

# Practical evading attacks on commercial AI image recognition services



**Kang Li**

Department of Computer Science  
University of Georgia

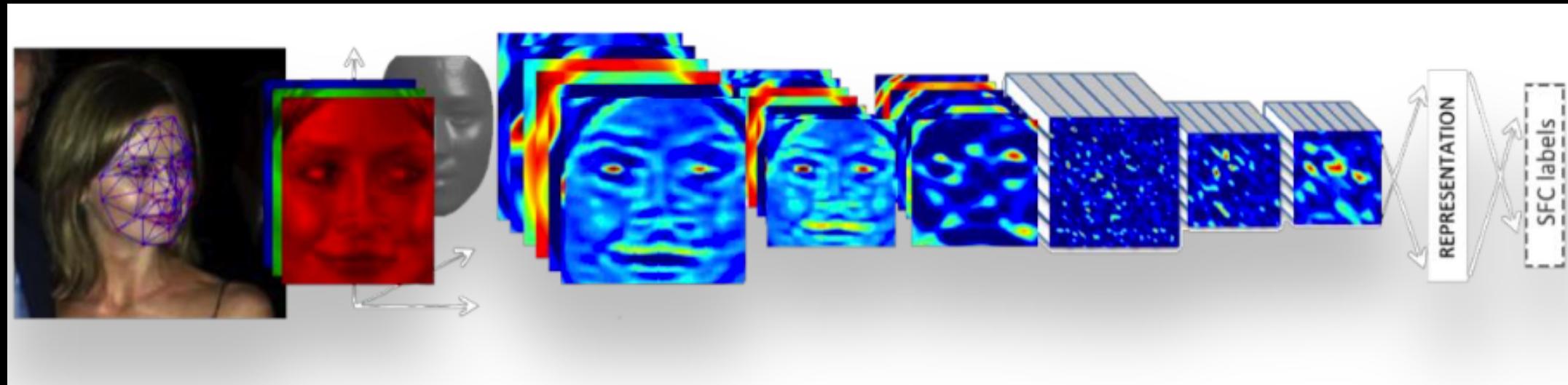
**Collaborators:** Yufei Chen, Qixue Xiao, Deyue Zhang

# About Me

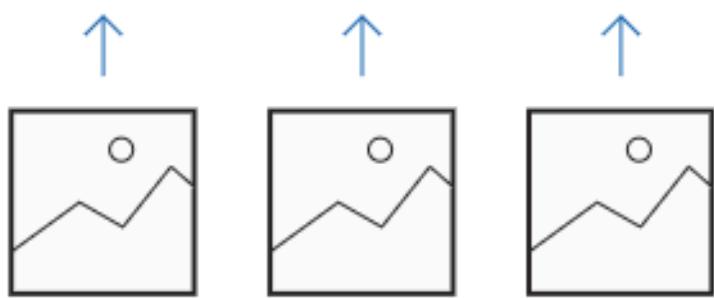
- Professor at the University of Georgia
- Director of UGA Institute for Cybersecurity and Privacy (ICSP)
- Founding mentor of *Blue-Lotus* CTF Team and *xCTF* League
- Founder of the *Disekt*, *SecDawgs* CTF Teams
- Finalist of 2016 DARPA Cyber Grand Challenge (CGC)



# Deep Learning and Advances in AI Applications

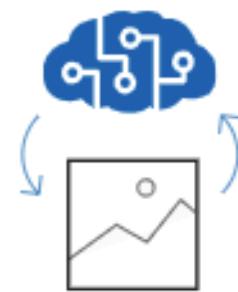


[https://www.cs.toronto.edu/~ranzato/publications/taigman\\_cvpr14.pdf](https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf)



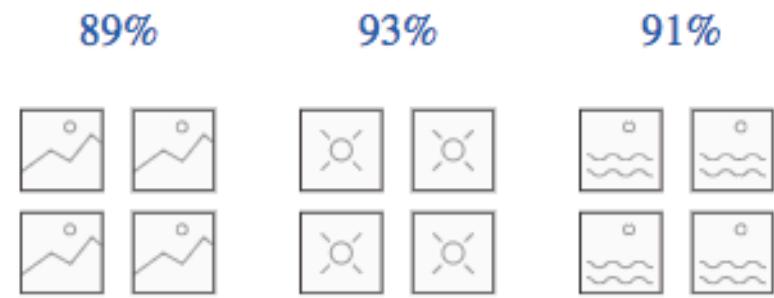
## Upload Images

Bring your own labeled images, or use Custom Vision to quickly add tags to any unlabeled images.



