# Differential Analysis on Adversarial Neural Network Executions

研究生：陳怡君
指導教授：郁方 博士

# CONTENT


01 Introduction

02 Methodology

03 Keras Application Analysis

04 Object Detection Analysis

05 Conclusion

# Part 01

## Introduction

# Motivation

- Image recognition and object detection have been widely adopted in practice. [1], [2], [3]
- On the other hand, several machine learning models have been found vulnerable to **adversarial examples**. One can generate adversarial samples to twist the results of these applications. [4]
- Having a systematic approach to identify adversarial samples is essential.
- Our goal is to detect adversarial samples from the perspective of program execution.

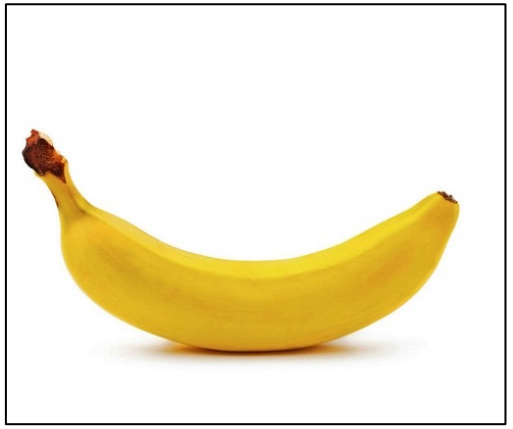
[1] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database" , 2009
[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E, " Imagenet classification with deep convolutional neural networks.", 2012
[3] J. Redmon , A. Farhadi, "YOLOv3: An Incremental Improvement", 2018
[4 ]Baluja, S., and Fischer, I. , "Learning to attack: Adversarial transformationnetworks.", 2018

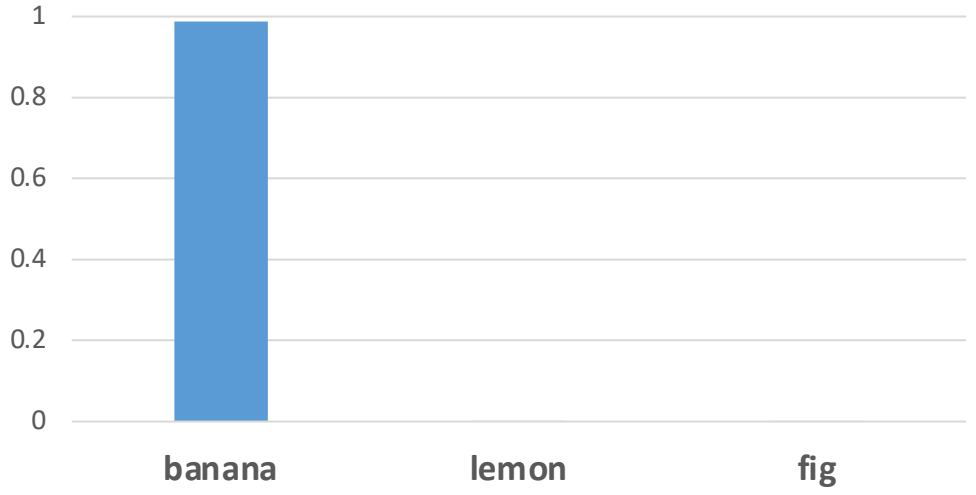
# Objective

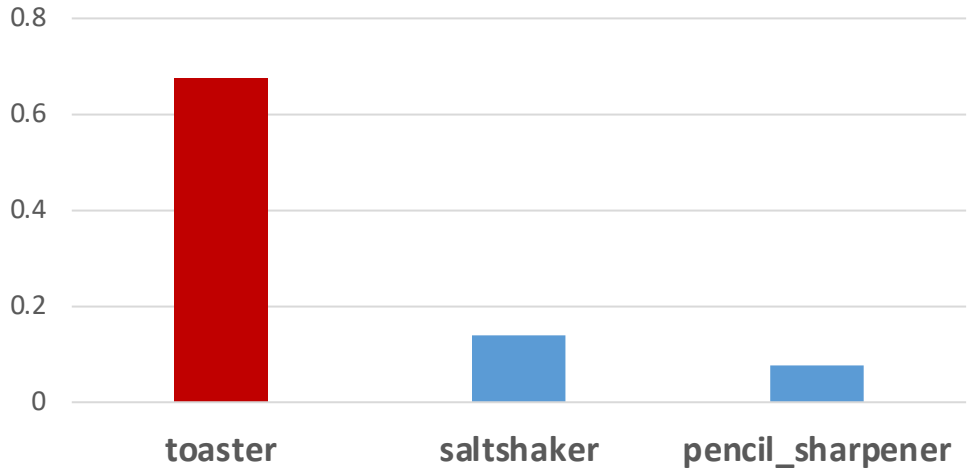

**Find the difference of program execution**

# Adversarial Patch

**Toaster**

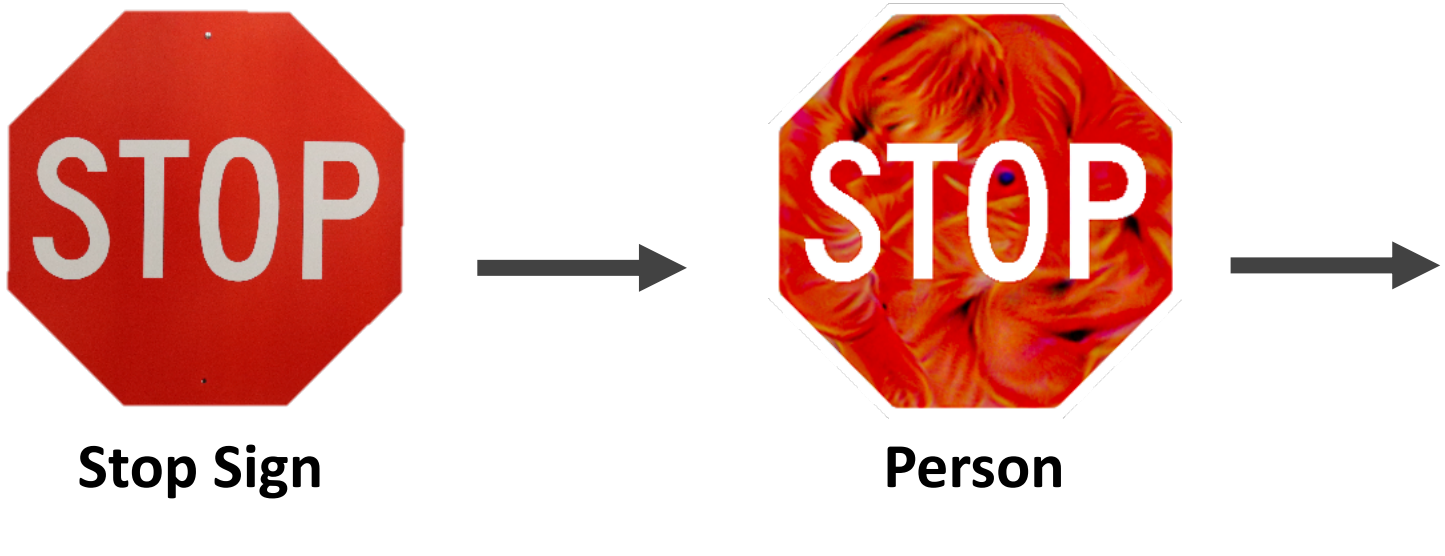





Brown, "Adversarial patch.", 2017.

# ShapeShifter



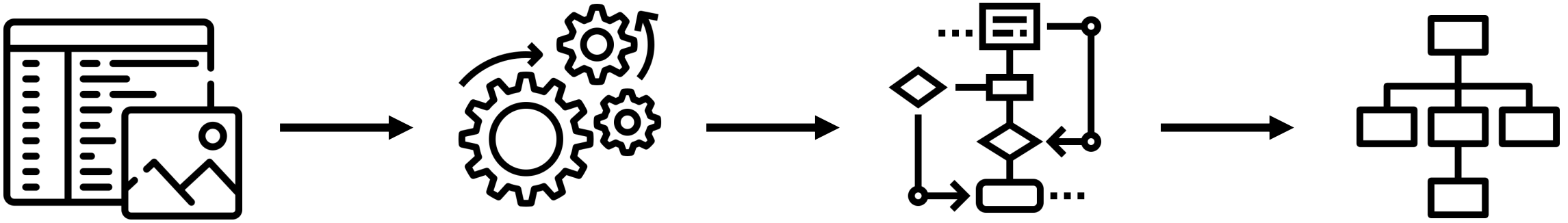

**Stop Sign** **Person**

Shang-Tse Chen, "ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector.", 2019.

# Part 02

## Methodology

# System Framework



# Python Profiling

- **An example of recorded program execution:**

```
  #Start#3#137940 validate_file()    keyword argument count = 1,1-th kw:key(str)='algorithm',value(str)='auto'
    #Start#4#137941 _hash_file()    keyword argument count = 0
    *End _hash_file() return:(str:64373286793e3c8b2b4e3219cbf3544b)   CostTime:1515000.0000000000
  *End validate_file() return:(bool:True)   CostTime:1515000.0000000000
*End get_file() return:(str:C:\Users\sosclubnccu\.keras\models\vgg16_weights_tf_dim_ordering_tf_kernels.h5)   CostTime:1515000.0000000000
#Start#2#137942 load_weights()    keyword argument count = 0
  #Start#3#137943 filename_encode()   @1arg:(str)C:\Users\sosclubnccu\.keras\models\vgg16_weights_tf_dim_ordering_tf_kernels.h5
    #Start#4#137944 fspath()   @1arg:(str)C:\Users\sosclubnccu\.keras\models\vgg16_weights_tf_dim_ordering_tf_kernels.h5
    *End fspath() return:(str:C:\Users\sosclubnccu\.keras\models\vgg16_weights_tf_dim_ordering_tf_kernels.h5)   CostTime:0.0000000000
  *End filename_encode() return:(bytes)   CostTime:0.0000000000
  #Start#3#137945 make_fcpl()    keyword argument count = 1,1-th kw:key(str)='track_order',value(bool)='False'
  *End make_fcpl() return:(NoneType)   CostTime:0.0000000000
  #Start#3#137946 make_fid()    keyword argument count = 2,1-th kw:key(str)='fcpl',value(NoneType),2-th kw:key(str)='swmr',value(bool)='Fa
  *End make_fid() return:(h5py.h5f.FileID)   CostTime:0.0000000000
  #Start#3#137947 __init__()   @1arg:(File)@2arg:(h5py.h5f.FileID)
  *End __init__() return:(NoneType)   CostTime:0.0000000000
```

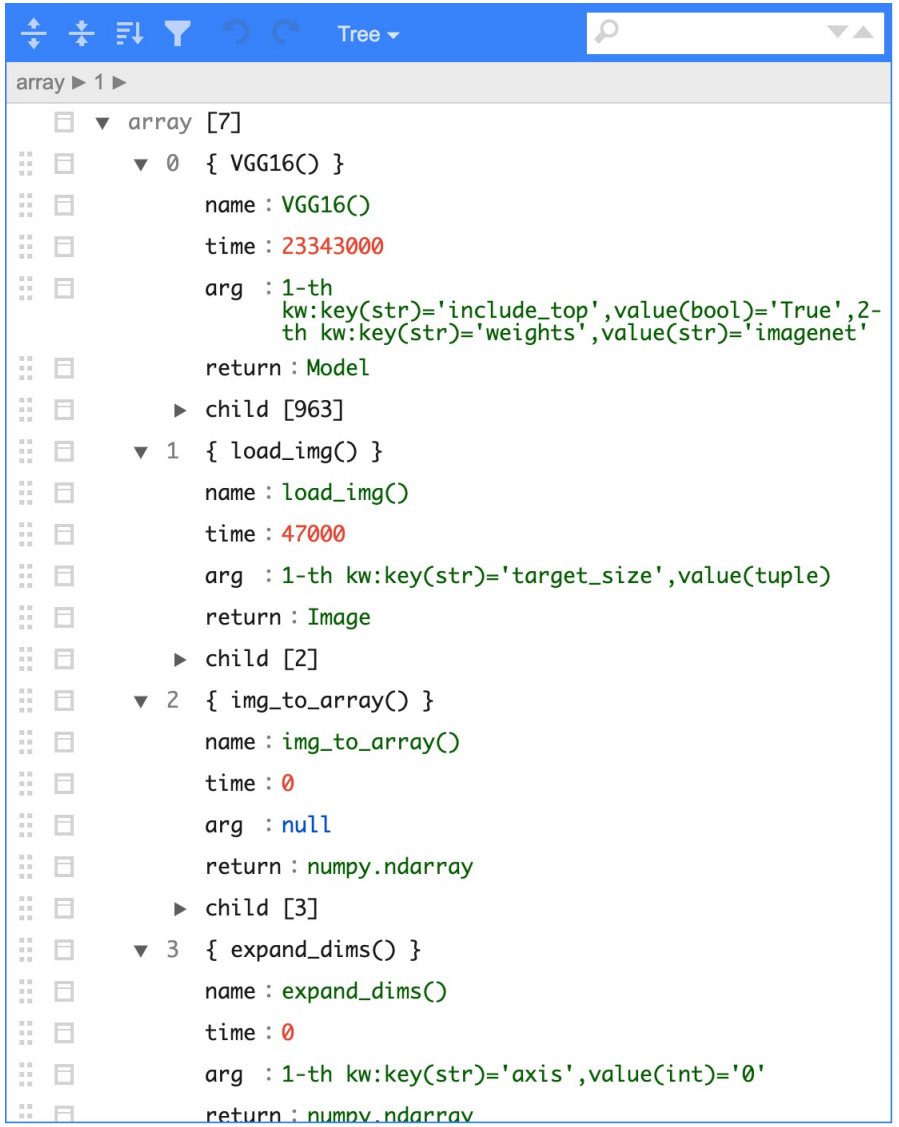


Y.H. Lin, S.W. Hsiao, F. Yu, "Runtime profiling on python calls of python scripts", TCSE 2020.

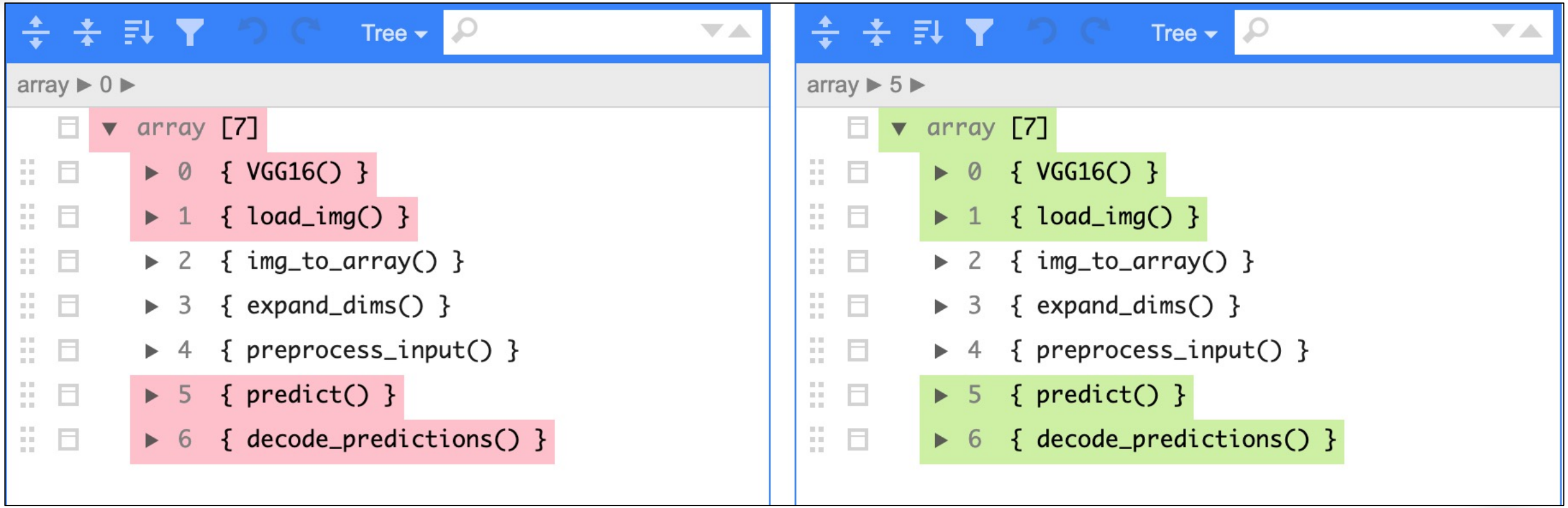
# Layer Program Execution

We implement a parser to synthesize execution in layers with:

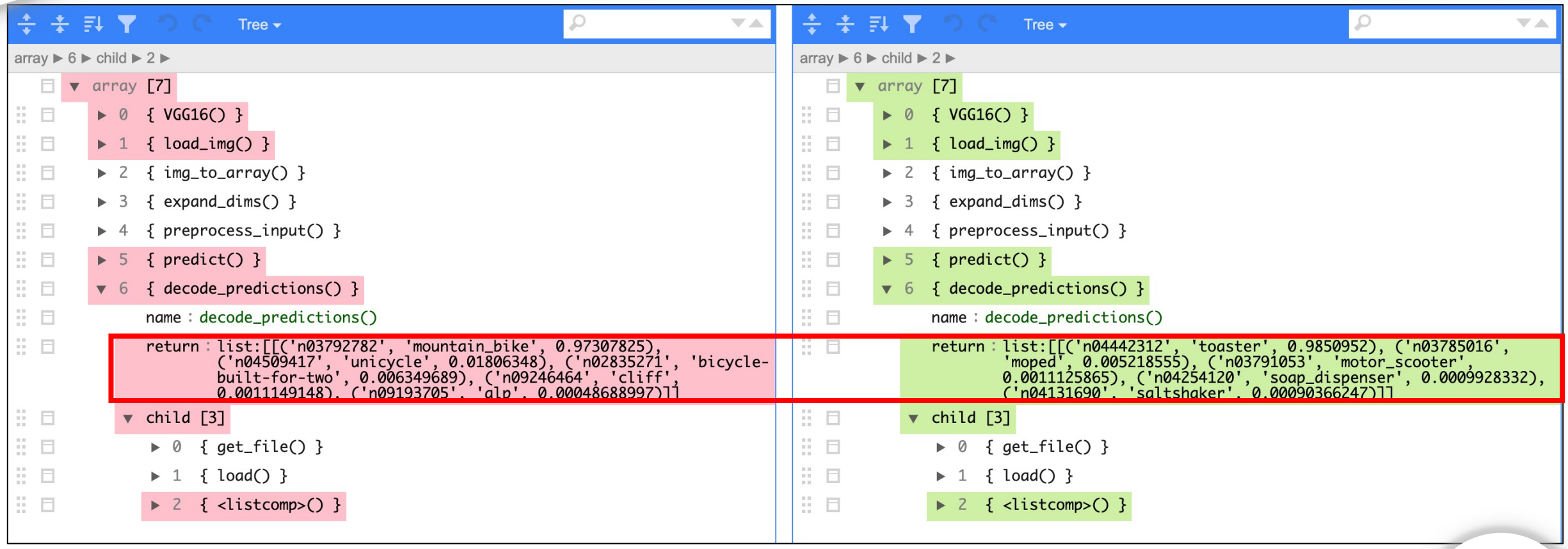
- Function Name
- Input Argument
- Execution Time
- Return Value



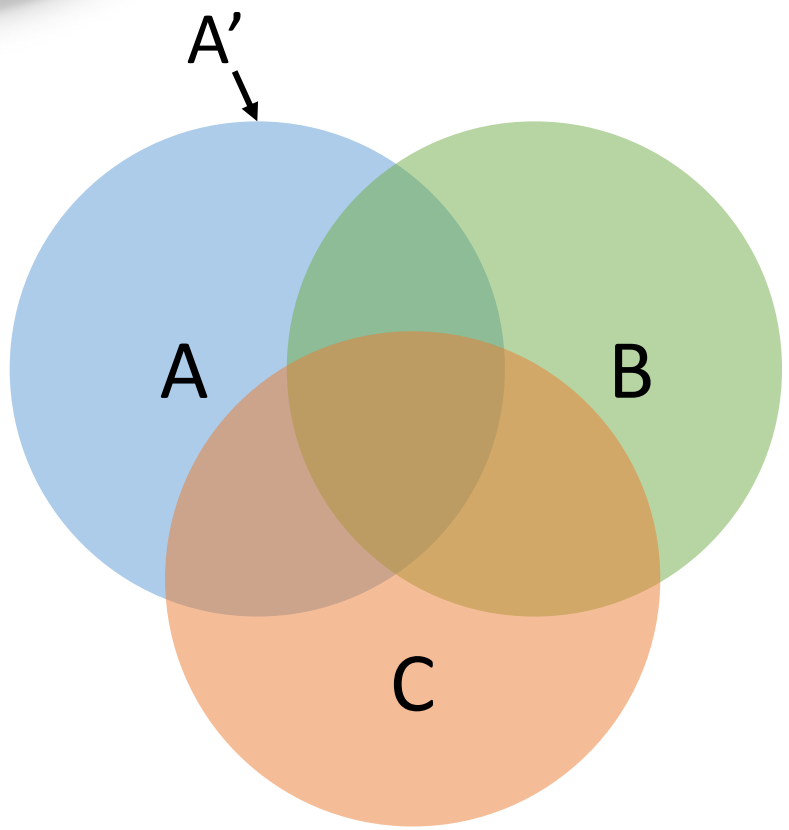
# Find and Visualize Difference



# Find and Visualize Difference



# Differential Analysis




A = conjunction of { diff between original and attack executions }

B = conjunction of { diff between original and original executions }

C = conjunction of { diff between attack and attack executions }

```
if (A is empty) :
        No difference.
else :
        A' = A - (B ∪ C)
        if (A' is empty) :
                No identifiable difference.
        else :
                Use A' as the rule to detect
                adversarial samples.
```

