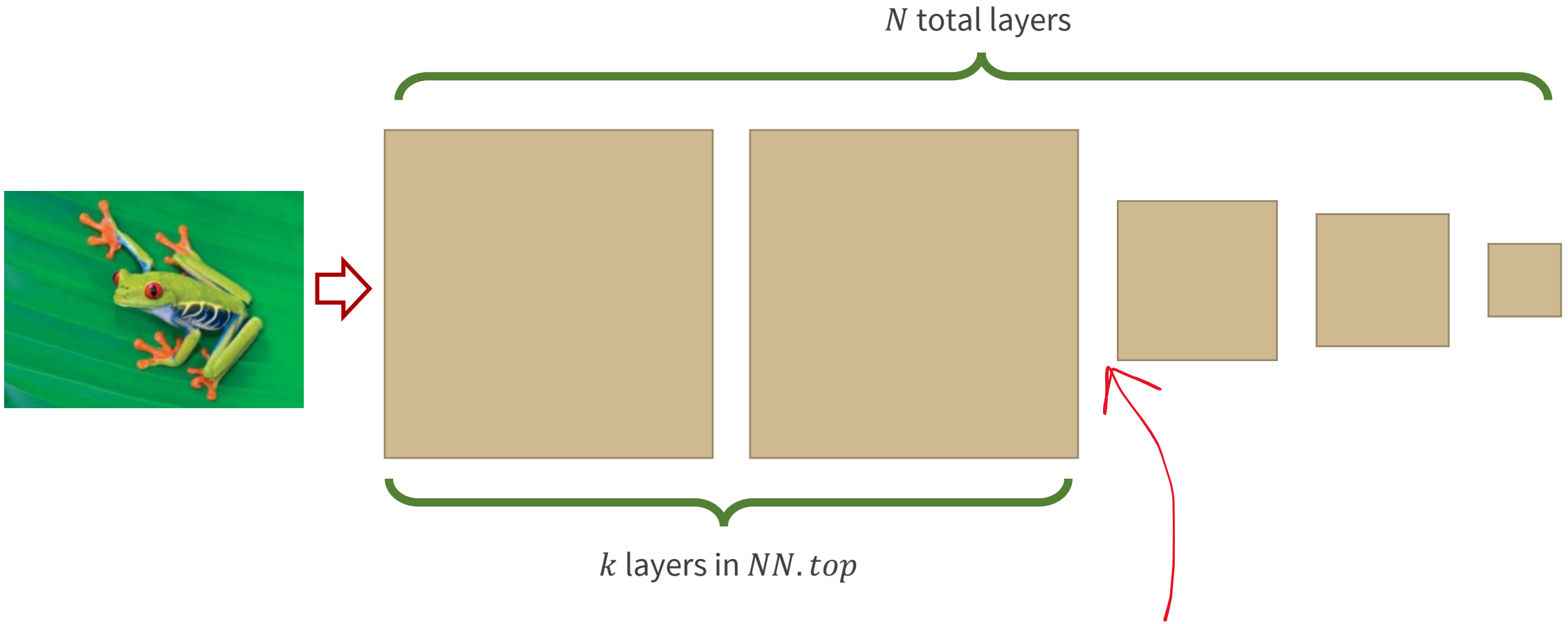
Fully Connected Layers

Proposed design: Fully self-contained “fCNN”