## Train

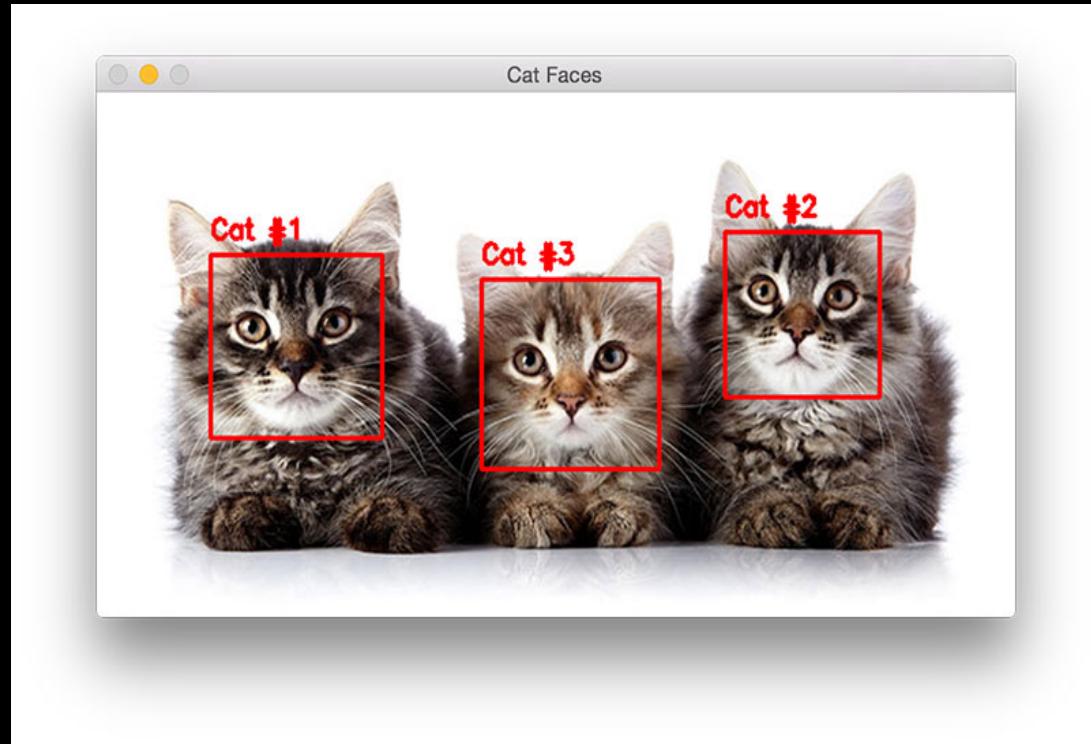
Use your labeled images to teach Custom Vision the concepts you care about.



## Evaluate

Use simple REST API calls to quickly tag images with your new custom computer vision model.

# AI & Image Recognition System



# Image Recognition As a Service

A screenshot of a web browser displaying the Google Cloud Vision API landing page. The URL bar shows a secure connection. The page header includes links for Products, Solutions, Pricing, Security, and Documentation, along with a search bar and sign-in options. A prominent section titled "AI & Machine Learning Products" features a "Try the API" button and a dashed box containing an input field with the placeholder text "Drag image file here or Browse from your computer". An orange rectangle highlights this input field. Below this, there's a small image of a mountain and the text "Insight from your images".



# Image Recognition As a Service

The diagram illustrates the process of image recognition. On the left, a small thumbnail image of a white daisy flower is shown. A large grey arrow points from this thumbnail to a larger, more detailed image of the same daisy flower. To the right of the images is a table titled "Results" which lists the recognized tags and their corresponding probabilities.