# Differential Analysis

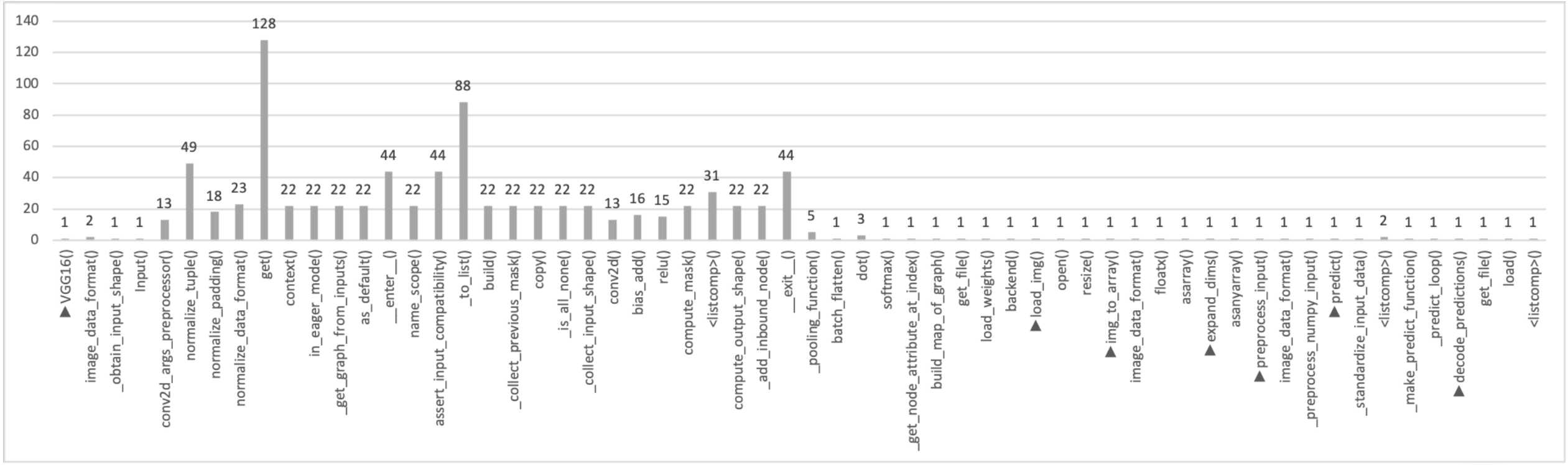
Difference Observations:

- Call Count on Different Calls
- Call Count on Different Execution Time
- Call count on Different Return Values

**org_cat**

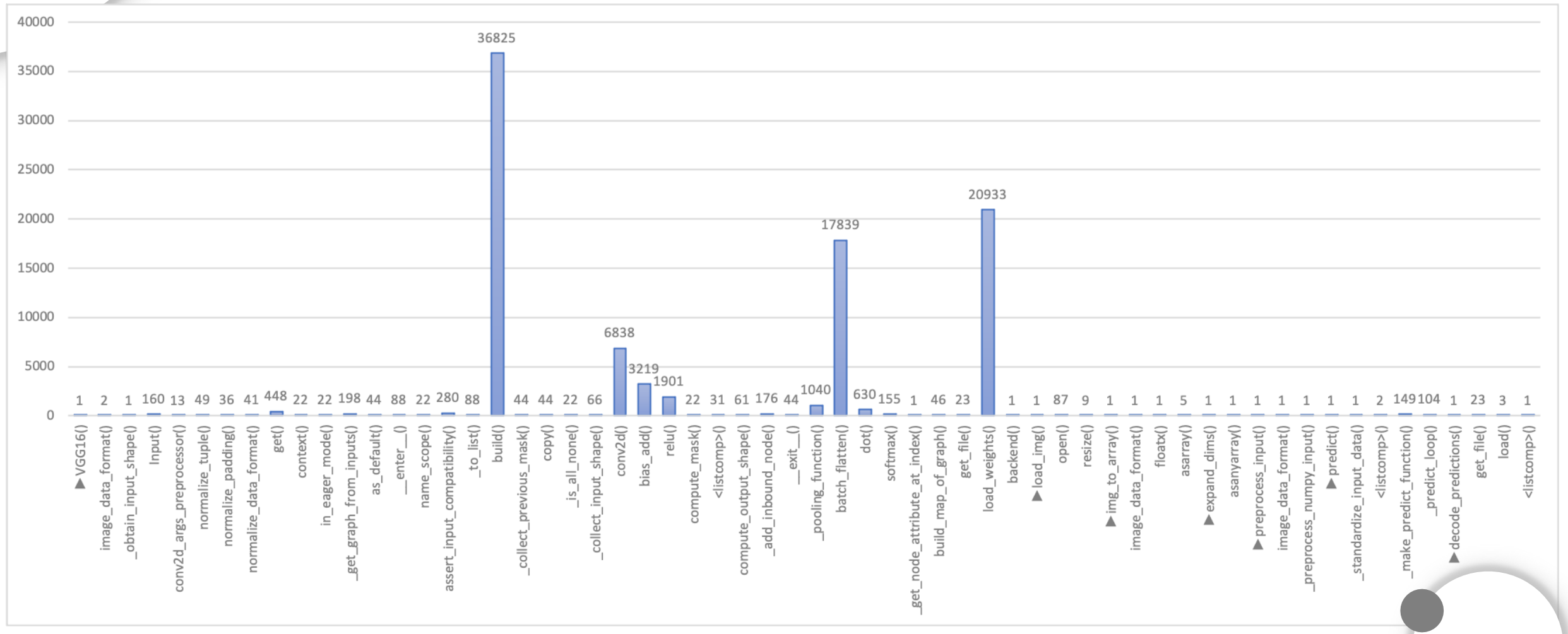
**adv_cat**

# Sequential Call Count Observation (the first two layers )

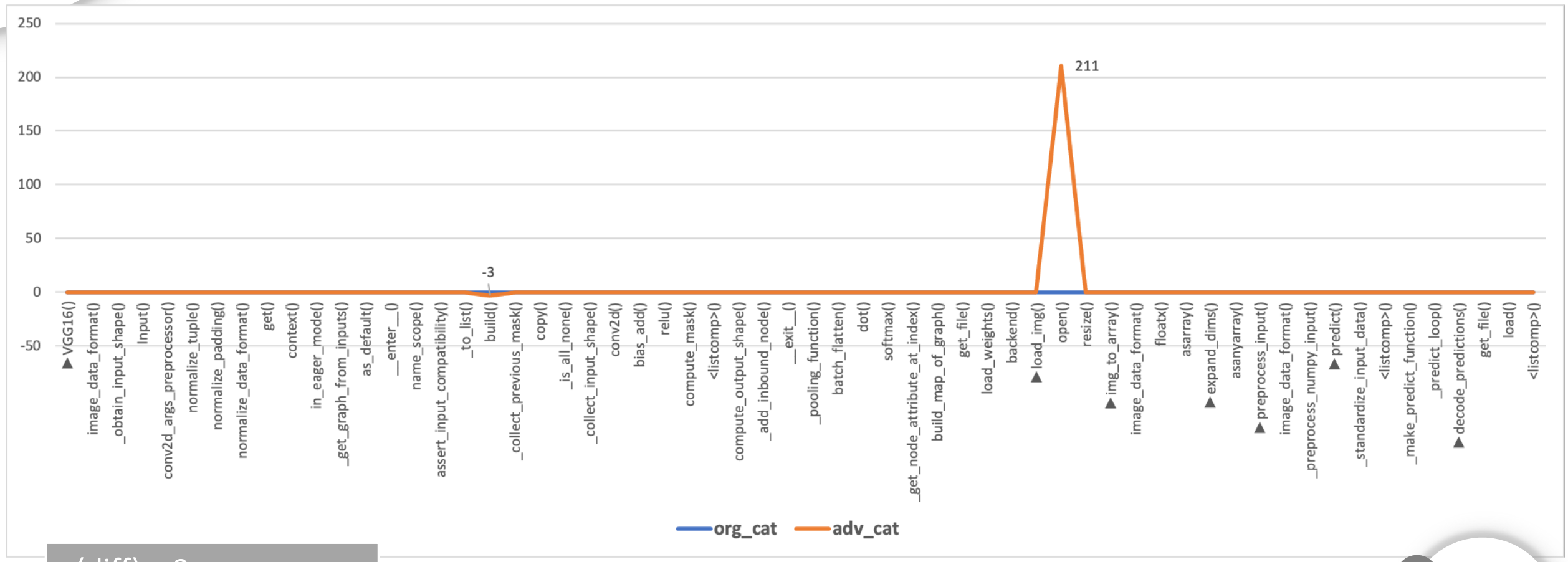



Under the VGG16() function

# Sequential Call Count Observation

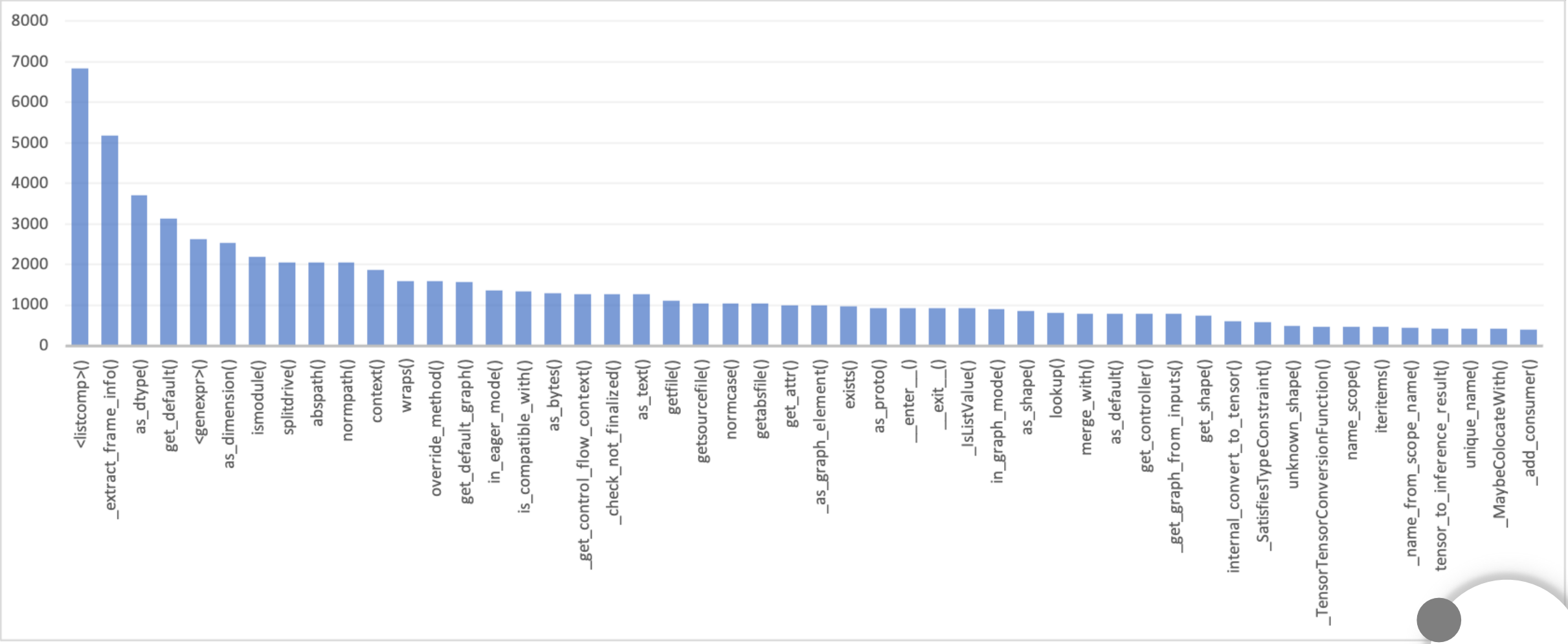


# Difference on Sequential Call Counts

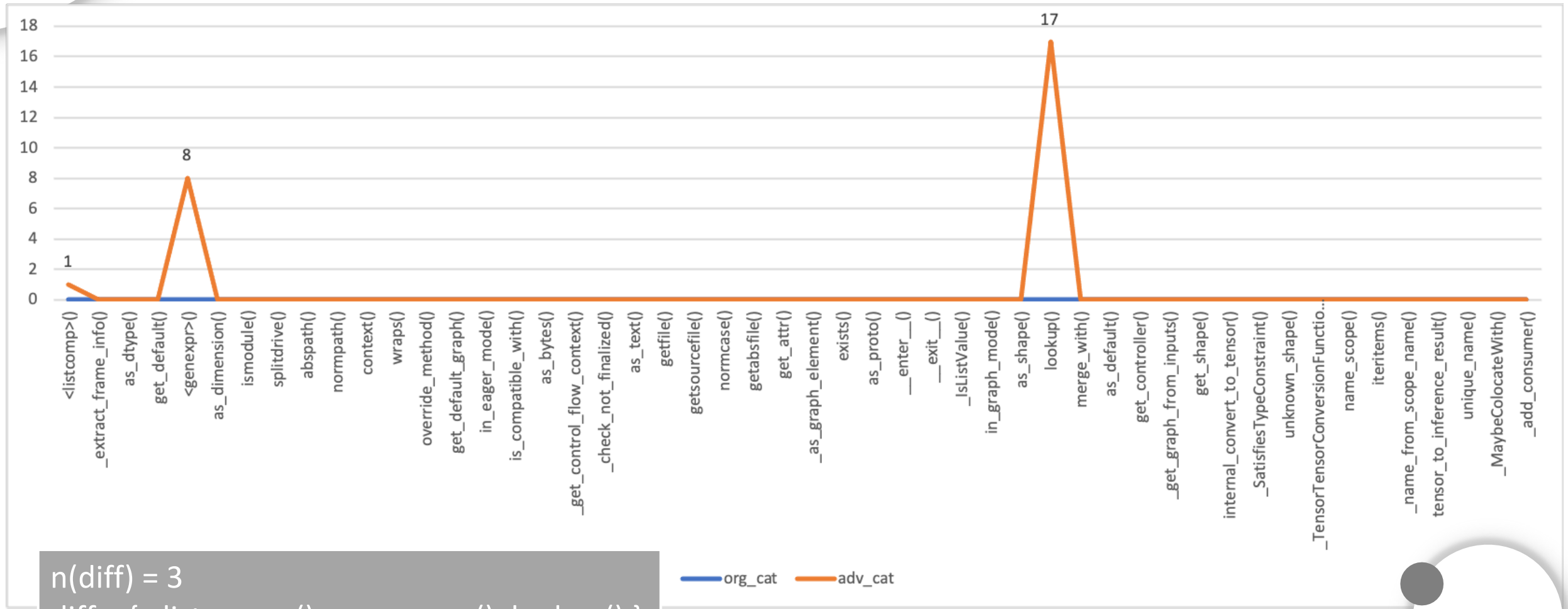



n(diff) = 2
diff = { build(), open() }

# Top50 Call Count Observation

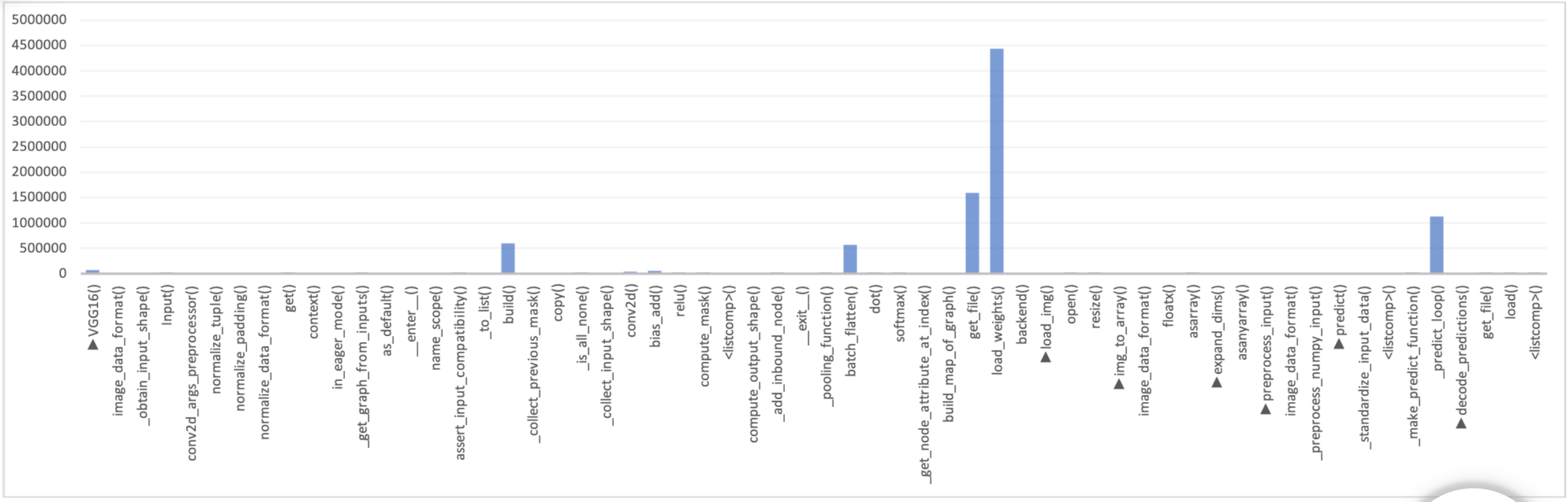


# Difference on Top50 Call Counts

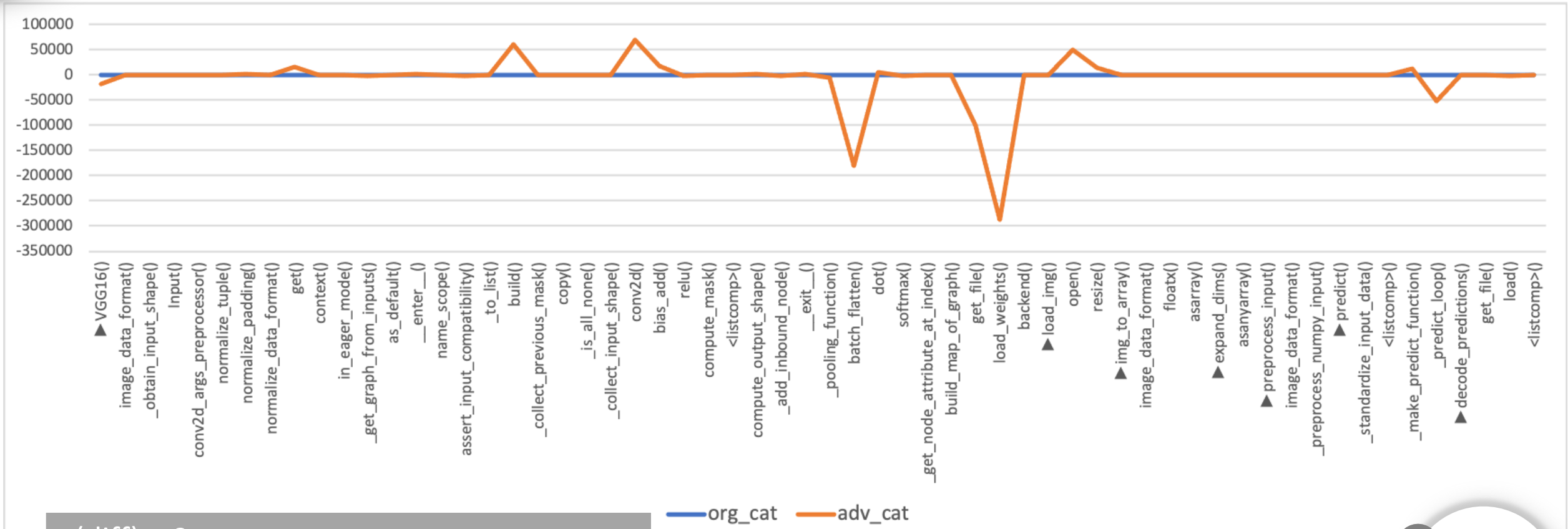



n(diff) = 3
diff = { <listcomp>(), <genexpr>(), lookup() }

# Sequential Execution Time Observation

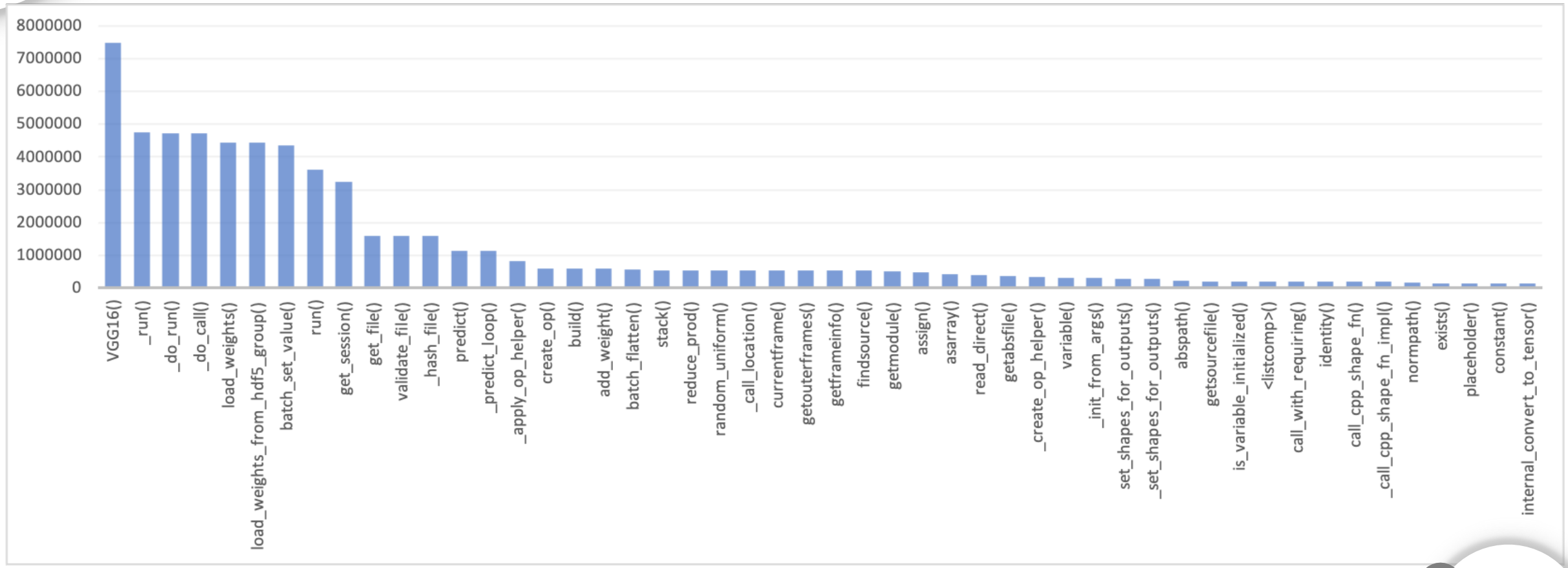


# Difference on Sequential Execution Time

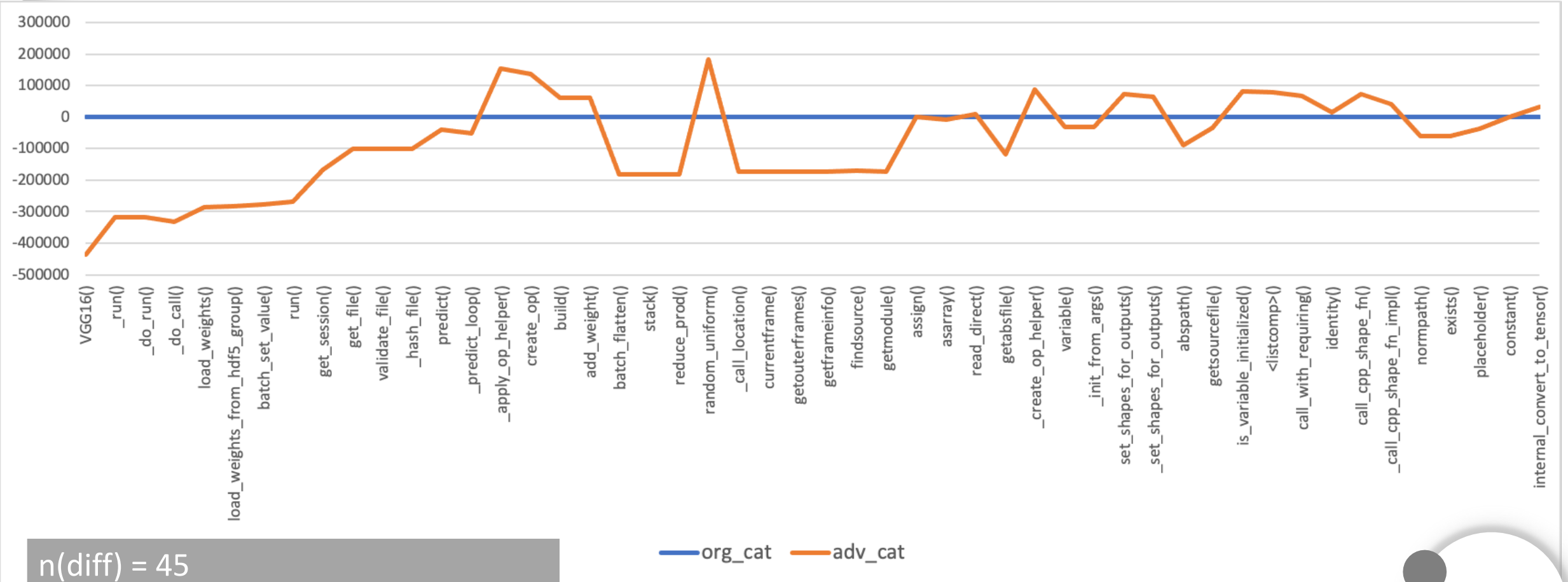



n(diff) = 9
diff = { VGG16(), get(), build(), conv2d(), … }

# Top50 Execution Time Observation



# Difference on Top50 Execution Time




n(diff) = 45
diff = { VGG16, _run(), _do_run(), … }

# Return Value Observation

- Distance of integer values :