How to choose the k layers in $NN.top$



How to choose the layers in $\boldsymbol{NN.top}$

$$E_{f,i} = aE_{c,i} + C$$

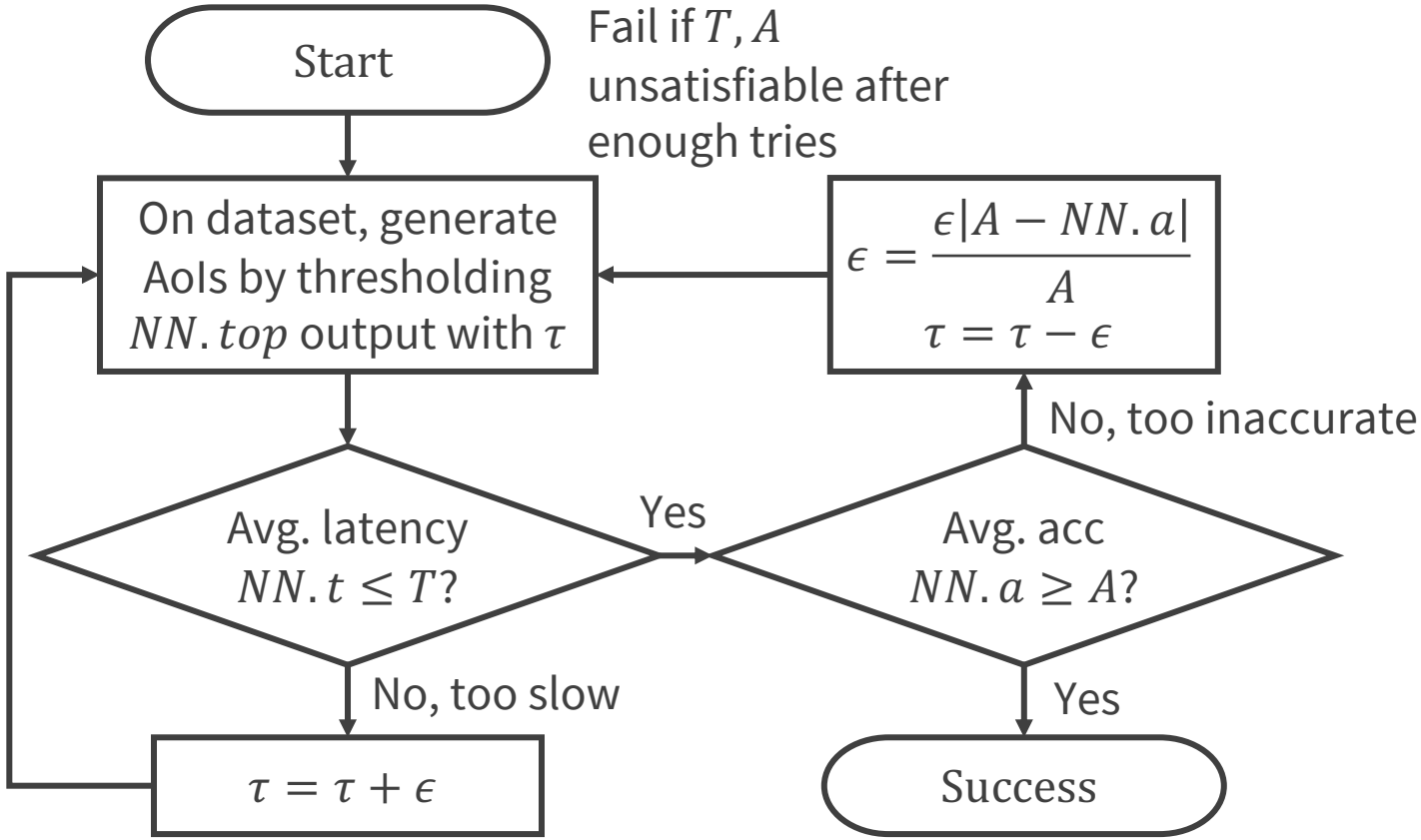
Overhead from focused conv

$$a \propto \frac{AoI\ Area}{Image\ Area}$$

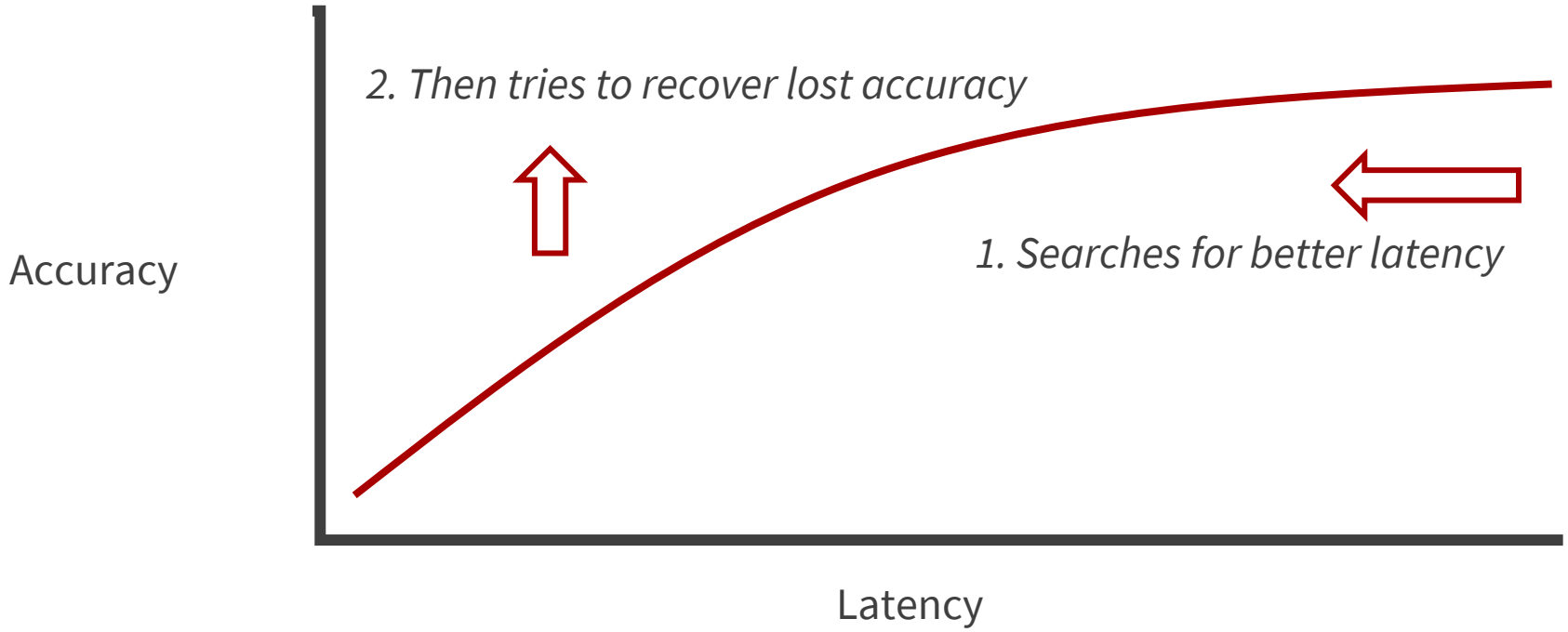
$$E_{total} = (N-k)c + \sum_{i=1}^{n} E_{c,i} + \sum_{i=n+1}^{N} E_{f,i}$$