Tag	Probability
daisy	99.9%
trillium	3.1%
lily of the valley	0.1%
dogwood	0.0%

<https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/>

# Image Recognition Service API

The screenshot shows the Microsoft Cognitive Services API landing page. At the top, there is a navigation bar with links for 'Directory', 'Pricing', 'Documentation', and 'Log in'. Below this, three steps are outlined: '1 Select your API' (highlighted in blue), '2 Get an API key' (gray), and '3 Start using the API' (gray). A horizontal navigation bar below these steps includes tabs for 'Vision APIs' (selected), 'Speech APIs', 'Language APIs', and 'Search APIs'. On the left, a section for 'Computer Vision' is shown, featuring a camera icon and the text 'Distill actionable information from images' and '5,000 transactions, 20 per minute.'. To the right of this section is a large blue button labeled 'Get API Key >'. This button is highlighted with an orange rounded rectangle.

Explore Cognitive Services: [Directory](#) [Pricing](#) [Documentation](#) [Log in](#)

1 Select your API    2 Get an API key    3 Start using the API

Vision APIs    Speech APIs    Language APIs    Search APIs

Computer Vision  
Distill actionable information from images  
5,000 transactions, 20 per minute.

[Get API Key >](#)

# Image Recognition Service API (Example #1)

```
1  ...
2  Animal Recognition
3
4  ** With the support of [REDACTED] SDK.
5
6
7  from aip import AipImageClassify
8  import sys
9  import os
10 import time
11 import json
12
13 APP_ID = '117[REDACTED]'[REDACTED]
14 API_KEY = 's3[REDACTED]'[REDACTED]
15 SECRET_KEY = 'od[REDACTED]'[REDACTED]
16 client = AipImageClassify(APP_ID, API_KEY, SECRET_KEY)
17
18
19 def get_file_content(image_path):
20     with open(image_path, 'rb') as fp:
21         return fp.read()
22
23
24 def api_test(image_path):
25     image = get_file_content(image_path)
26     # [REDACTED]imal Recognition
27     content = client.animalDetect(image)
28     return content
29
30
31 def main():
32     image_path = sys.argv[1]
33     response = api_test(image_path)
34     print('Result from the [REDACTED] on API:')
35     print(json.dumps(response, ensure_ascii=False, indent=4, separators=(',', ':')))
36
37
38 if __name__ == '__main__':
39     main()
```

```
import AipImageClassify
client = AipImageClassify(APP_ID, API_KEY, SECRET_KEY)

content = client.animalDetect(image)
return content
```

# Image Recognition Service API (Example #2)

```
5 import requests
6 import sys
7 import os
8 import time
9 import json
10 from PIL import Image
11 from io import BytesIO
12
13
14
15 def api_test(image_path):
16     subscription_key = "c"          3"
17     assert subscription_key
18
19     vision_base_url = "https://e"   /"
20
21     analyze_url = vision_base_url + "analyze"
22
23     # Set image_path to the local path of an image that you want to analyze.
24
25     # Read the image into a byte array
26     image_data = open(image_path, "rb").read()
27     headers    = {'Ocp-Apim-Subscription-Key': subscription_key,
28                   'Content-Type': 'application/octet-stream'}
29     params     = {'visualFeatures': 'Categories,Tags,ImageType,Description,Celebrity'}
30     response = requests.post(
31         analyze_url, headers=headers, params=params, data=image_data)
32     response.raise_for_status()
33
34     rsp = response.json()
35     return rsp
36
37
38 def main():
39     image_path = sys.argv[1]
40     response = api_test(image_path)
41     print('Result'                                in API:')
42     print()
43     print(json.dumps(response, ensure_ascii=False, indent=4, separators=(',', ':')))
```

→ set subscription\_key to “*your valid subscription key*”