  Euclidean Distance : $dist(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

- Distance of string values :

  Levenshtein Distance : $$\operatorname{lev}_{X,Y}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(X_i \neq Y_j)} \end{cases} & \text{otherwise.} \end{cases}$$

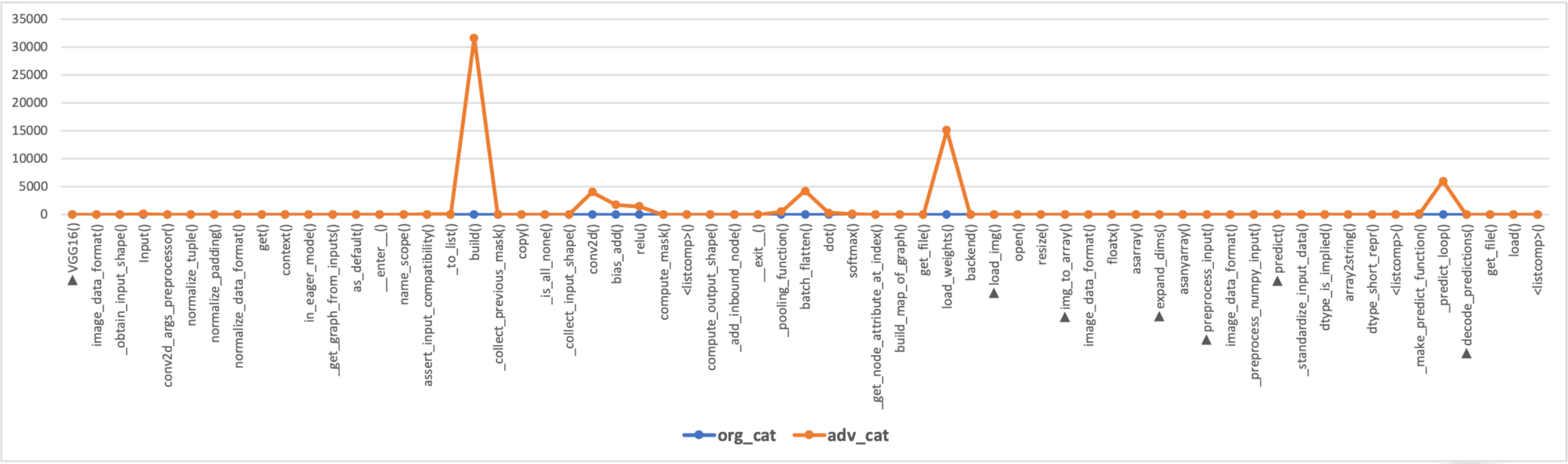
- Distance of boolean values:

  Unit Distance: $if(X \neq Y)\ dist(X,Y) = 1$

- Distance of matrix values :

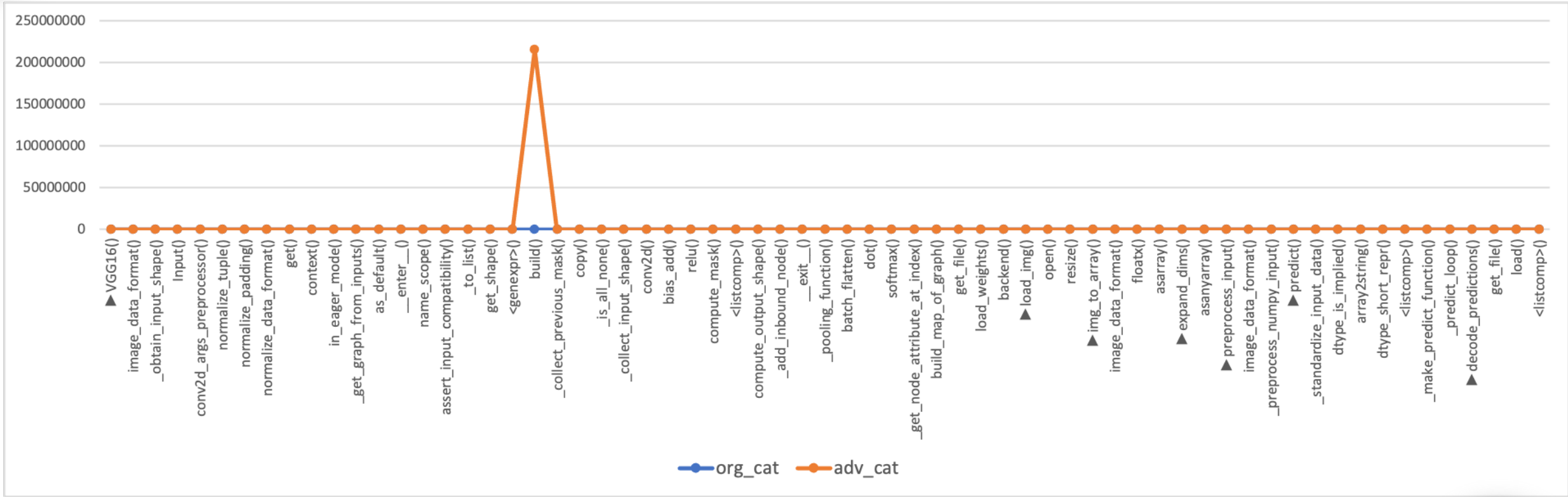
  Euclidean Distance : $dist(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