Adjust k until E_{total} estimate is satisfactory

How to choose the threshold $\boldsymbol{\tau}$



This method searches along a curve



Original

```
VGG(
  (features): Seque
    (0): Conv2d(3, 64, kernel_size=(3, 3), )
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), )
    (3): ReLU(inplace=True)

    # AoI threshold layer will be inserted here
    # Layers 4-30 currently operating on
    # irrelevant pixels
    (4): MaxPool2d(kernel_size=2, stride=2, )
    (5): Conv2d(64, 128, kernel_size=(3, 3), )
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), )
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, )
    (10): Conv2d(128, 256, kernel_size=(3, 3), )
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), )
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), )
```

Focused

```
VGG(
  (features): Sequen
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), )
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), )
      (3): ReLU(inplace=True)
    )
    (1): AoIThresholder()
    (2): Sequential(
      (4): MaxPool2d(kernel_size=2, stride=2, )
      (5): FocusedConv2d(64, 128, kernel_size=(3, 3), )
      (6): ReLU(inplace=True)
      (7): FocusedConv2d(128, 128, kernel_size=(3, 3), )
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, )
      (10): FocusedConv2d(128, 256, kernel_size=(3, 3), )
      (11): ReLU(inplace=True)
      (12): FocusedConv2d(256, 256, kernel_size=(3, 3), )
      (13): ReLU(inplace=True)
      (14): FocusedConv2d(256, 256, kernel_size=(3, 3), )
```