→ set recognition request target to regional *service URL*

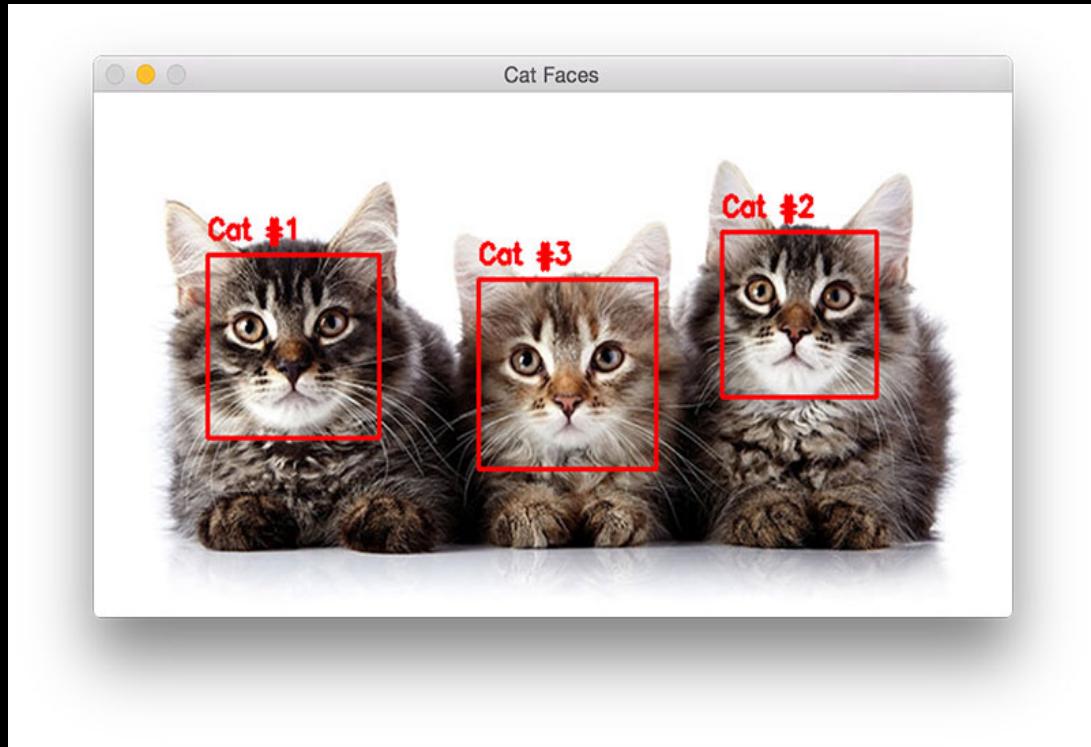
→ `requests.post(analyze_url, headers=headers,  
params=params, json=data)`

# Applications based on Image Recognition Service

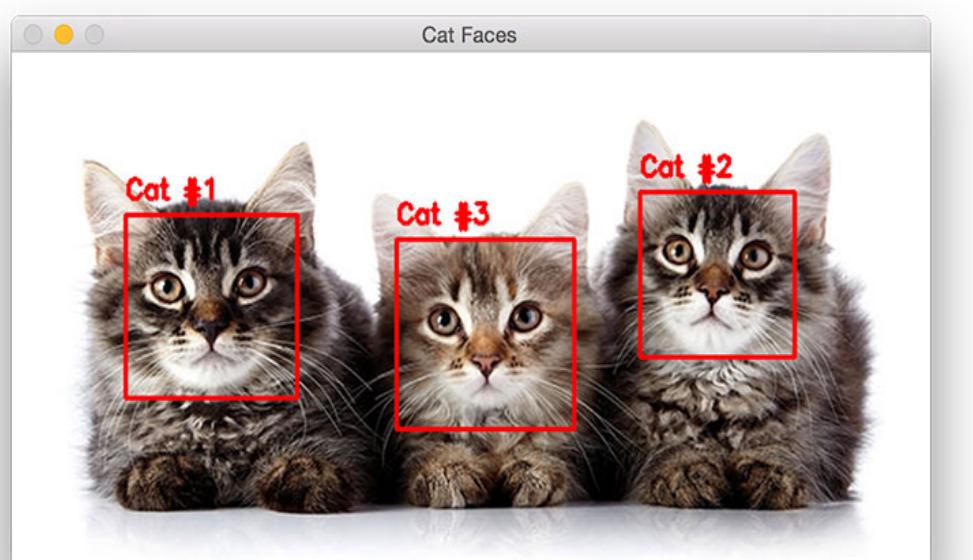
- Image Classifier
- Optical character recognition (OCR) in images
- Object, scene, and activity detection
- Person Identification and Emotion Recognition
- Explicit or offensive content moderation for images