# Difference on Sequential Call Count with Return Values



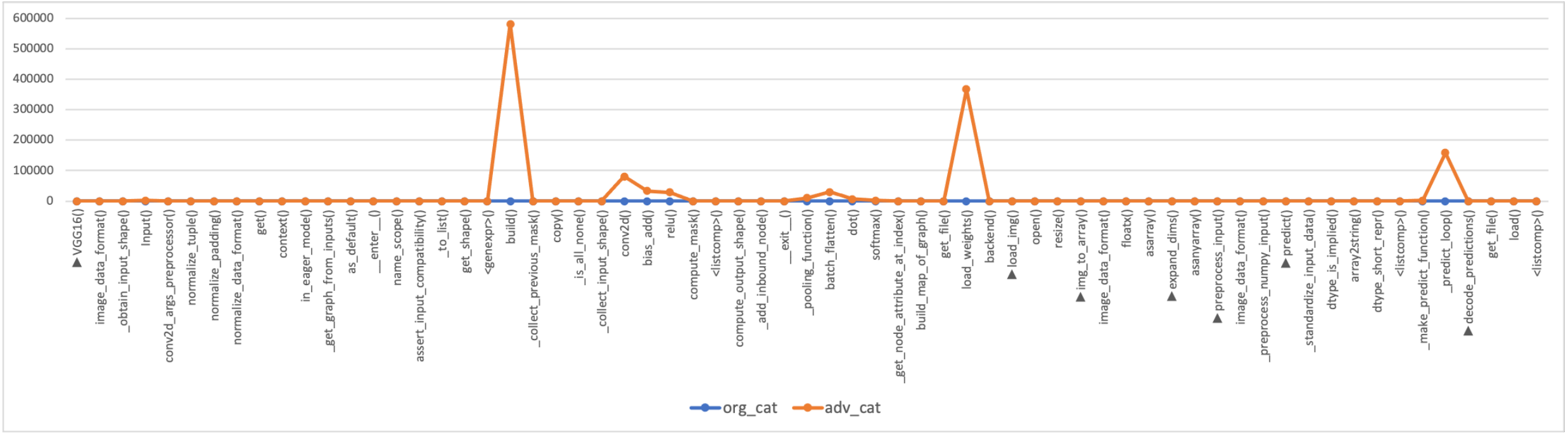

n(diff) = 7
diff = { build(), conv2d(), bias_add(), relu(), … }

# Difference on Sequential Integer Distance



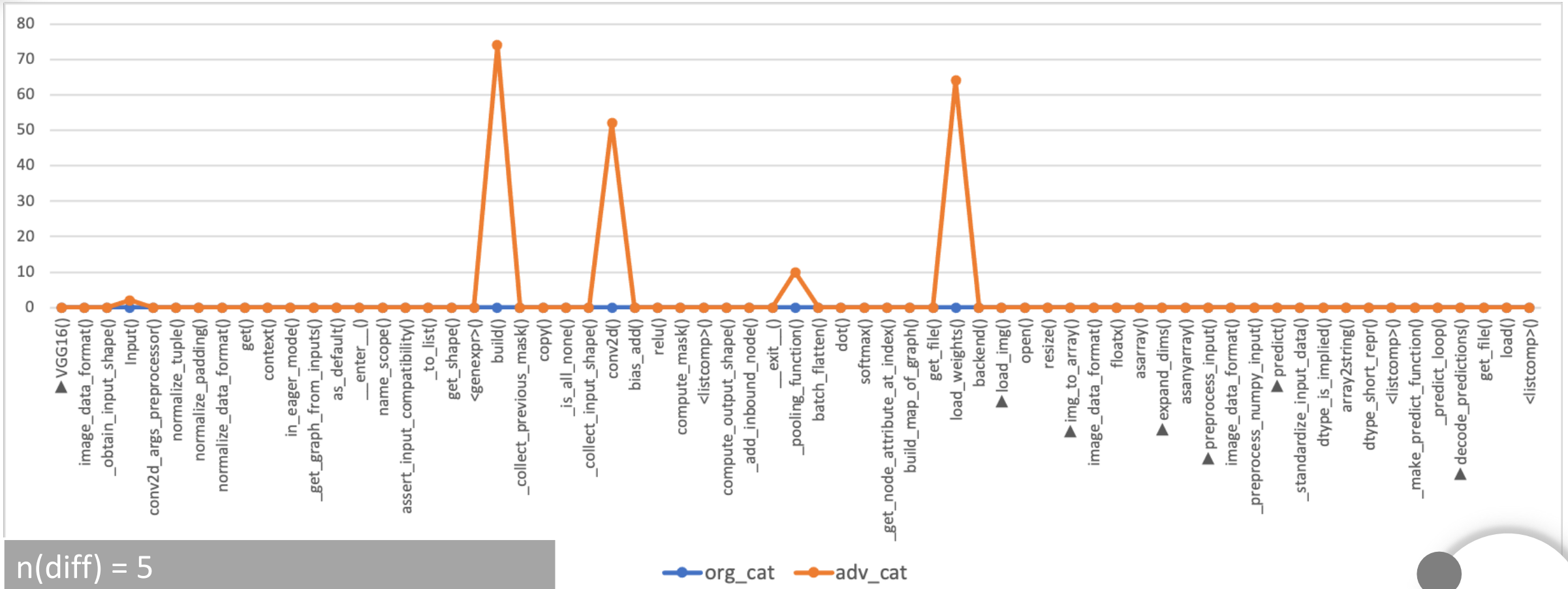

n(diff) = 1
diff = { build() }

# Difference on Sequential String Distance



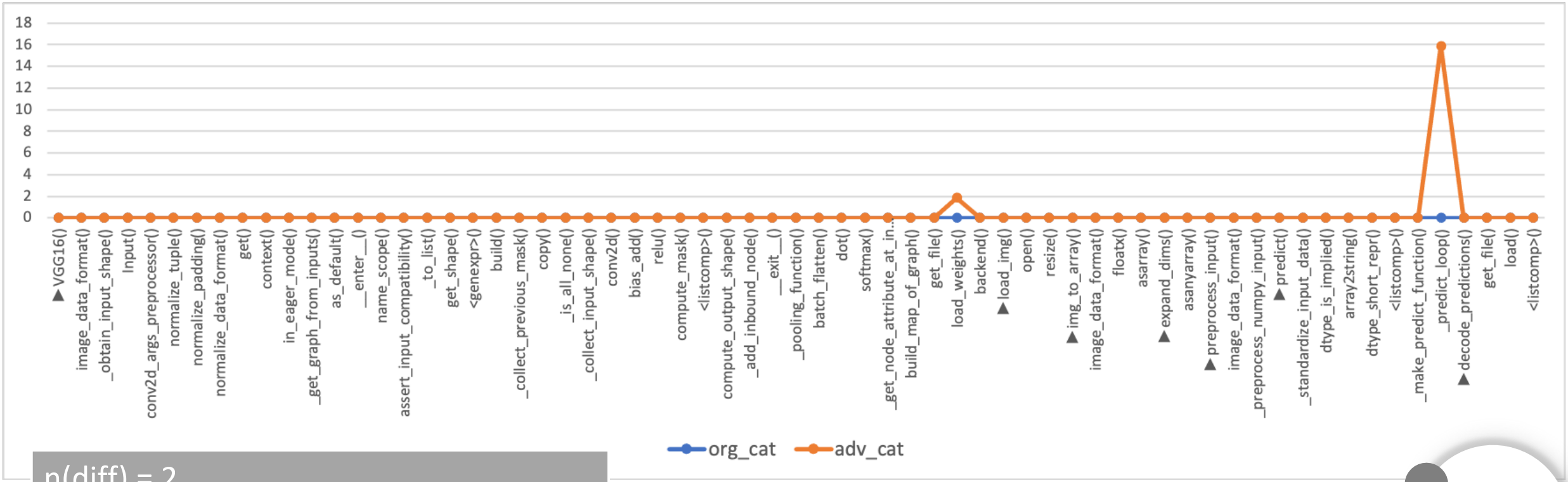

n(diff) = 7
diff = { build(), conv2d(), bias_add(), relu(), … }

# Difference on Sequential Boolean Distance




n(diff) = 5
diff = { Input(), build(), conv2d(), … }

# Difference on Sequential Matrix Distance



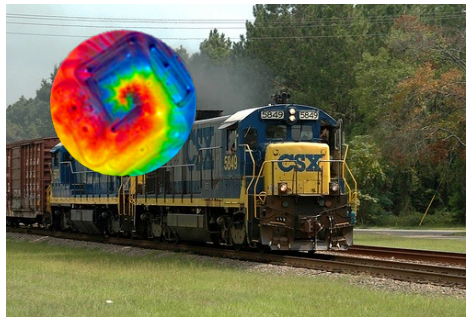

n(diff) = 2
diff = { load_weights(), _predict_loop() }

# Part 03

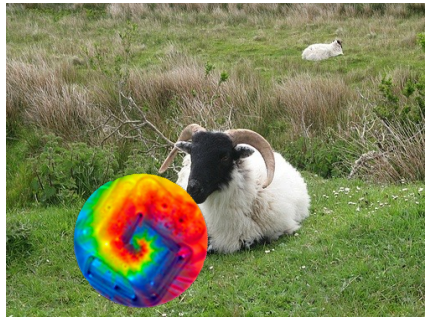
## Keras Applications Analysis

# Craft Adversarial Samples

Car O Toster X



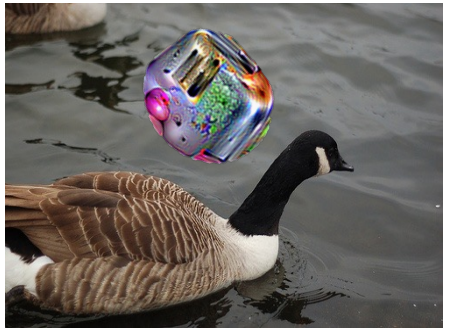

Train O Toster X



Goat O Toster X

Plain O Toster X


Bus X Toster O

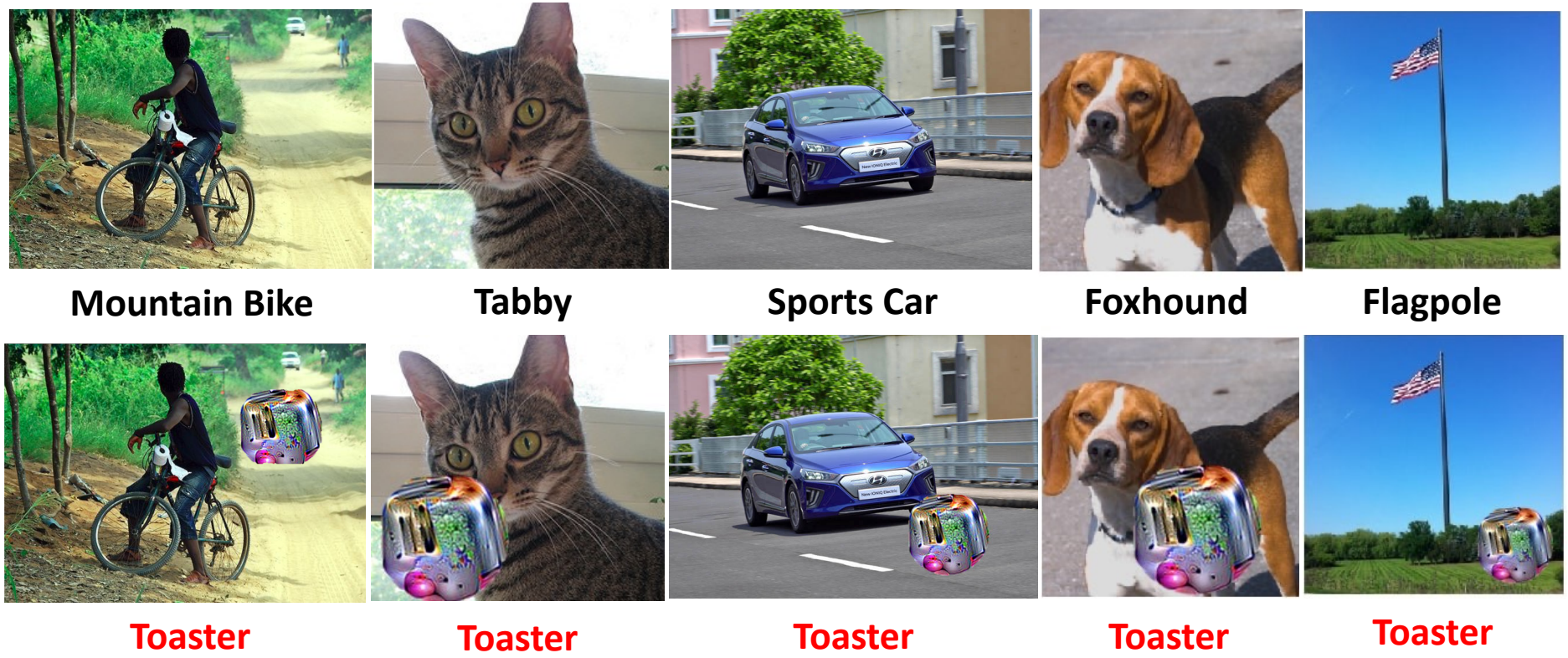


Duck X Toster O

# Craft Adversarial Samples

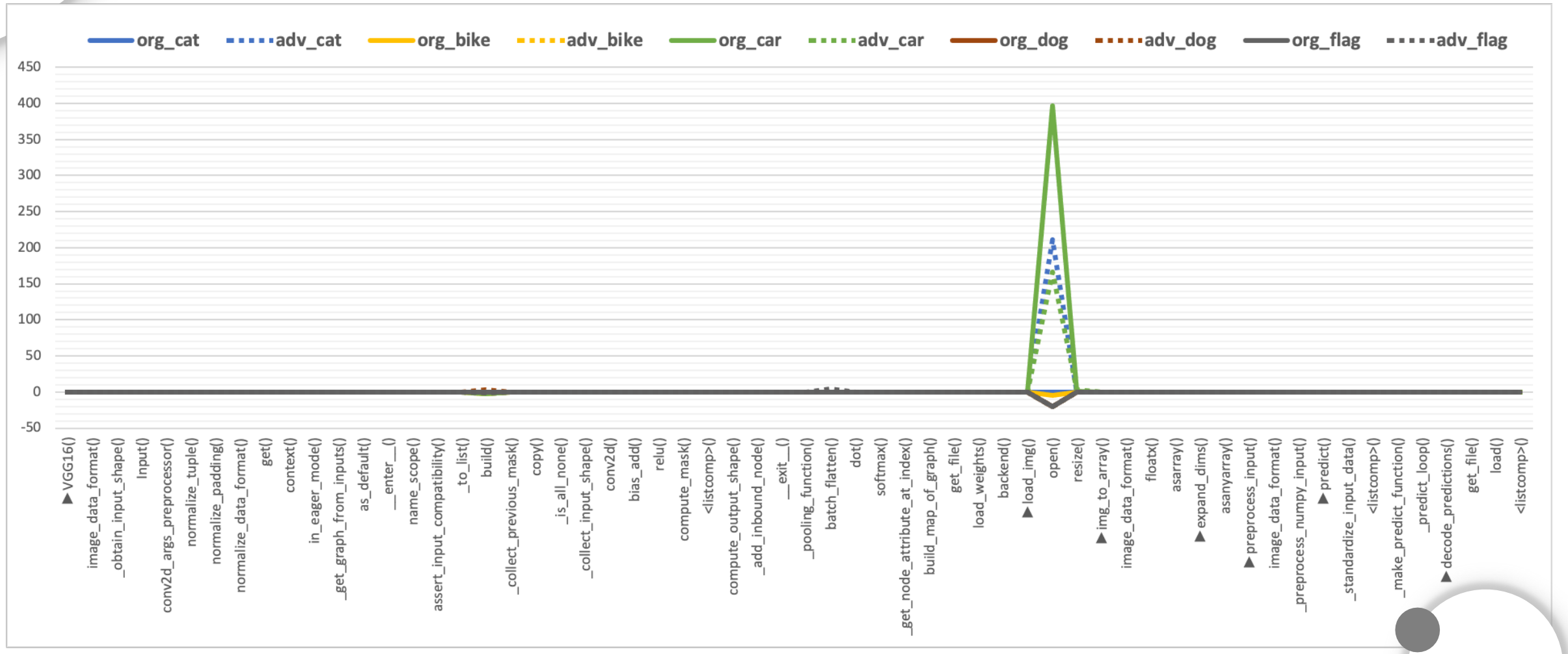
可攻擊五種模型：

- VGG16
- VGG19
- InceptionV3
- Xception
- Resnet50



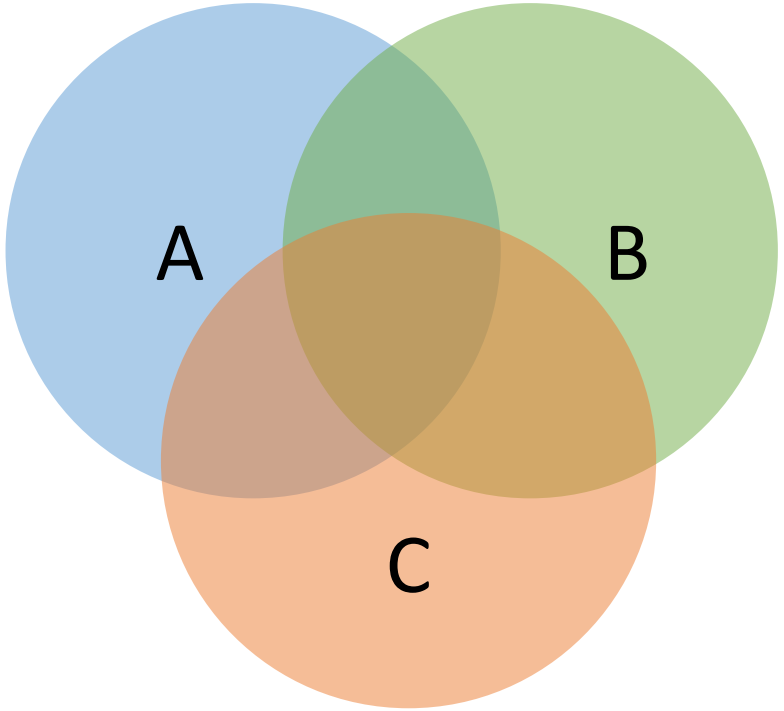
VGG16

# Difference on Sequential Call Counts



VGG16

# Difference on Sequential Call Counts




A = conjunction of { diff between original and attack }

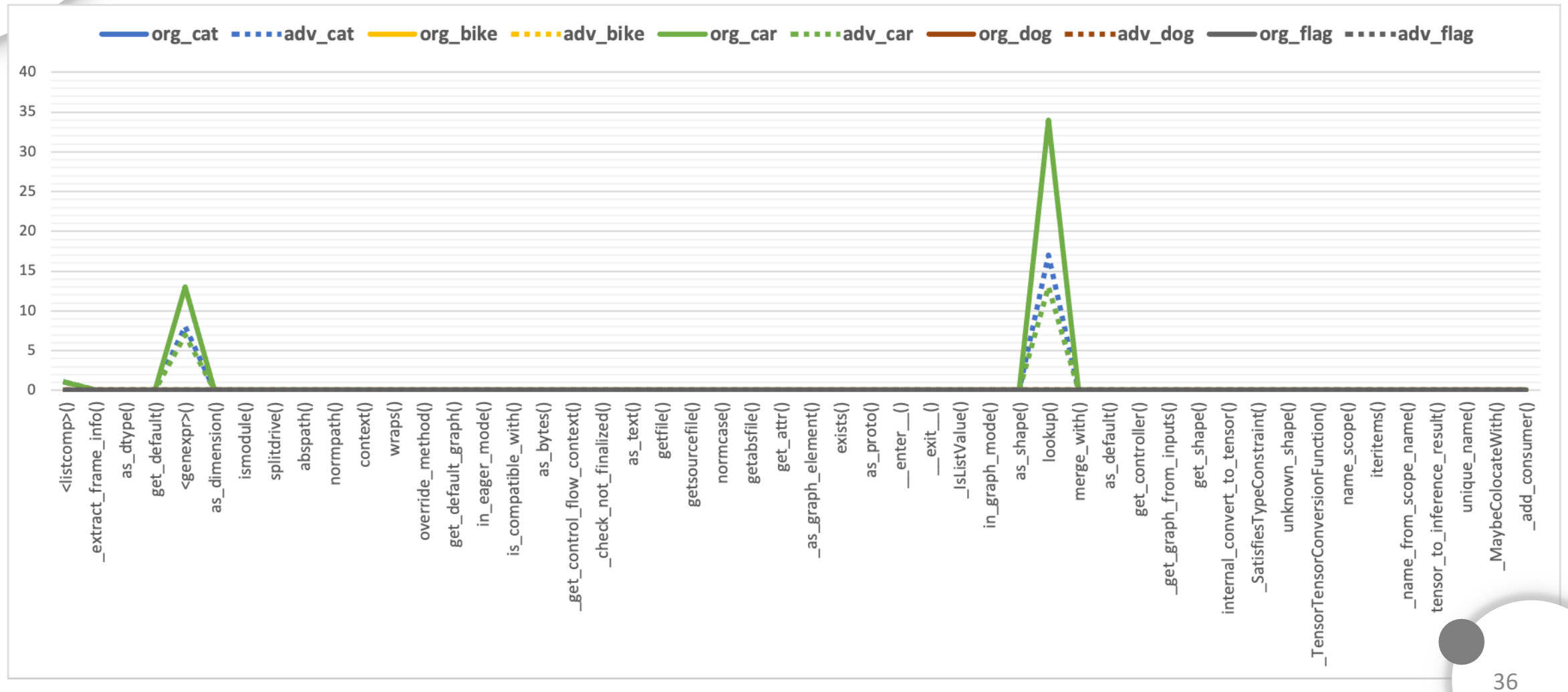
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 1
n(B) = 1
n(C) = 1
n(A') = 0
∴ No identifiable difference.

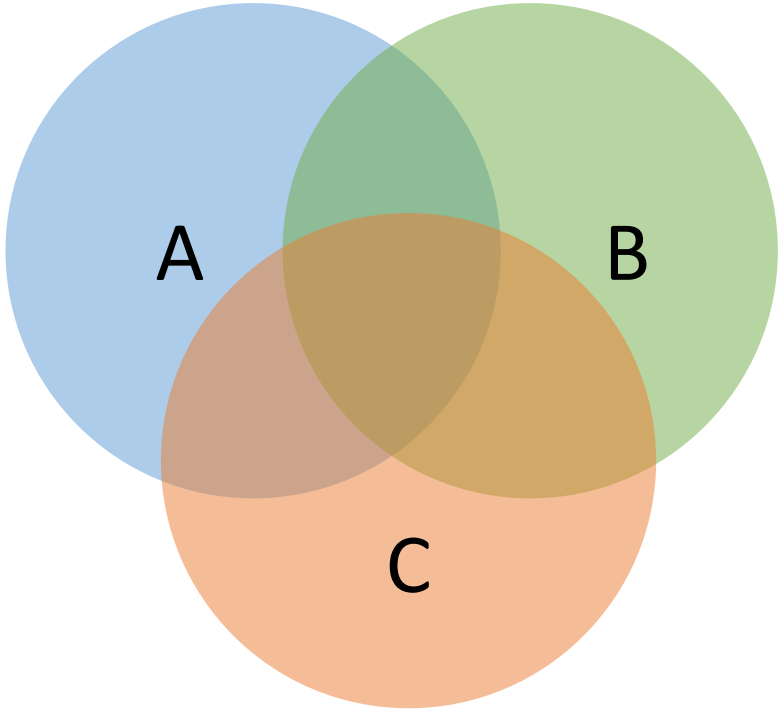
VGG16

# Difference on Top50 Call Counts



VGG16

# Difference on Top50 Call Counts




A = conjunction of { diff between original and attack }

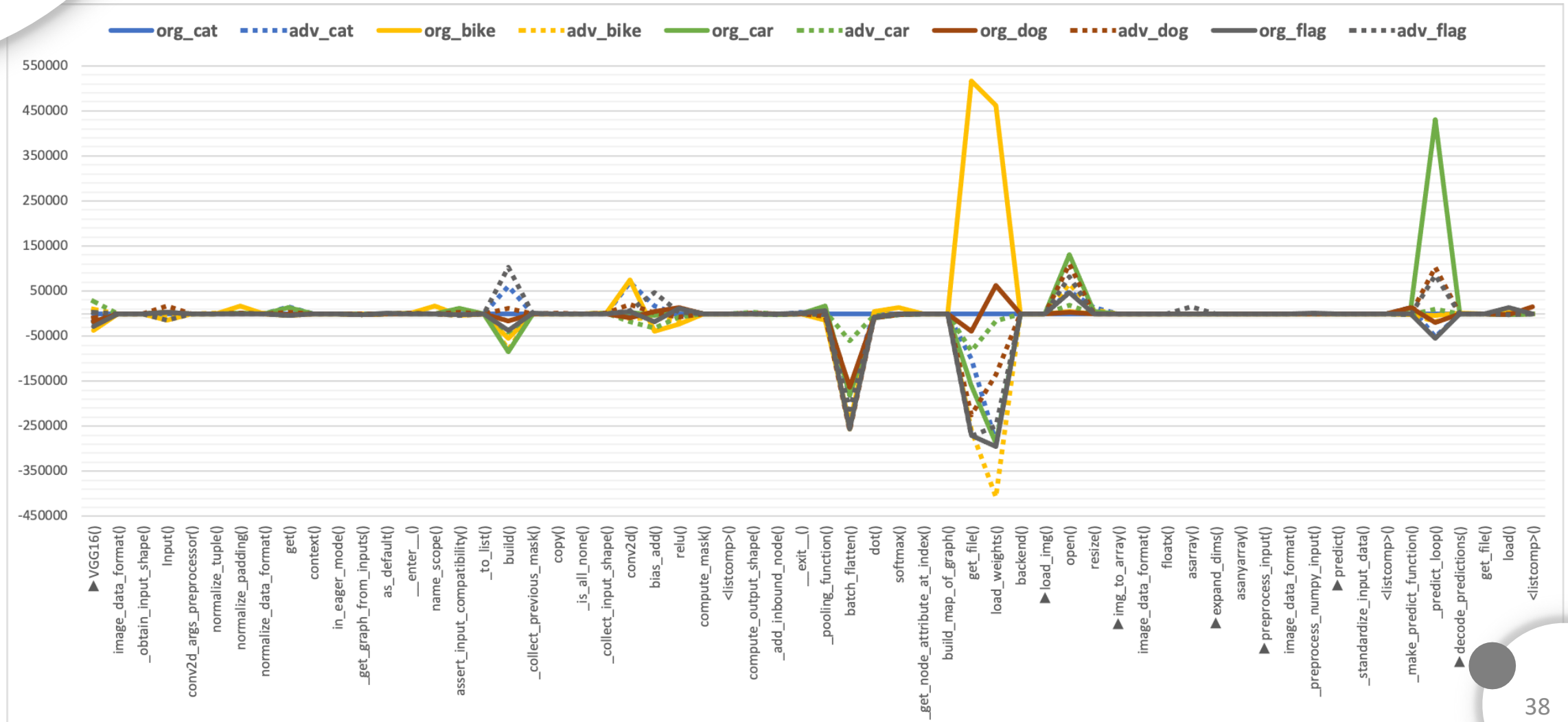
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 0
n(B) = 0
n(C) = 0
∴ No difference.

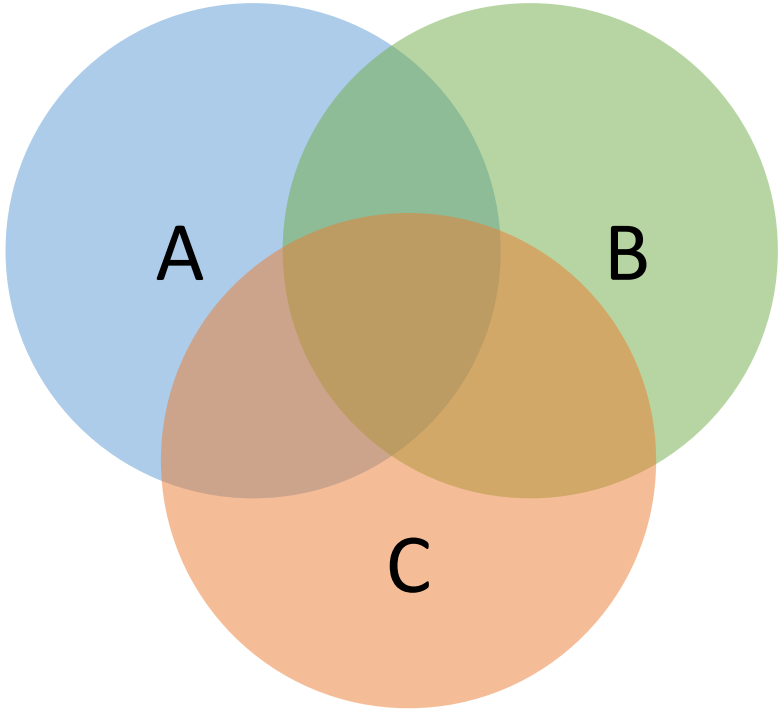
VGG16

# Difference on Sequential Execution Time



VGG16

# Difference on Sequential Execution Time




A = conjunction of { diff between original and attack }

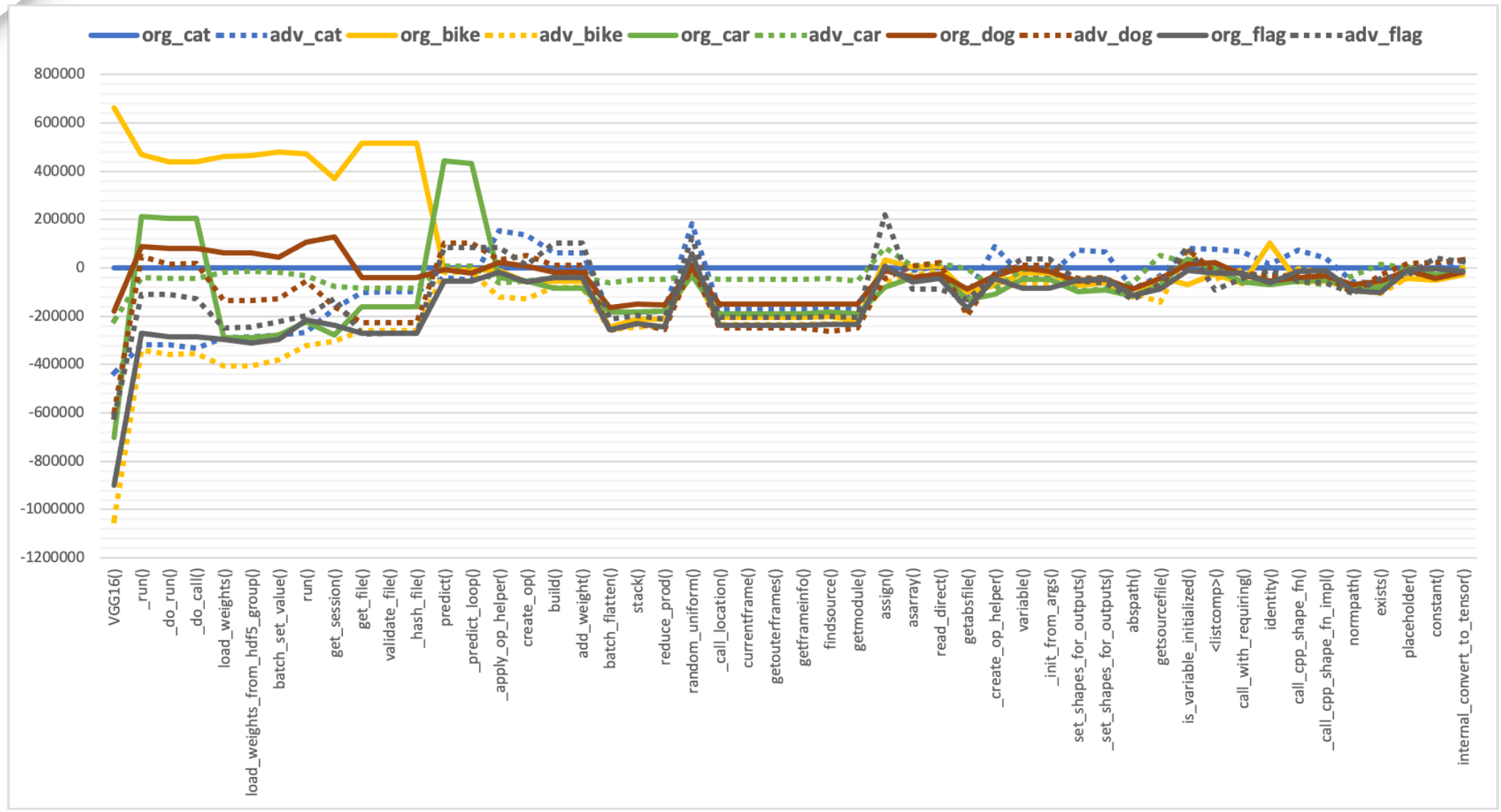
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 17
n(B) = 22
n(C) = 18
n(A') = 0
∴ No identifiable difference.

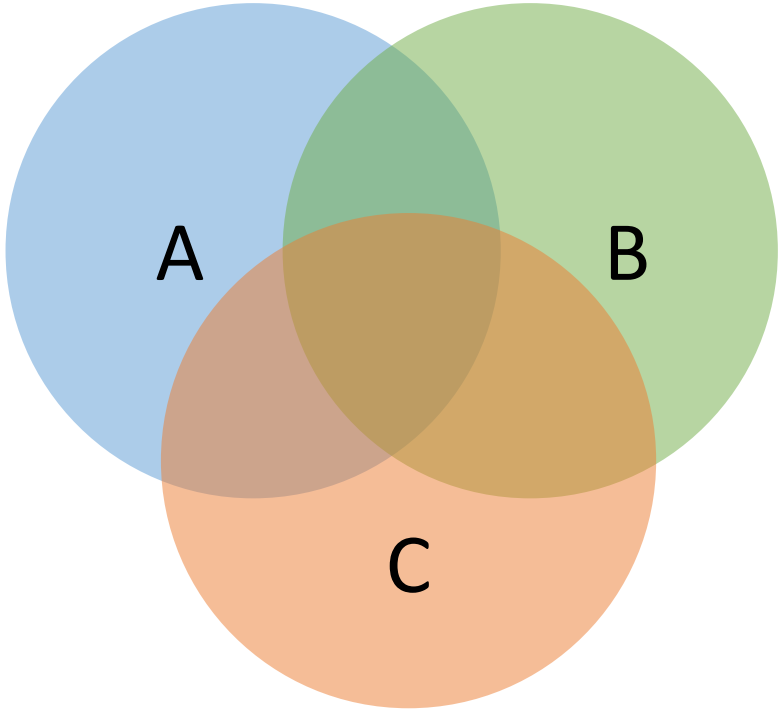
VGG16

# Difference on Top50 Execution Time



VGG16

# Difference on Top50 Execution Time




A = conjunction of { diff between original and attack }

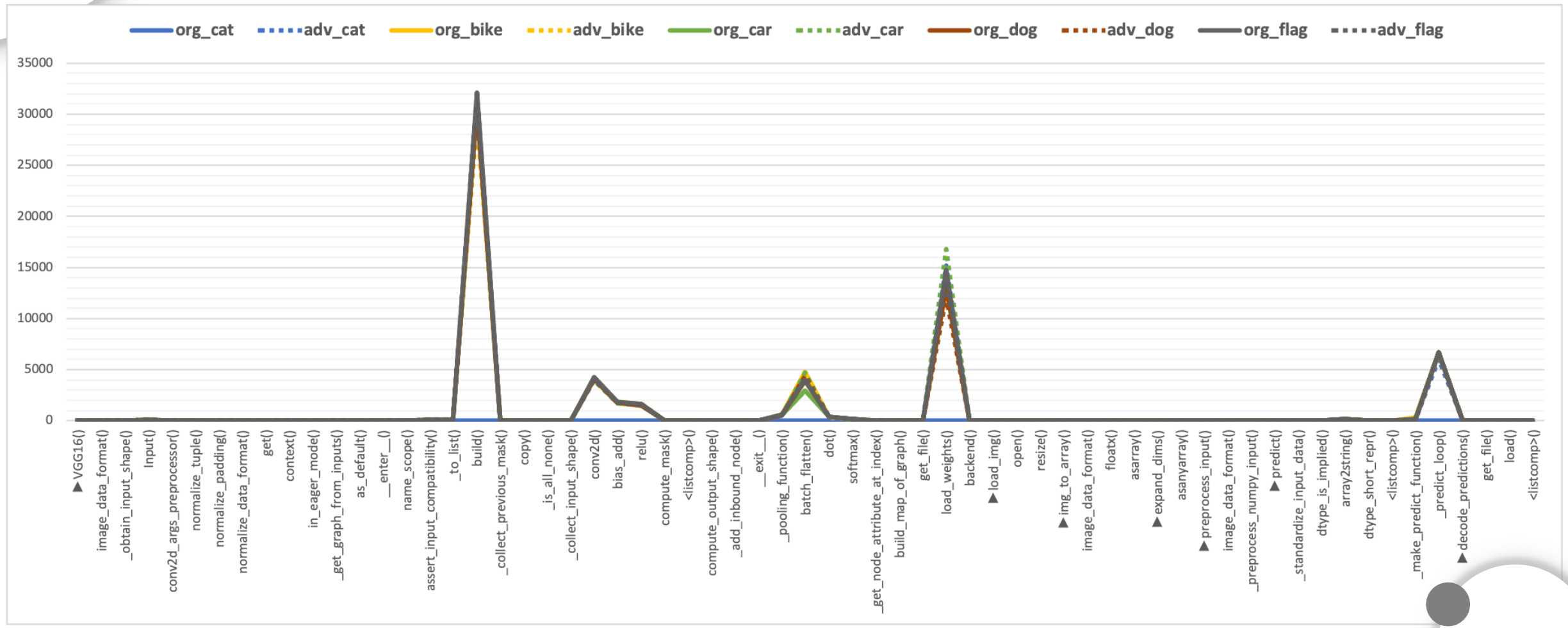
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 46
n(B) = 50
n(C) = 48
n(A') = 0
∴ No identifiable difference.