Results

A qualitative look

Microsoft COCO image

Without training, compared to original CNN…

VGG-16, ResNet-18, ConvNext-T

ImageNet

12% energy savings
15% faster

Faster-RCNN, SSD-Lite

COCO

28% energy savings
30% faster

arm AMD intel

MONSOON HV Power Meter

0-3% Accuracy Loss

fCNNs are faster and comparably accurate

ResNet-18: 18ms

fResNet-18: 16ms

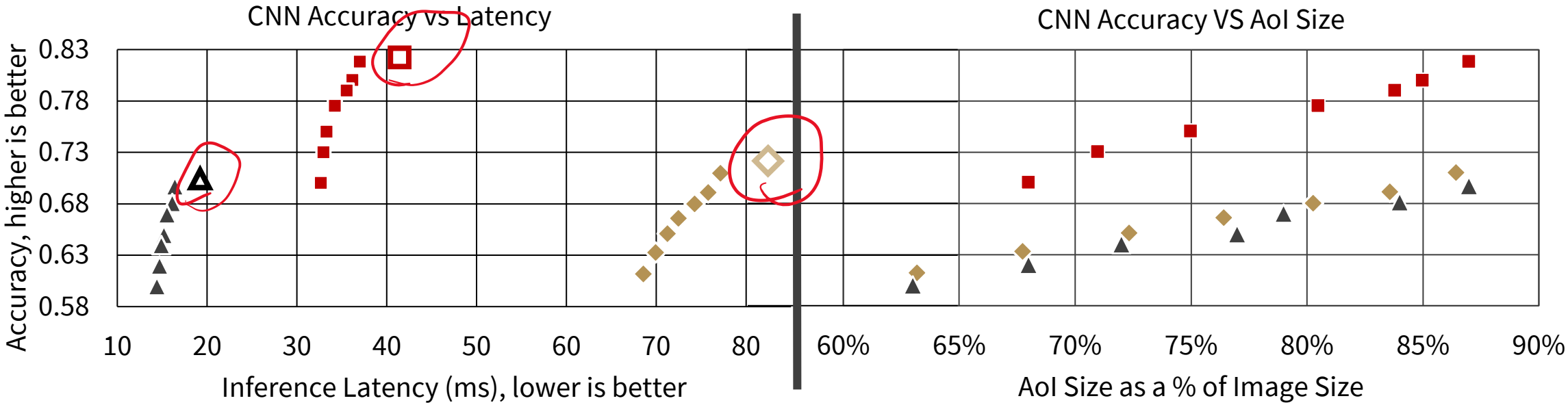
ConvNeXt-T: 42ms

fConvNeXt-T: 37ms

VGG-16: 83ms

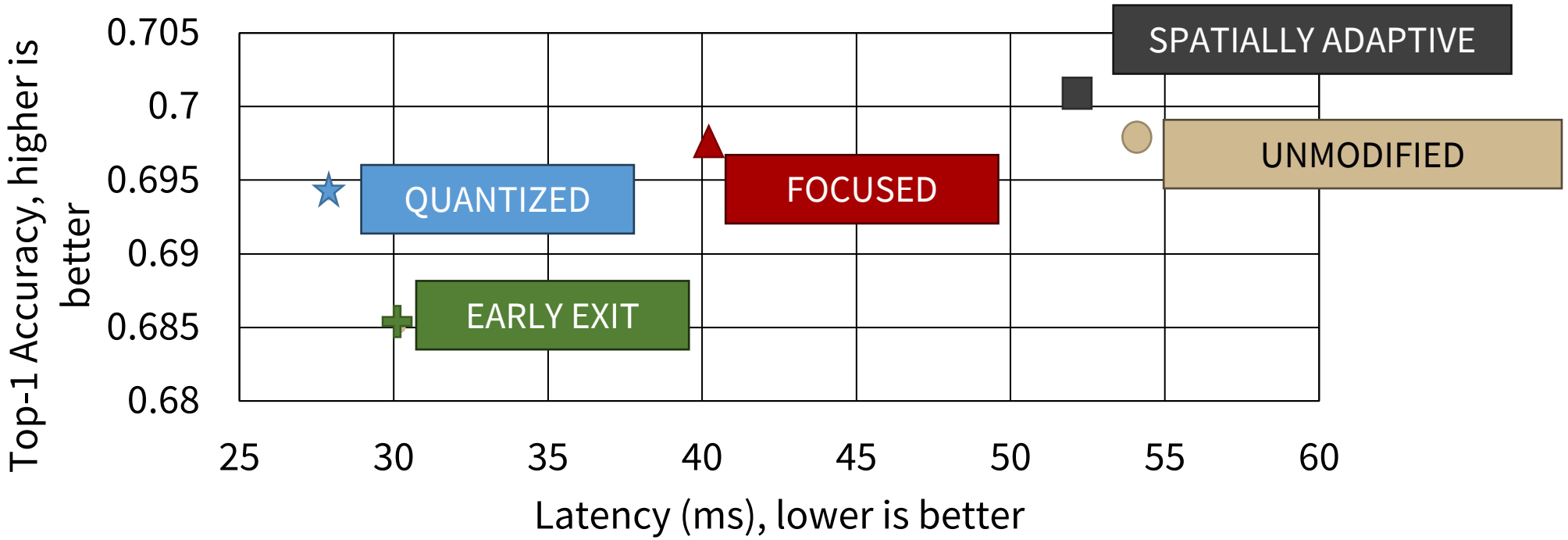
fVGG-16: 77ms

Circled points are unmodified



AMD

fResNet-18 VS State-of-the-Art ResNet-18 modifications



An automated approach for improving the inference latency and energy efficiency of pretrained CNNs by removing irrelevant pixels with focused convolutions

Caleb Tung, Nicholas Eliopoulos, Purvish Jajal, Gowri Ramshankar, Chen-Yun Yang (Purdue Univ., USA), Nicholas Synovic (Loyola Univ. Chicago, USA), Xuecen Zhang, Vipin Chaudhary (Case Western Reserve Univ., USA), George K. Thiruvathukal (Loyola Univ. Chicago, USA), Yung-Hsiang Lu (Purdue Univ., USA)

PURDUE UNIVERSITY
Elmore Family School of Electrical and Computer Engineering

LOYOLA UNIVERSITY CHICAGO
DEPARTMENT OF COMPUTER SCIENCE

CASE WESTERN RESERVE UNIVERSITY
CASE SCHOOL OF ENGINEERING