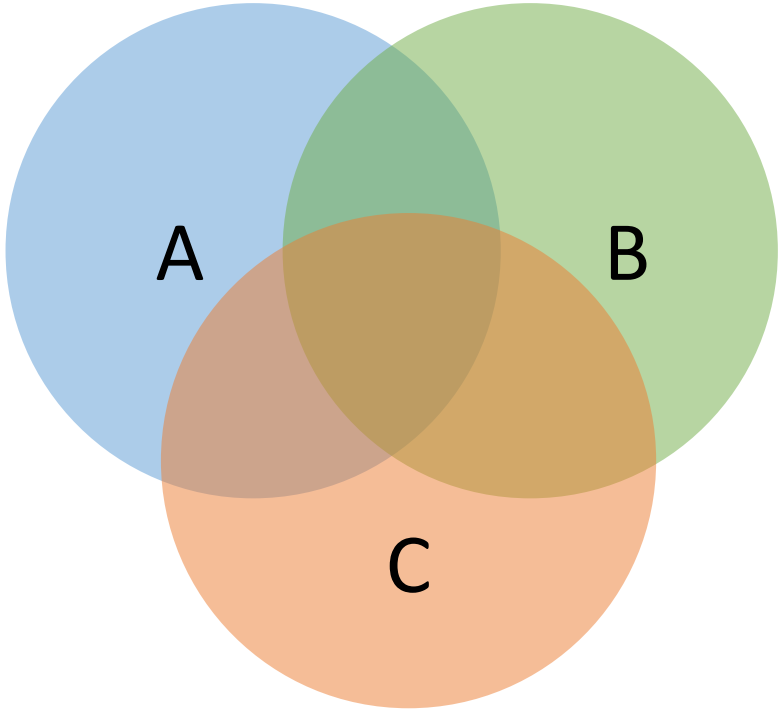
VGG16

# Difference on Sequential Call Count with Return Values



VGG16

# Difference on Sequential Call Count with Return Values




A = conjunction of { diff between original and attack }

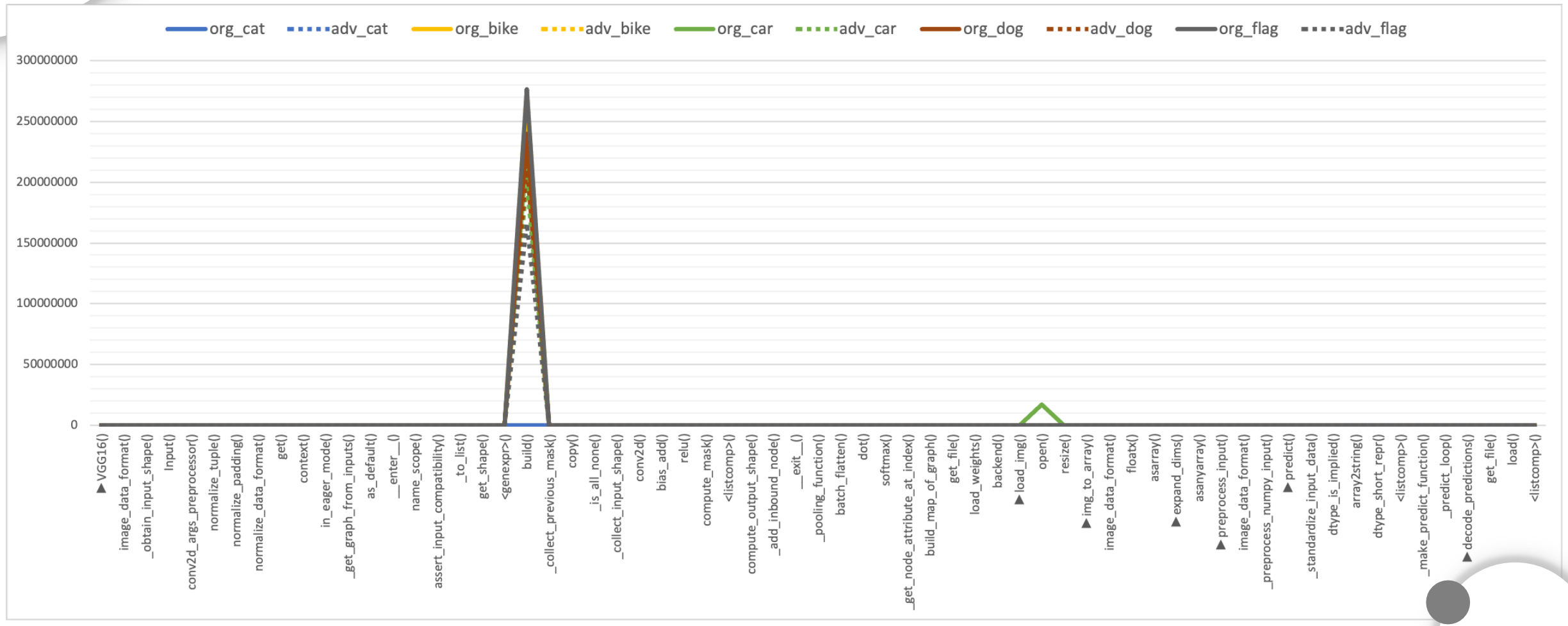
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 5
n(B) = 6
n(C) = 6
n(A') = 0
∴ No identifiable difference.

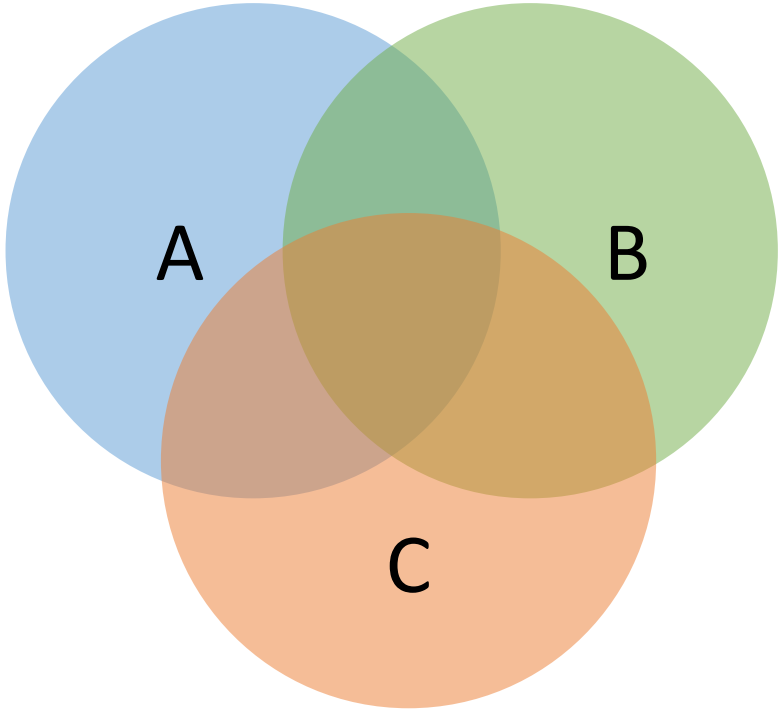
VGG16

# Difference on Sequential Integer Distance



VGG16

# Difference on Sequential Integer Distance




A = conjunction of { diff between original and attack }

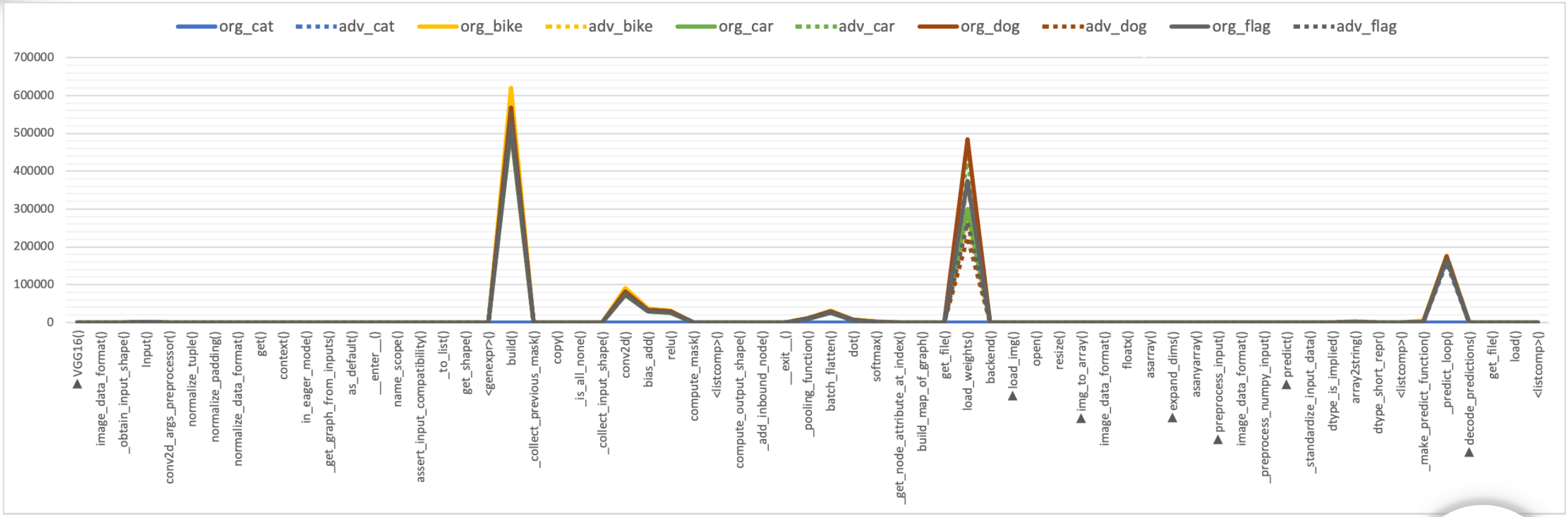
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 2
n(B) = 2
n(C) = 2
n(A') = 0
∴ No identifiable difference.

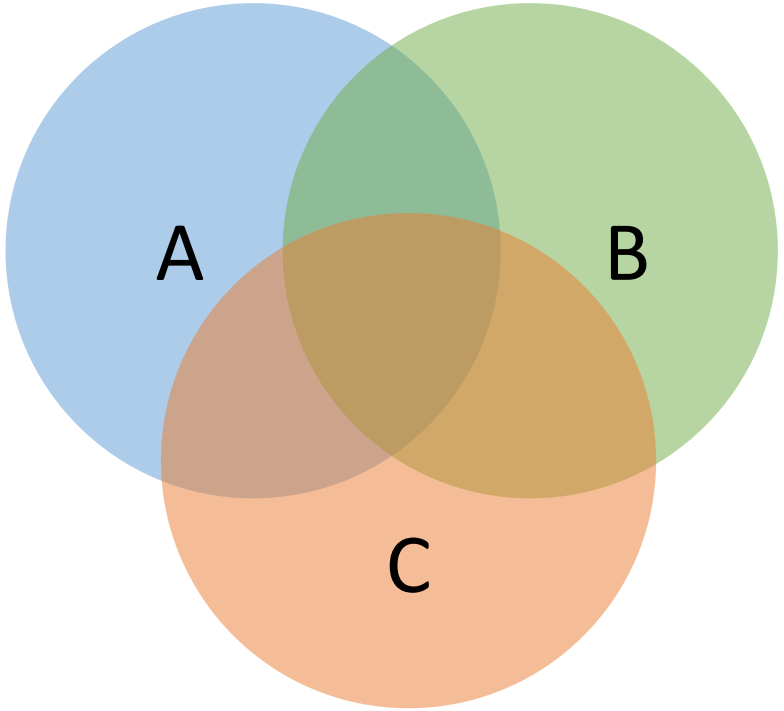
VGG16

# Difference on Sequential String Distance



VGG16

# Difference on Sequential String Distance




A = conjunction of { diff between original and attack }

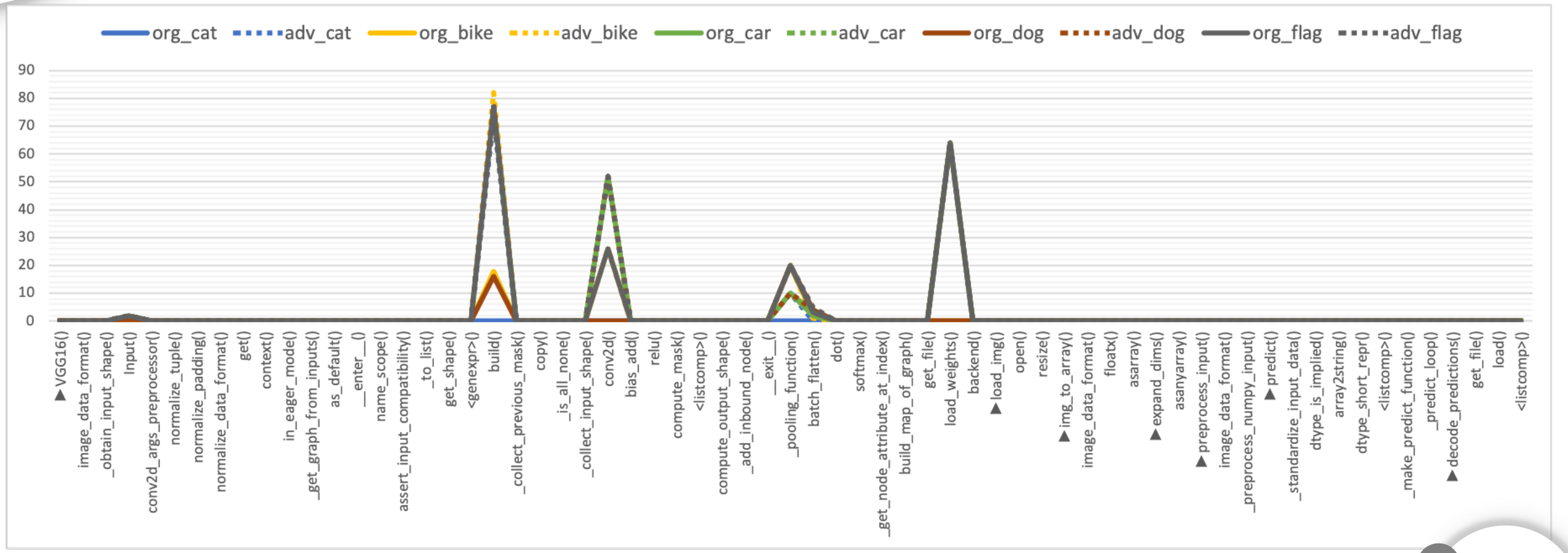
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 11
n(B) = 12
n(C) = 12
n(A') = 0
∴ No identifiable difference.

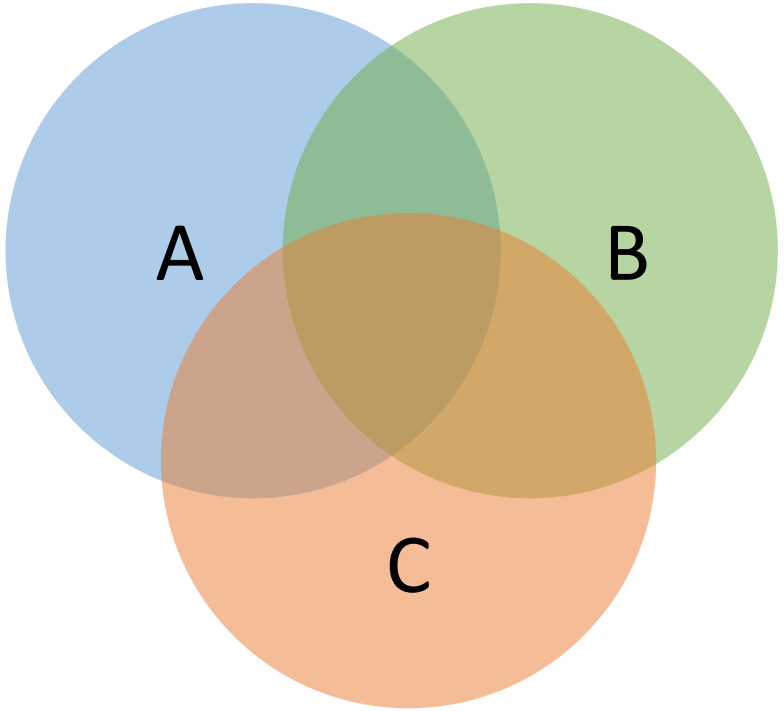
VGG16

# Difference on Sequential Boolean Distance



VGG16

# Difference on Sequential Boolean Distance




A = conjunction of { diff between original and attack }

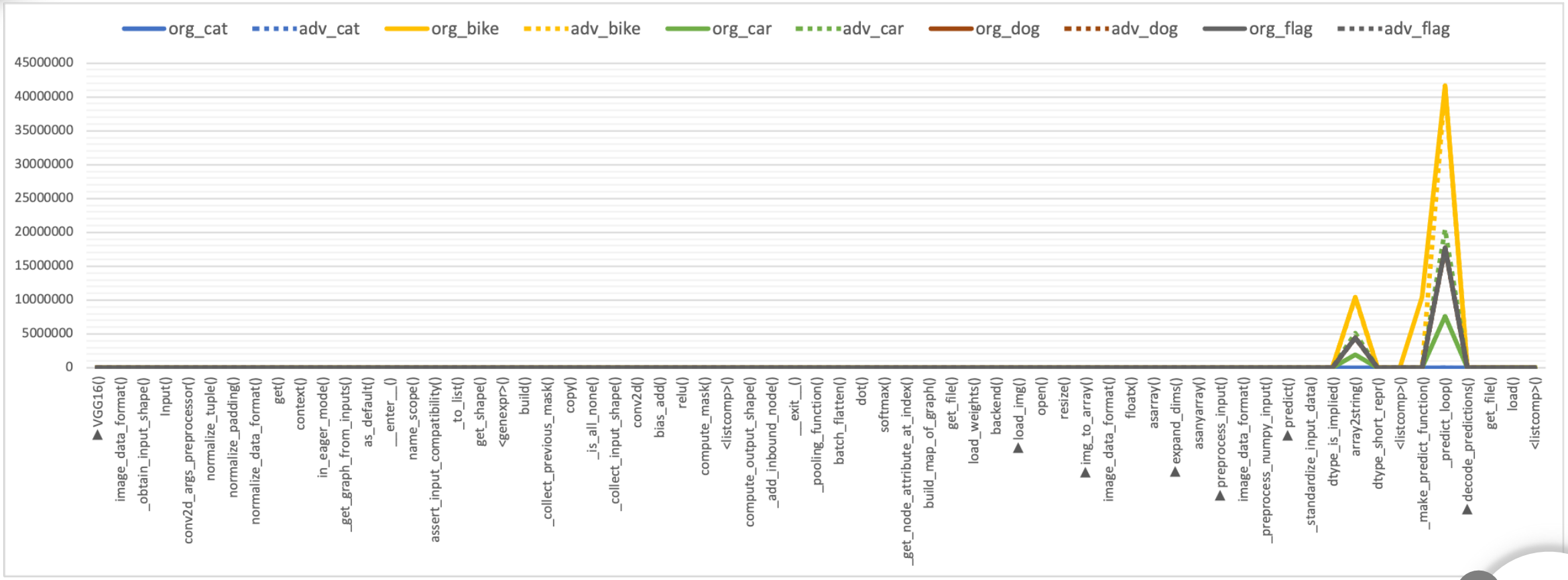
B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 5
n(B) = 3
n(C) = 4
n(A') = 0
∴ No identifiable difference.

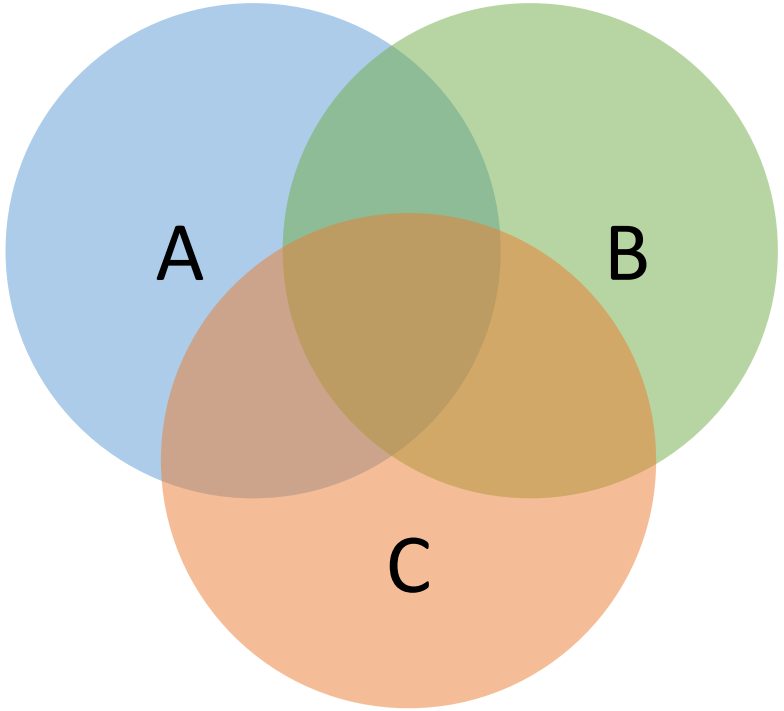
VGG16

# Difference on Sequential Matrix Distance



VGG16

# Difference on Sequential Matrix Distance




A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 1
n(B) = 1
n(C) = 1
n(A') = 0
∴ No identifiable difference.

# Results of Each Model

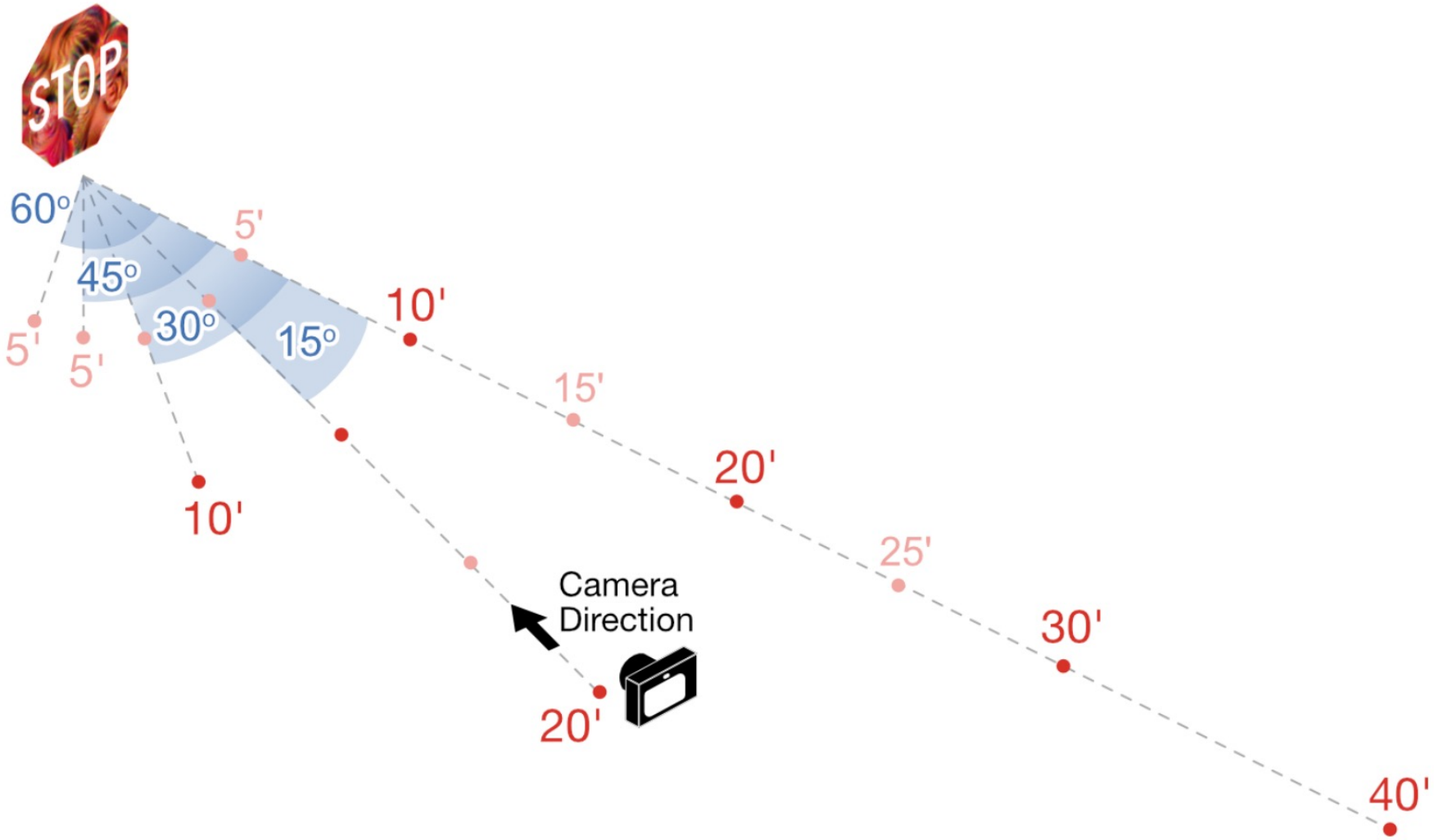
| | Call Count | Execution Time | Return Value |
|---|---|---|---|
| VGG16 | 0 | 0 | 0 |
| VGG19 | 0 | 0 | 0 |
| ResNet50 | 0 | 0 | 0 |
| InceptionV3 | 0 | 0 | 0 |
| Xception | 0 | 0 | 0 |

Unit: n(A')

# Part 04

## Object Detection Analysis

# Craft Adversarial Samples

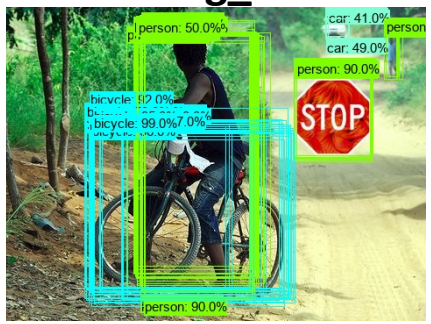


# Craft Adversarial Samples

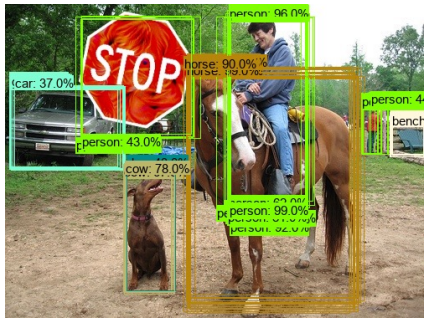
Org_1

Org_2

Org_3

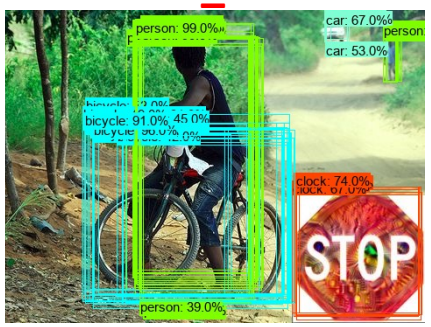



Adv_1-1

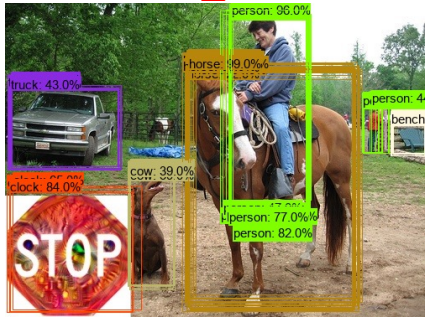



Adv_1-2


Adv_1-3



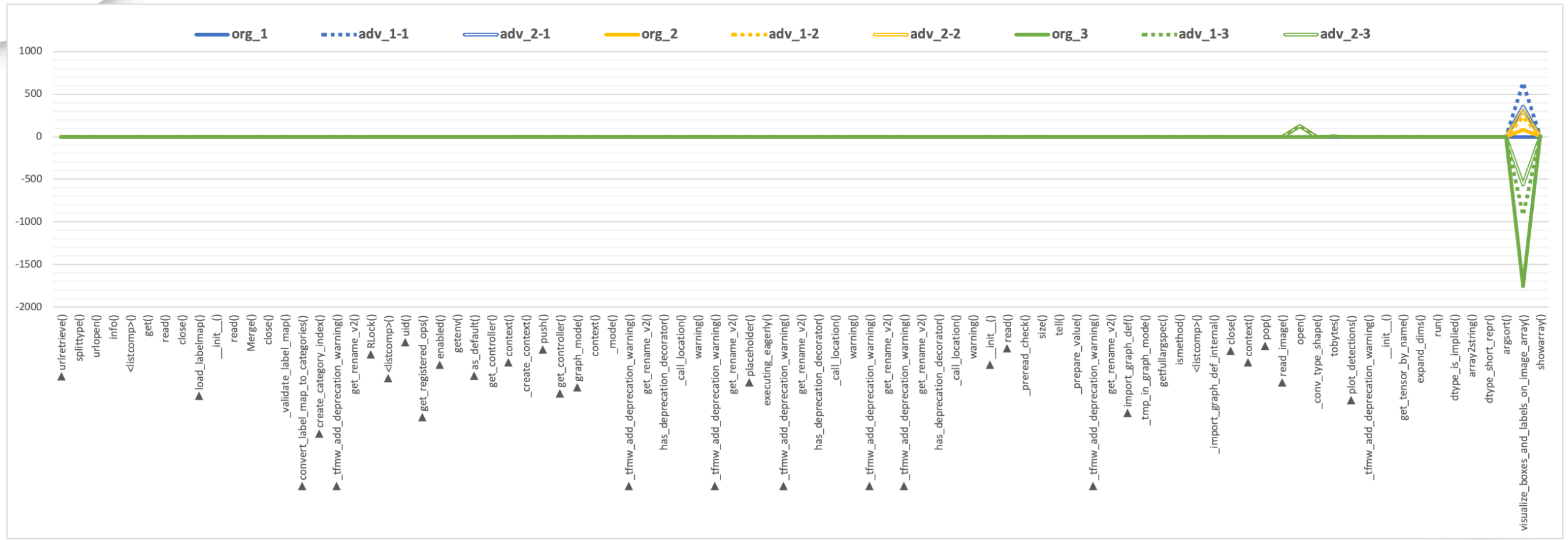

Adv_2-1



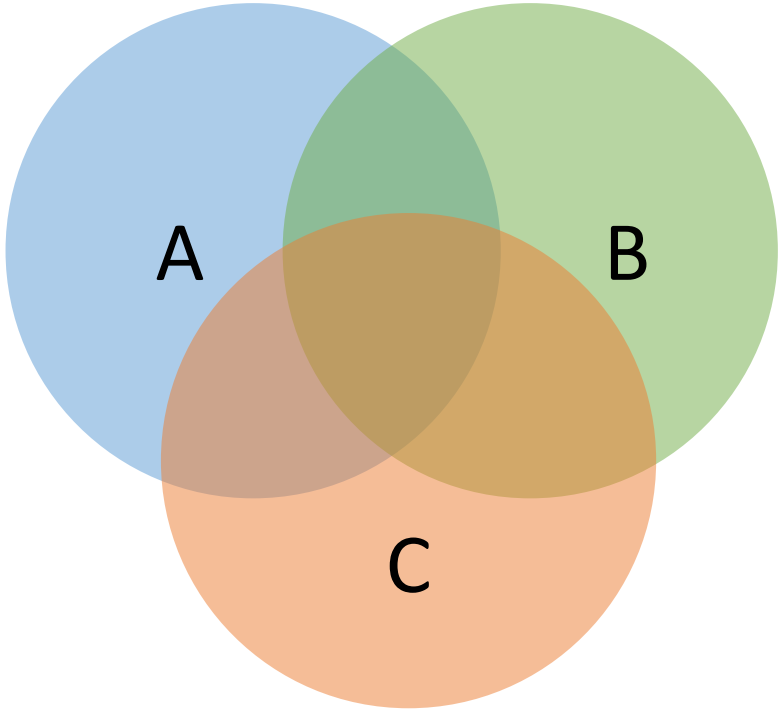

Adv_2-2


Adv_2-3

# Difference on Sequential Call Counts



# Difference on Sequential Call Counts



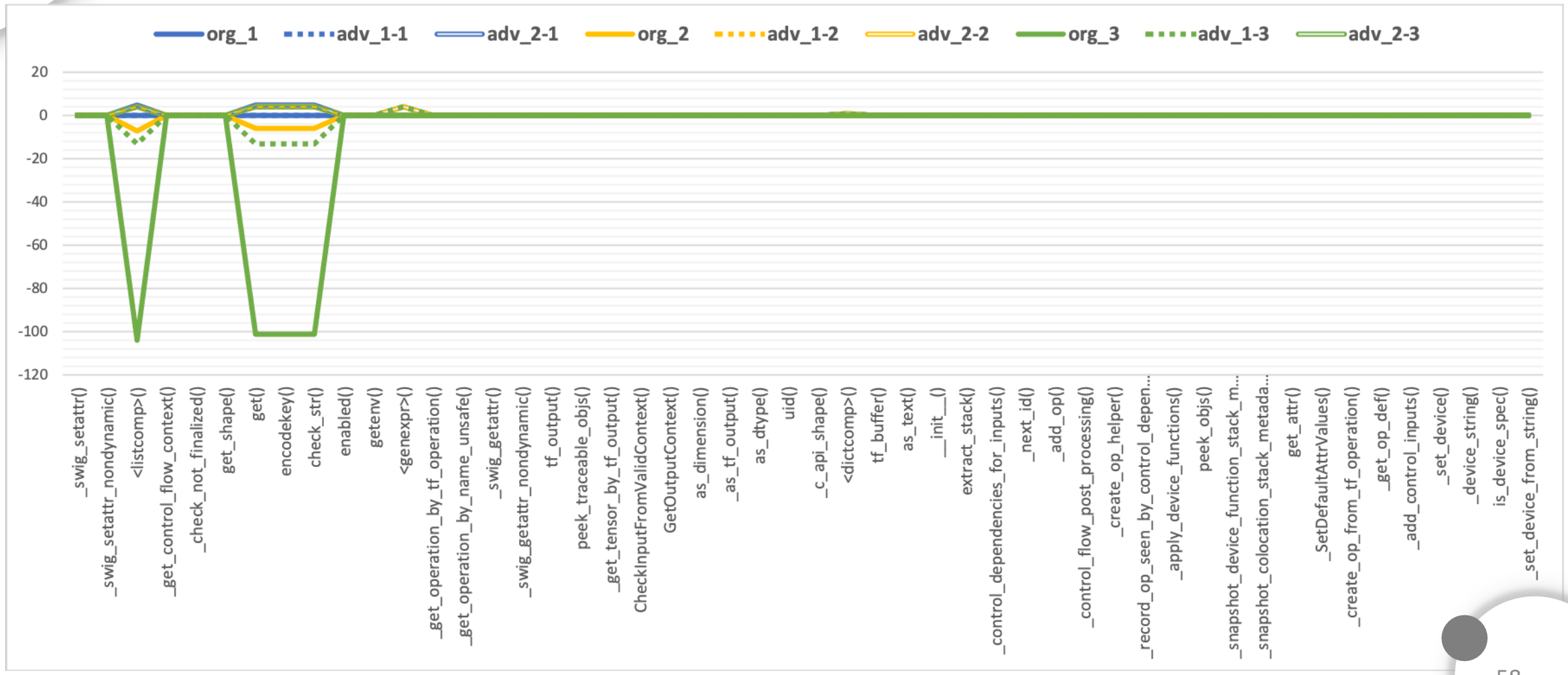

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

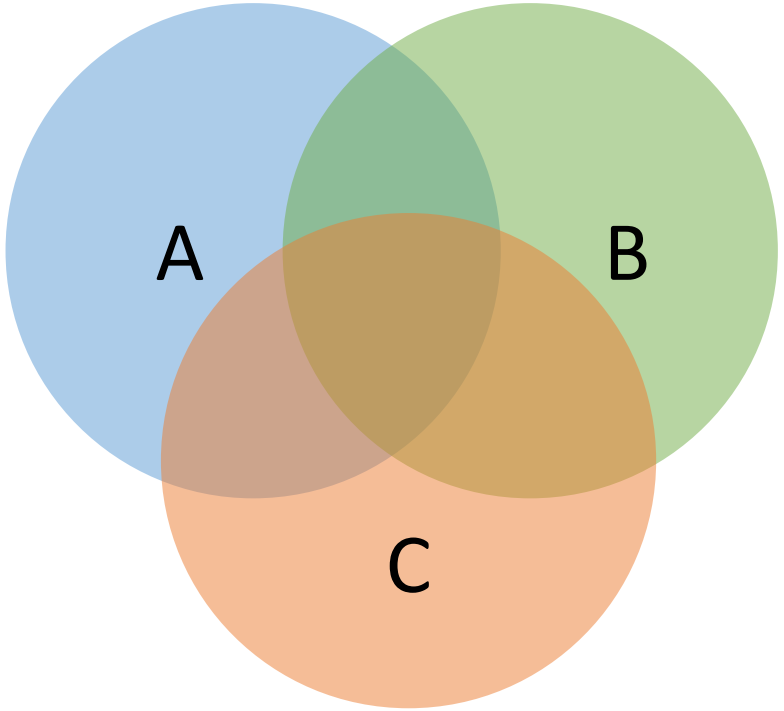
C = conjunction of { diff between attack and attack }

n(A) = 3
n(B) = 2
n(C) = 2
n(A') = 1
A' = { open() }

# Difference on Top50 Call Counts



# Difference on Top50 Call Counts



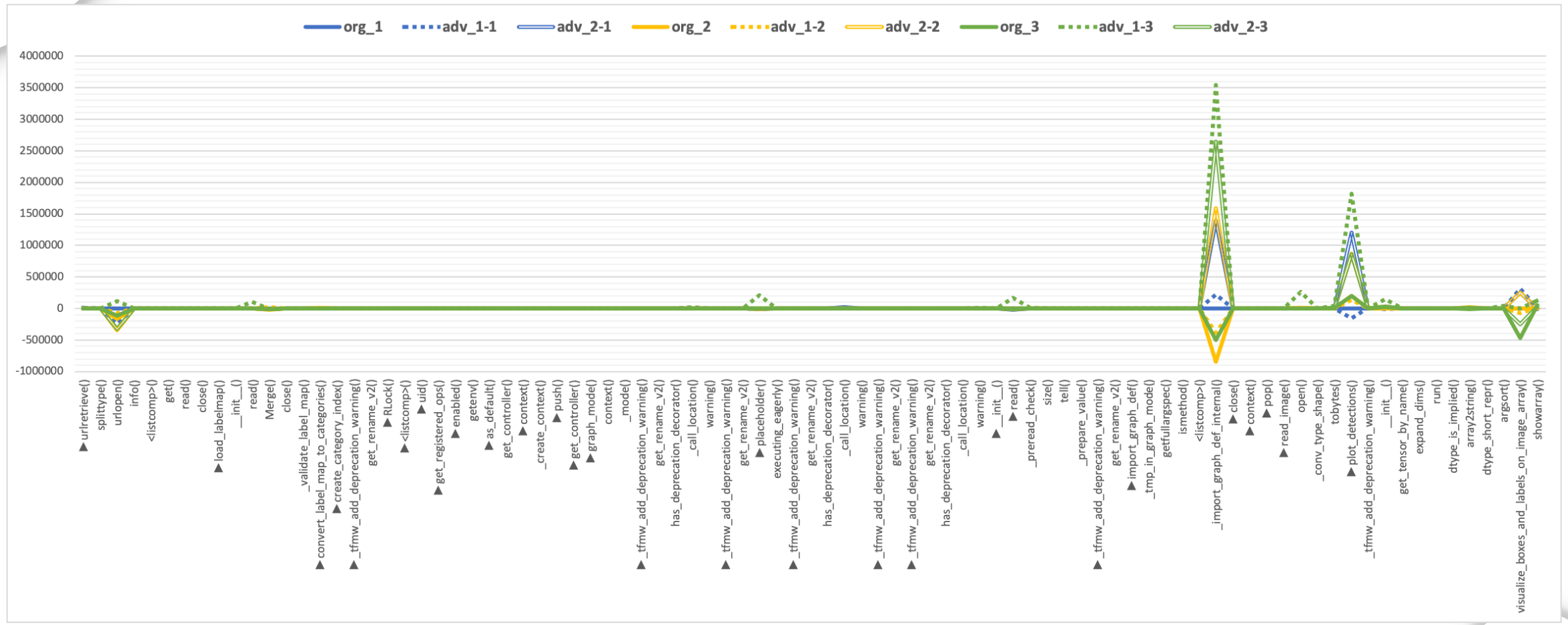

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

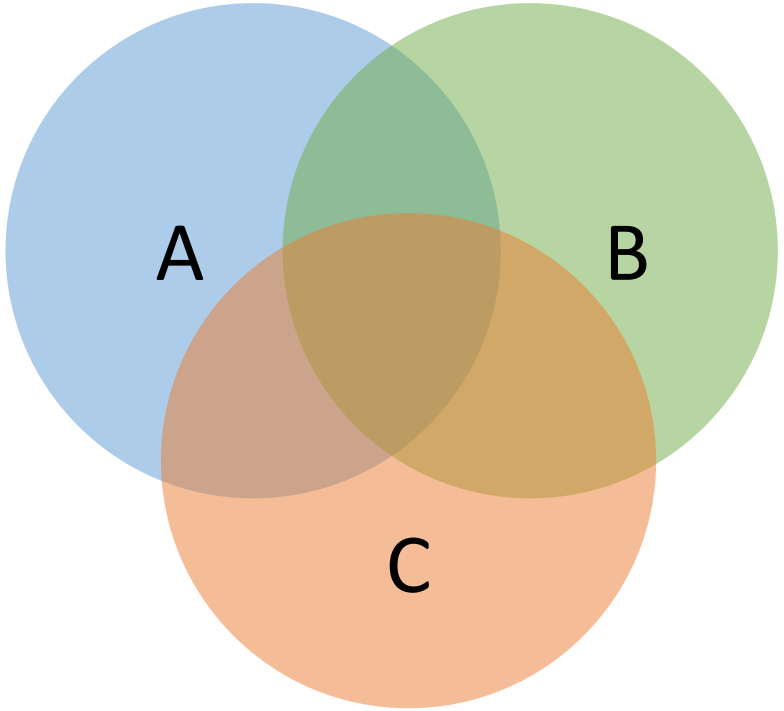
C = conjunction of { diff between attack and attack }

n(A) = 0
n(B) = 4
n(C) = 4
∴ No difference.

# Difference on Sequential Execution Time



# Difference on Sequential Execution Time



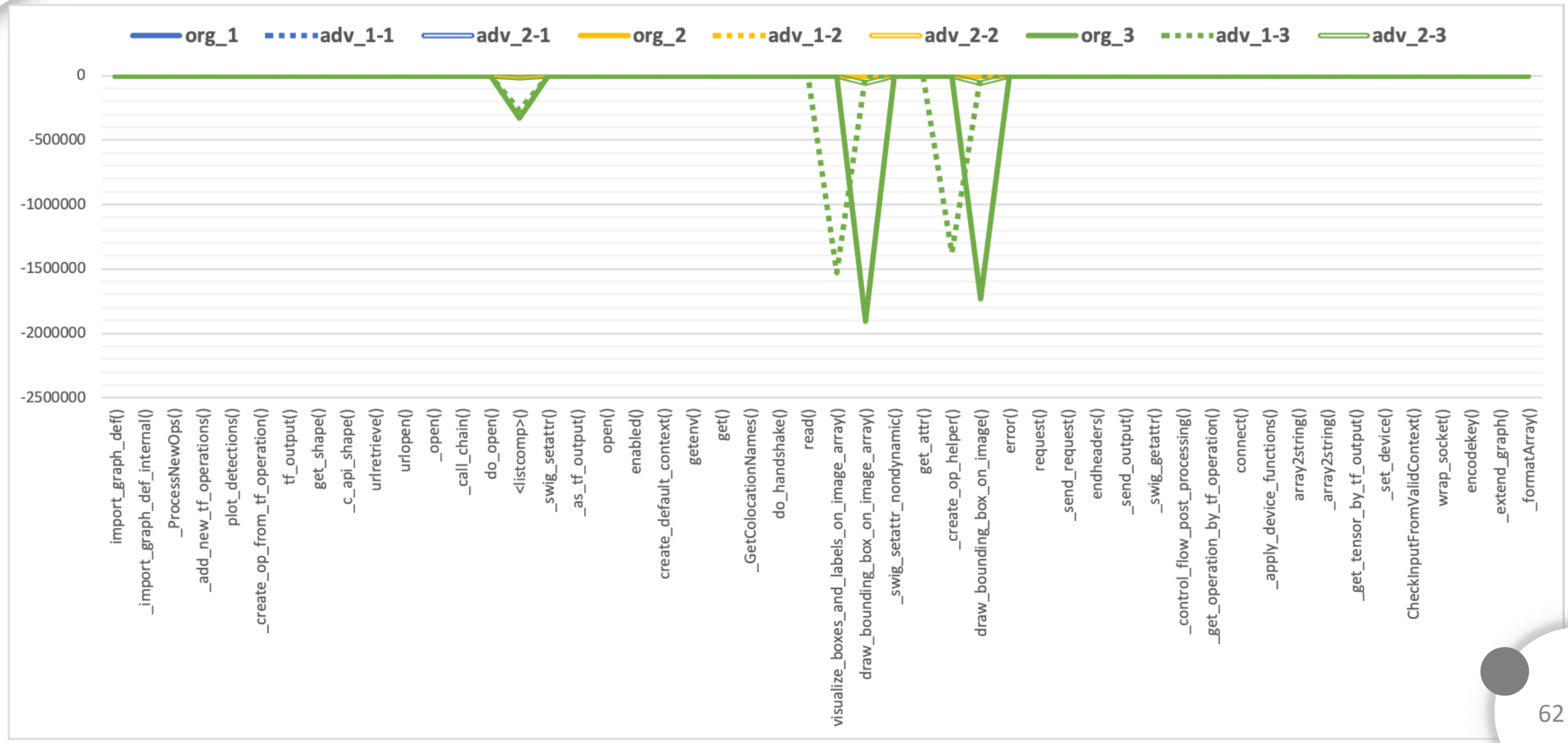

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

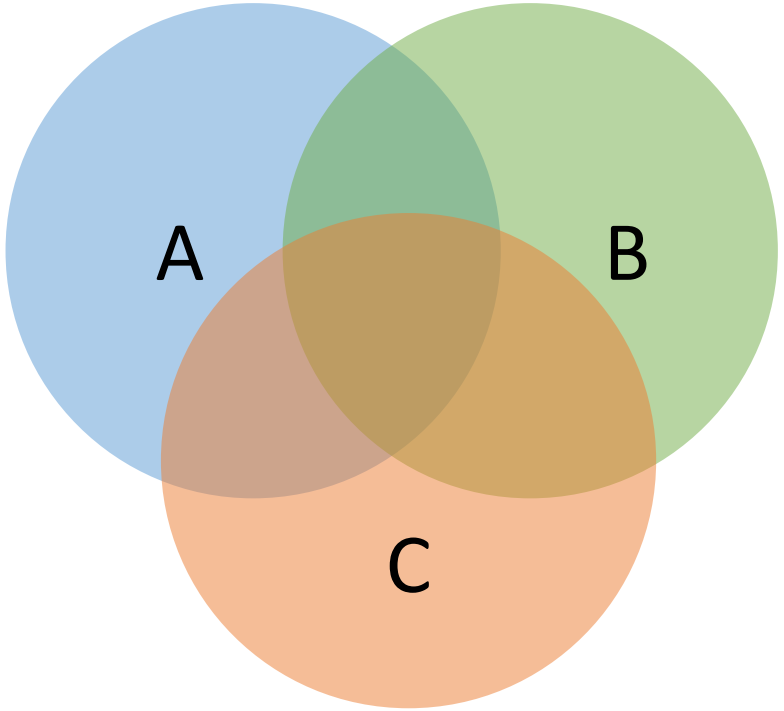
C = conjunction of { diff between attack and attack }

n(A) = 16
n(B) = 14
n(C) = 15
n(A') = 0
∴ No identifiable difference.

# Difference on Top50 Execution Time



# Difference on Top50 Execution Time



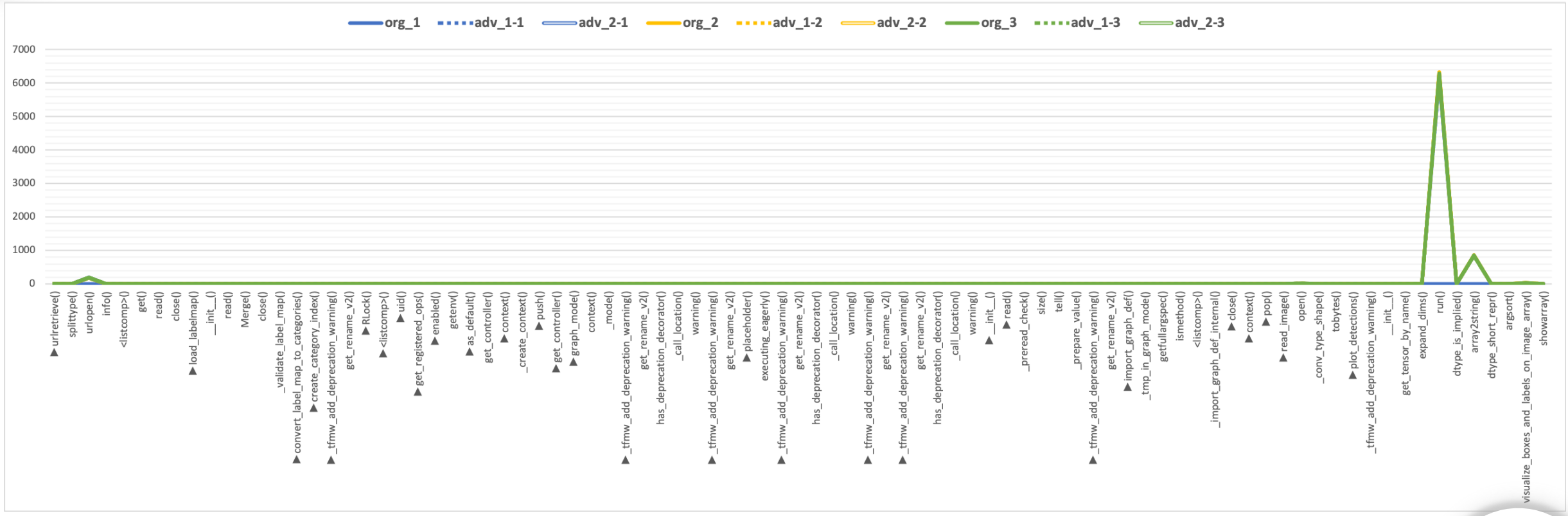

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

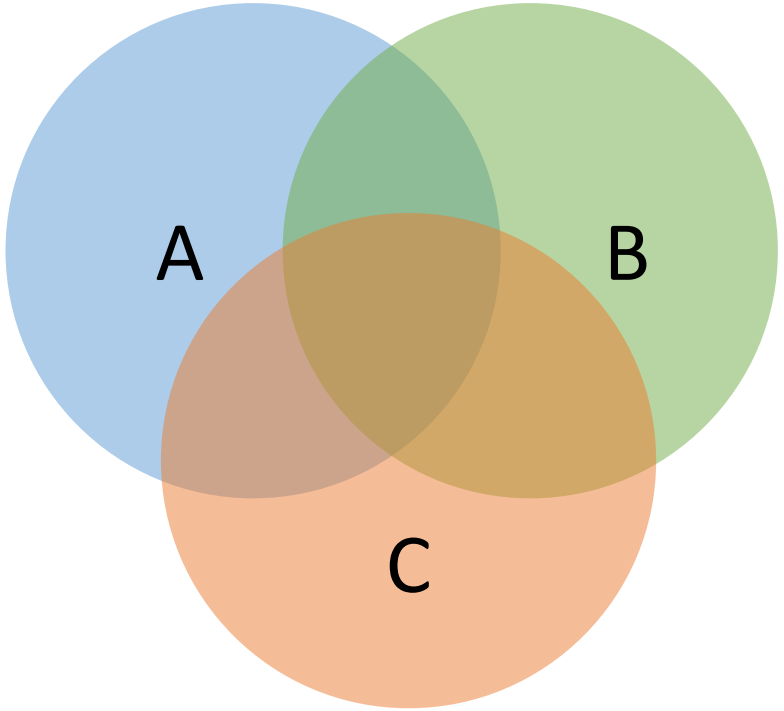
C = conjunction of { diff between attack and attack }

n(A) = 0
n(B) = 0
n(C) = 0
∴ No difference.

# Difference on Sequential Call Count with Return Values



# Difference on Sequential Call Count with Return Values



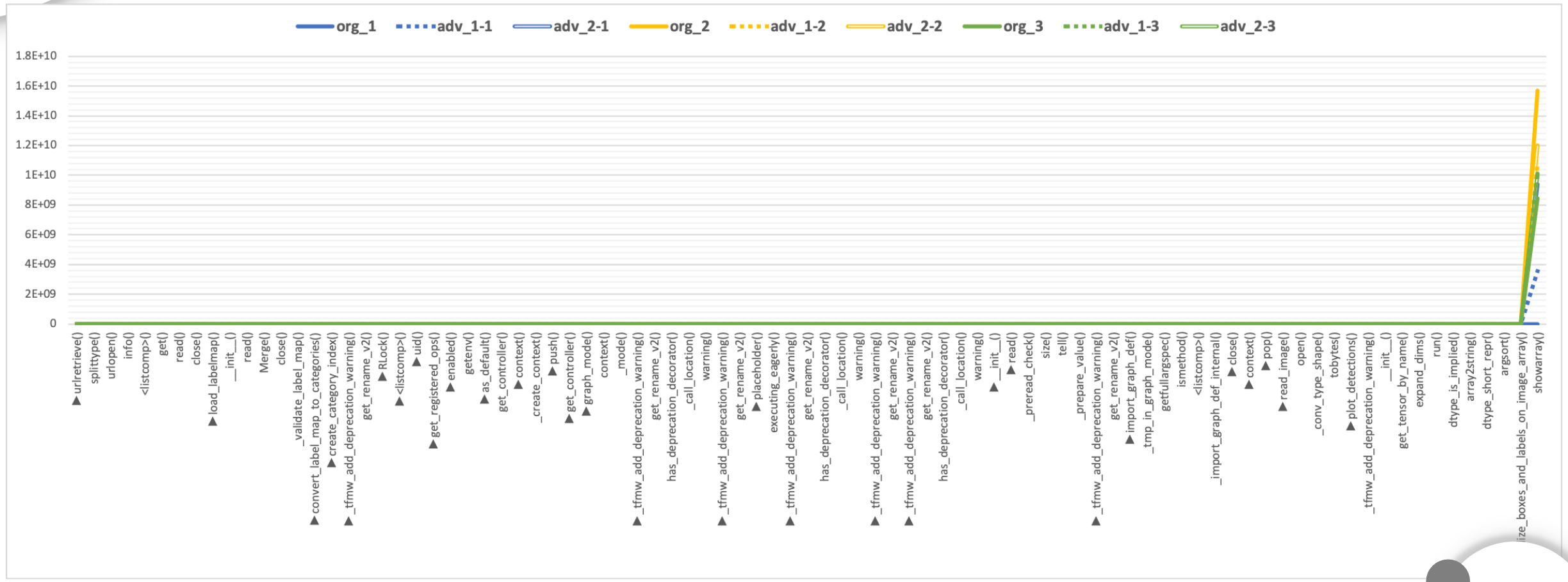

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

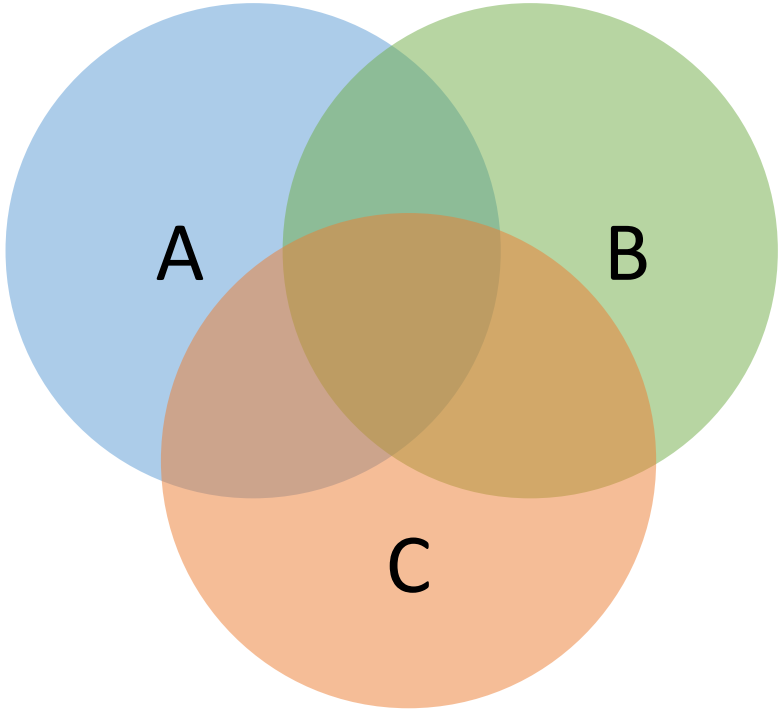
C = conjunction of { diff between attack and attack }

n(A) = 7
n(B) = 8
n(C) = 8
n(A') = 0
∴ No identifiable difference.

# Difference on Sequential Integer Distance



# Difference on Sequential Integer Distance



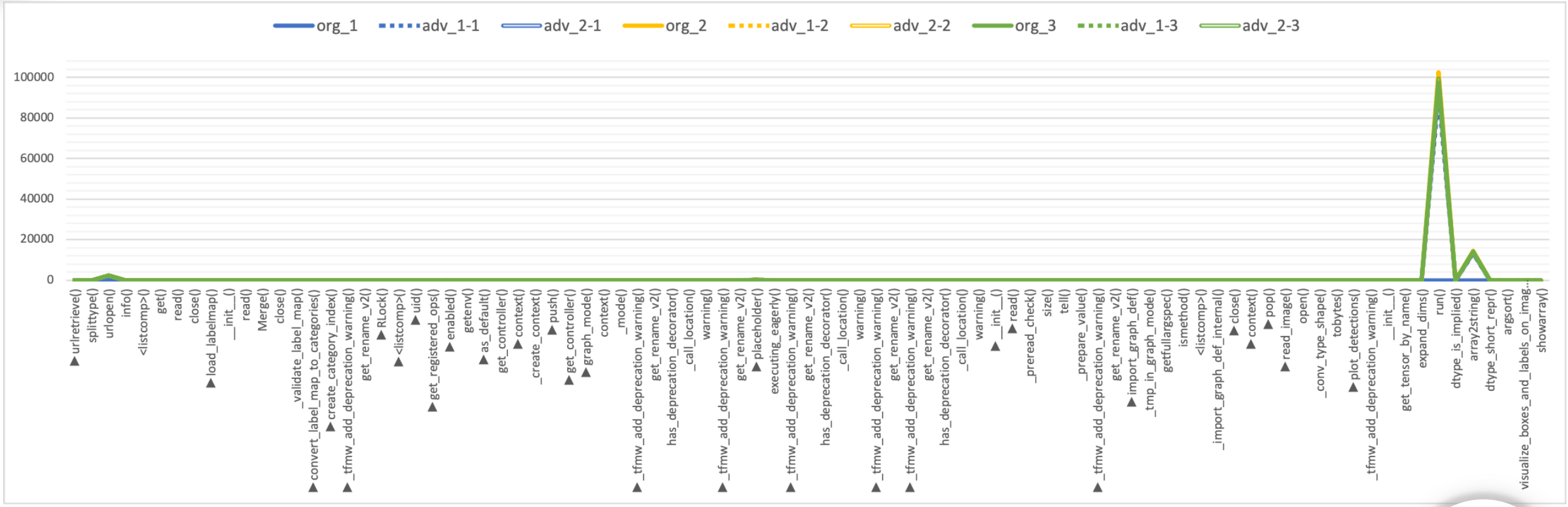

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

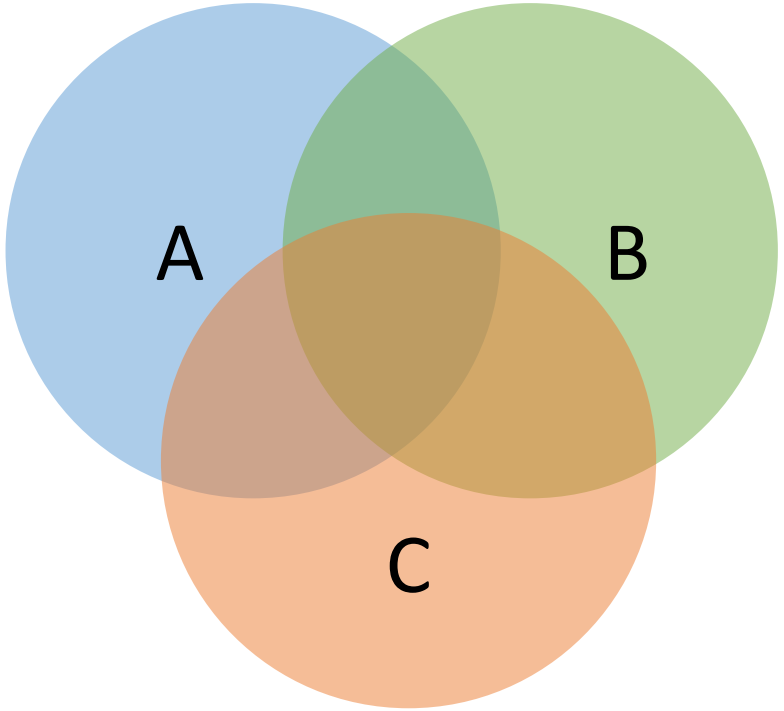
C = conjunction of { diff between attack and attack }

n(A) = 3
n(B) = 3
n(C) = 3
n(A') = 0
∴ No identifiable difference.

# Difference on Sequential String Distance



# Difference on Sequential String Distance



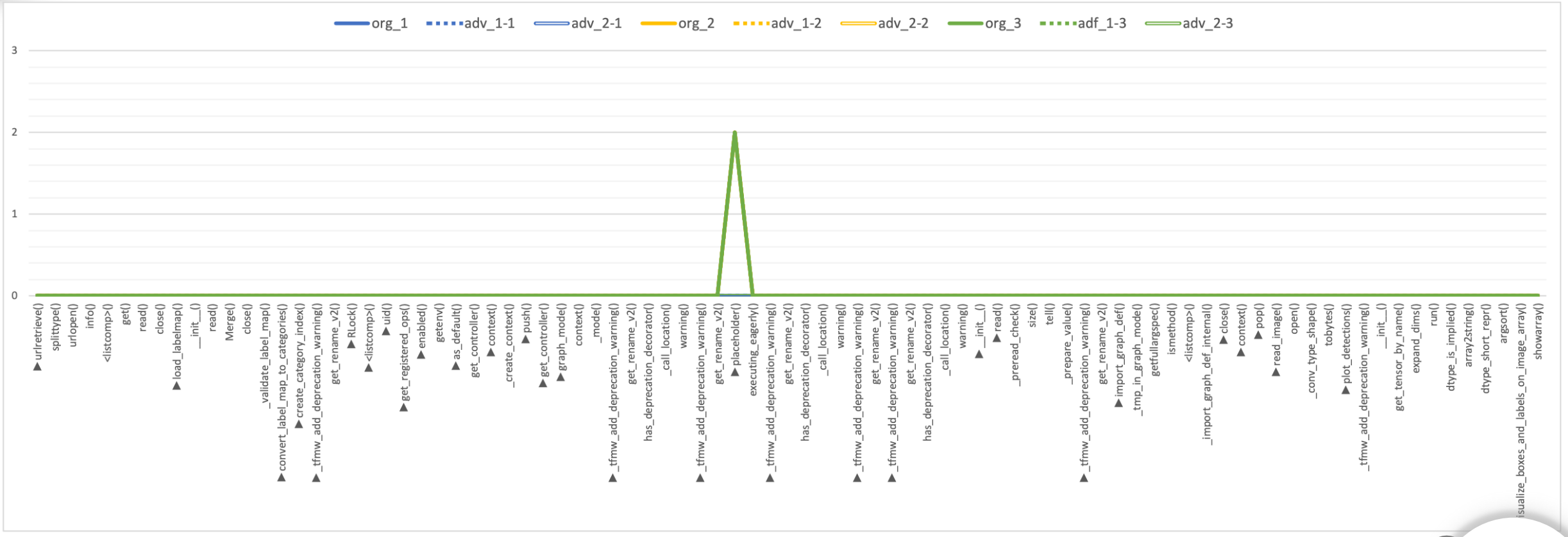

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

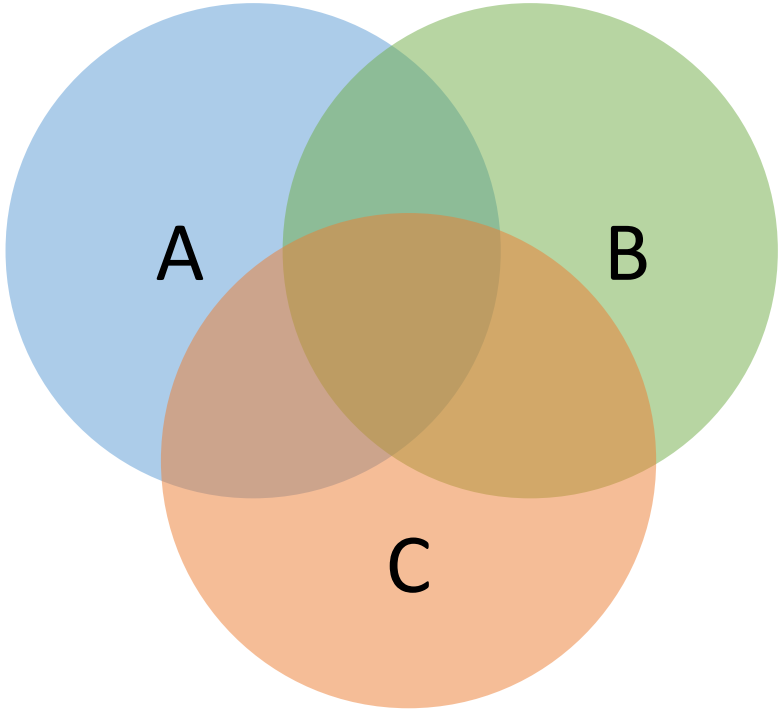
C = conjunction of { diff between attack and attack }

n(A) = 3
n(B) = 3
n(C) = 3
n(A') = 0
∴ No identifiable difference.

# Difference on Sequential Boolean Distance



# Difference on Sequential Boolean Distance



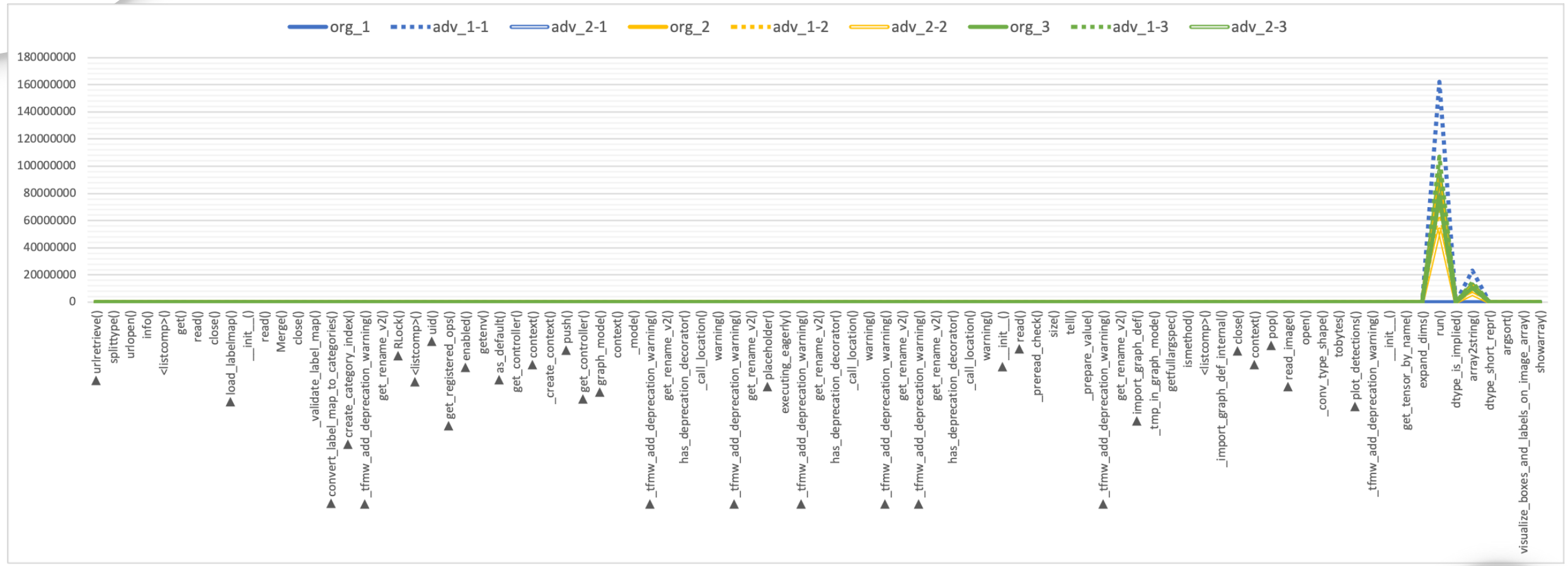

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

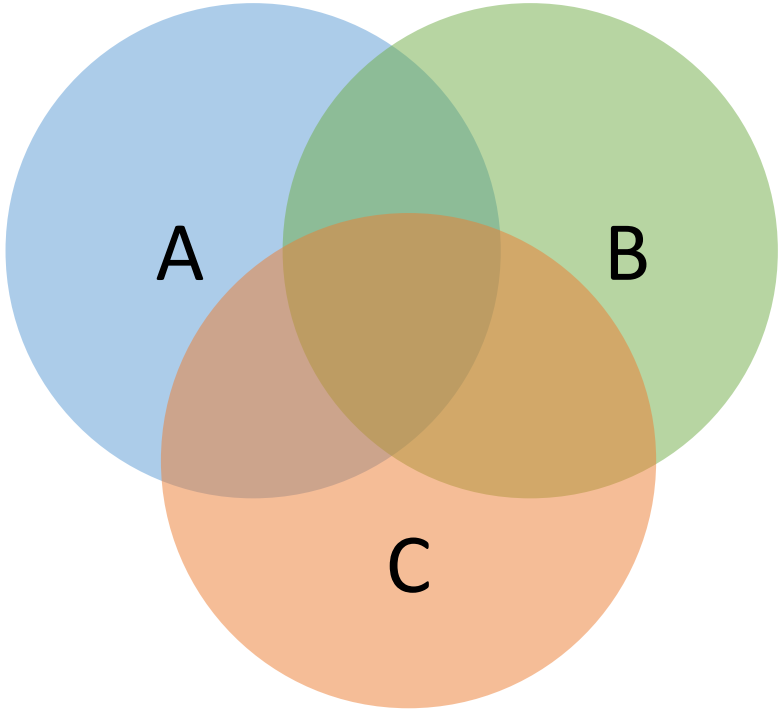
C = conjunction of { diff between attack and attack }

n(A) = 0
n(B) = 1
n(C) = 0
∴ No difference.

# Difference on Sequential Matrix Distance



# Difference on Sequential Matrix Distance




A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 2
n(B) = 2
n(C) = 1
n(A') = 0
∴ No identifiable difference.

# Summary of Results

| | Call Count | Execution Time | Return Value |
|---|---|---|---|
| Faster R-CNN | 1 | 0 | 0 |

Unit: n(A’)

- n(A’) = { open() }

# Derive Detection Rule

- Call count on different return values under the open() function

| Layer2 | Layer3 | Layer4 | org_1 | adv_1-1 | adv_2-1 | org_2 | adv_1-2 | adv_2-2 | org_3 | adv_1-3 | adv_2-3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| open() | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | isPath() | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | preinit() | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | _open_core() | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | _accept() | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | _dib_accept() | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | jpeg_factory() | 0 | 12 | 12 | 5 | 13 | 13 | 5 | 13 | 13 |
| | | _decompression_bomb_check() | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Derive Detection Rule

- **Call count on different calls under the open() function**

| | org_1 | adv_1-1 | adv_2-1 | org_2 | adv_1-2 | adv_2-2 | org_3 | adv_1-3 | adv_2-3 |
|---|---|---|---|---|---|---|---|---|---|
| adv_2-3 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| adv_1-3 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| org_3 | 0 | 21 | 21 | 0 | 21 | 21 | 0 | 21 | 21 |
| adv_2-2 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| adv_1-2 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| org_2 | 0 | 21 | 21 | 0 | 21 | 21 | 0 | 21 | 21 |
| adv_2-1 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| adv_1-1 | 21 | 0 | 0 | 21 | 0 | 0 | 21 | 0 | 0 |
| org_1 | 0 | 21 | 21 | 0 | 21 | 21 | 0 | 21 | 21 |

| Number | Org | Adv |
|---|---|---|
| 1 | DQT() | APP() |
| 2 | i8() | getexif() |
| 3 | - | load(),child[...] |
| 4 | DQT() | APP() |
| 5 | i8() | i16be() |
| 6 | - | i8() |
| 7 | - | i32be() |
| 8 | - | i16be() |
| 9 | - | i8() |
| 10 | - | i32be() |
| 11 | Skip() | DQT() |
| 12 | - | i8() |
| 13 | - | i8() |
| 14 | - | i16be() |
| 15 | - | DQT(),child[...] |
| 16 | - | i8() |
| 17 | - | i16be() |
| 18 | - | Skip(),child[...] |
| 19 | - | i8() |
| 20 | - | i16be() |
| 21 | - | Skip(),child[...] |

# Rule Derivation

| Layer2 | Layer3 | Layer4 | Layer5 | Layer6 |
|---|---|---|---|---|
| open() | | | | |
| | isPath() | | | |
| | preinit() | | | |
| | _open_core() | | | |
| | | _accept() | | |
| | | _dib_accept() | | |
| | | jpeg_factory() | | |
| | | | _init_() | |
| | | | isPath() | |
| | | | _open() | |
| | | | | _accept() |
| | | | | i8() |
| | | | | i16be() |
| | | | | ... |

| Number | Org | Adv |
|---|---|---|
| 1 | DQT() | APP() |
| 2 | i8() | getexif() |
| 3 | - | load(),child[...] |
| 4 | DQT() | APP() |
| 5 | i8() | i16be() |
| 6 | - | i8() |
| 7 | - | i32be() |
| 8 | - | i16be() |
| 9 | - | i8() |
| 10 | - | i32be() |
| 11 | Skip() | DQT() |
| 12 | - | i8() |
| 13 | - | i8() |
| 14 | - | i16be() |
| 15 | - | DQT(),child[...] |
| 16 | - | i8() |
| 17 | - | i16be() |
| 18 | - | Skip(),child[...] |
| 19 | - | i8() |
| 20 | - | i16be() |
| 21 | - | Skip(),child[...] |

# Rule Derivation

1. If the name of the seventh function under _open() is DQT(), it is a general picture, if the name is APP(), it is an adversarial picture.
2. If the number of child in _open() is 37, it is an adversarial picture, otherwise it is a general picture.
3. If the name of the third function under the seventh function under _open() is getexif(), it will be represented as an adversarial picture. Otherwise it is a general picture.

| Number | Org | Adv |
|---|---|---|
| 1 | DQT() | APP() |
| 2 | i8() | getexif() |
| 3 | - | load(),child[...] |
| 4 | DQT() | APP() |
| 5 | i8() | i16be() |
| 6 | - | i8() |
| 7 | - | i32be() |
| 8 | - | i16be() |
| 9 | - | i8() |
| 10 | - | i32be() |
| 11 | Skip() | DQT() |
| 12 | - | i8() |
| 13 | - | i8() |
| 14 | - | i16be() |
| 15 | - | DQT(),child[...] |
| 16 | - | i8() |
| 17 | - | i16be() |
| 18 | - | Skip(),child[...] |
| 19 | - | i8() |
| 20 | - | i16be() |
| 21 | - | Skip(),child[...] |

# Testing

In 100 negative pictures:

- 40 adversarial stop signs
- 40 general stopsigns
- 20 arbitrary objects

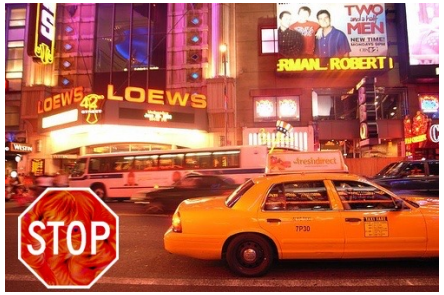
| | | Prediction | |
|---|---|---|---|
| | n=1000 | positive | negative |
| Label | positive | 900 | 0 |
| | negative | 0 | 100 |

# Testing

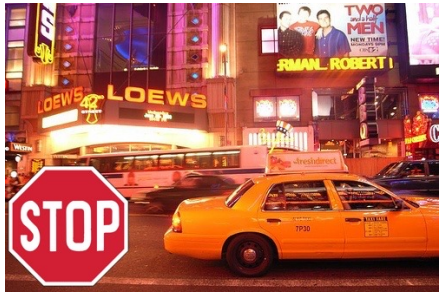

1_1


1_2

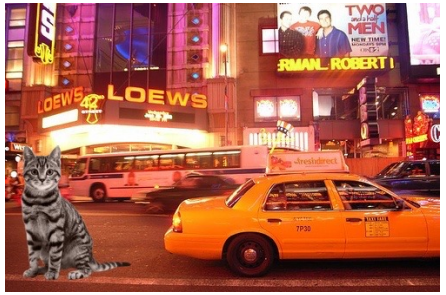
1_3




2_1




2_2



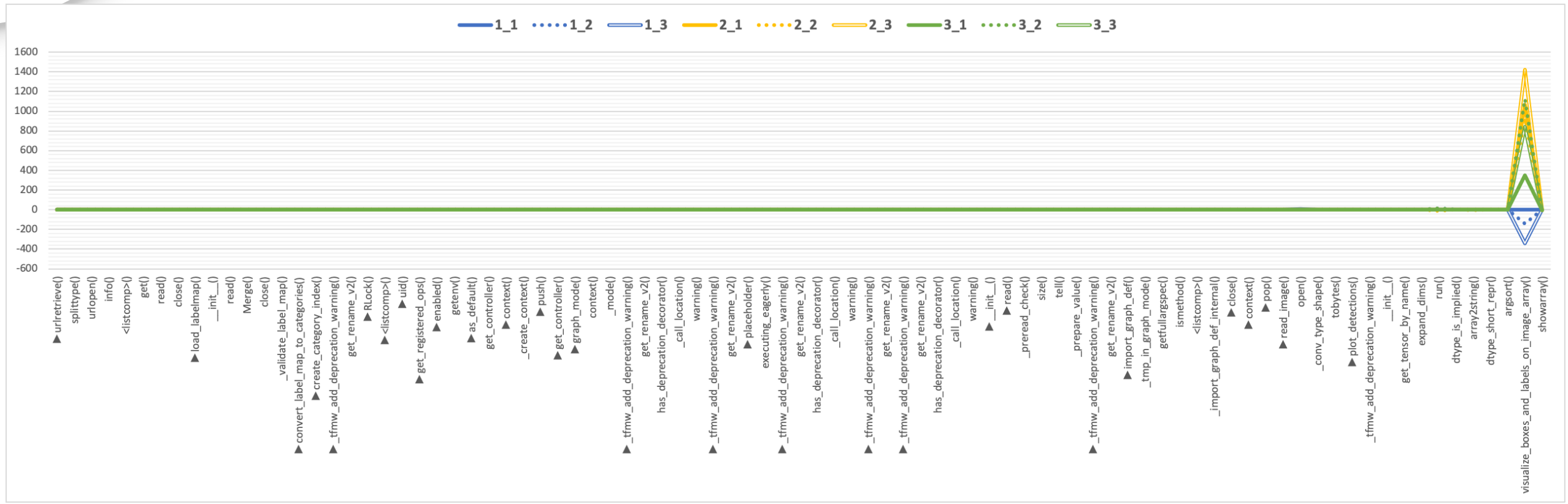

2_3


3_1

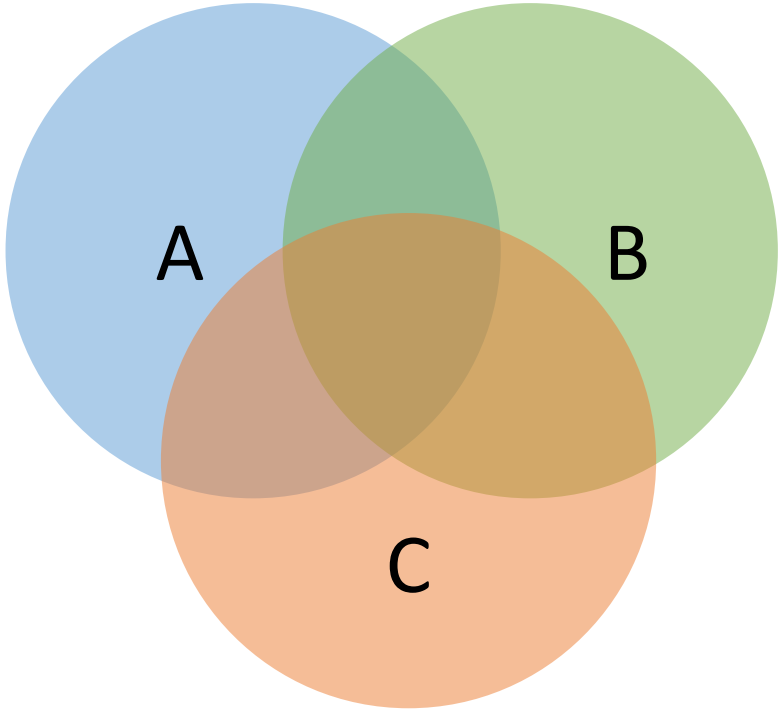

3_2

3_3

# Difference on Sequential Call Counts



# Difference on Sequential Call Counts



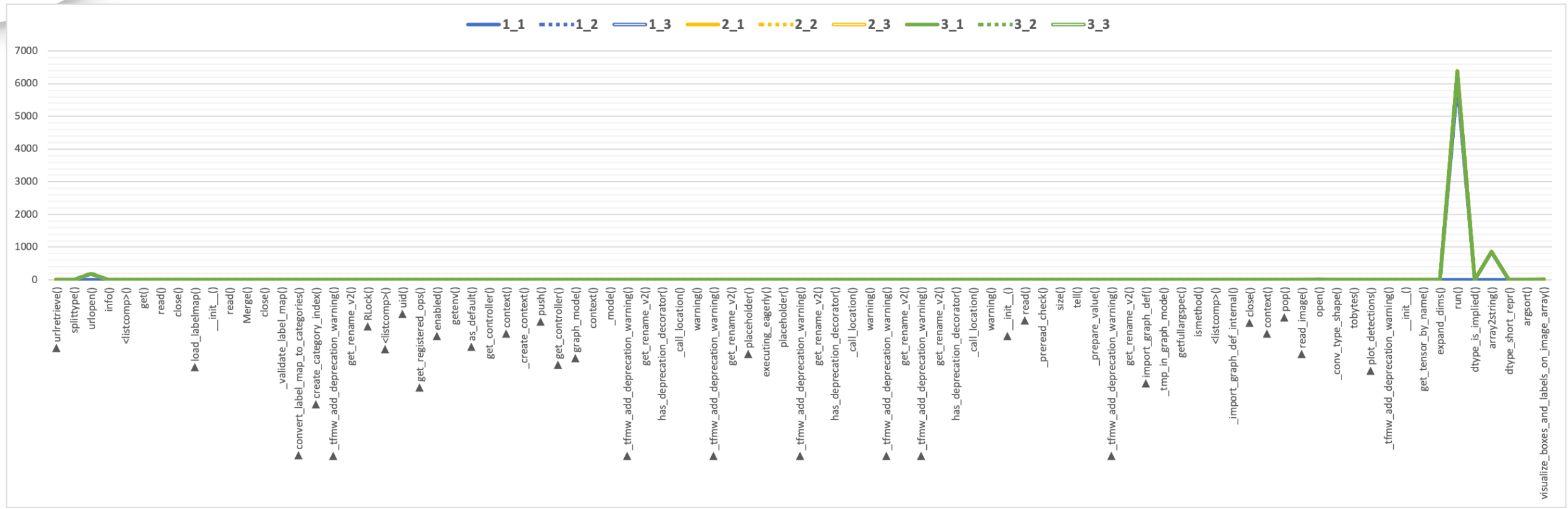

A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

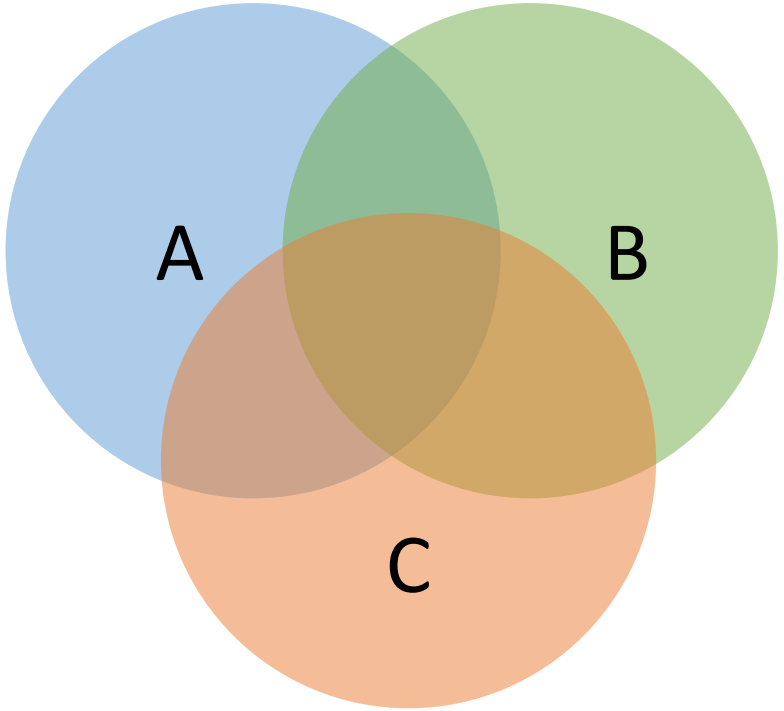
C = conjunction of { diff between attack and attack }

n(A) = 1
n(B) = 1
n(C) = 2
n(A') = 0
∴ No identifiable difference.

# Difference on Sequential Call Count with Return Values



# Difference on Sequential Call Count with Return Values




A = conjunction of { diff between original and attack }

B = conjunction of { diff between original and original }

C = conjunction of { diff between attack and attack }

n(A) = 5
n(B) = 4
n(C) = 8
n(A') = 0
∴ No identifiable difference.

# Part 05

## Conclusion

# Conclusion

- We proposed a systematic approach for differential analysis on detecting adversarial examples from Python program execution.
- We evaluated our approach against Keras and object detection applications with adversarial patches.
- We are able to derive an effective rule to distinguish original pictures and pictures with inserted objects.
- We report no identifiable differences from calls, times, and return values to distinguish adversarial patches and objects that twist the image recognition results.

# Thanks for Listening!