# Attacks on Image Recognition Services

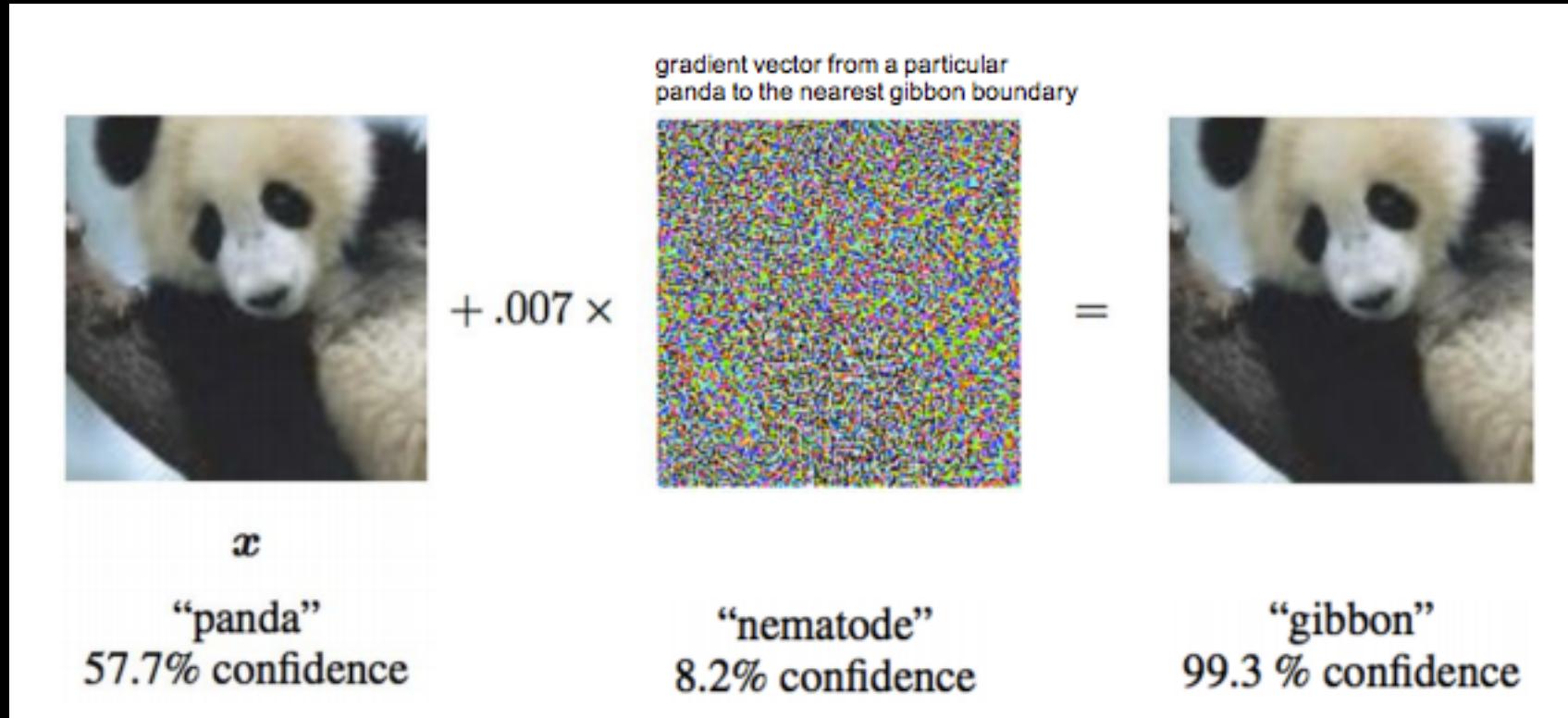
# How to exploit an AI image recognition system?



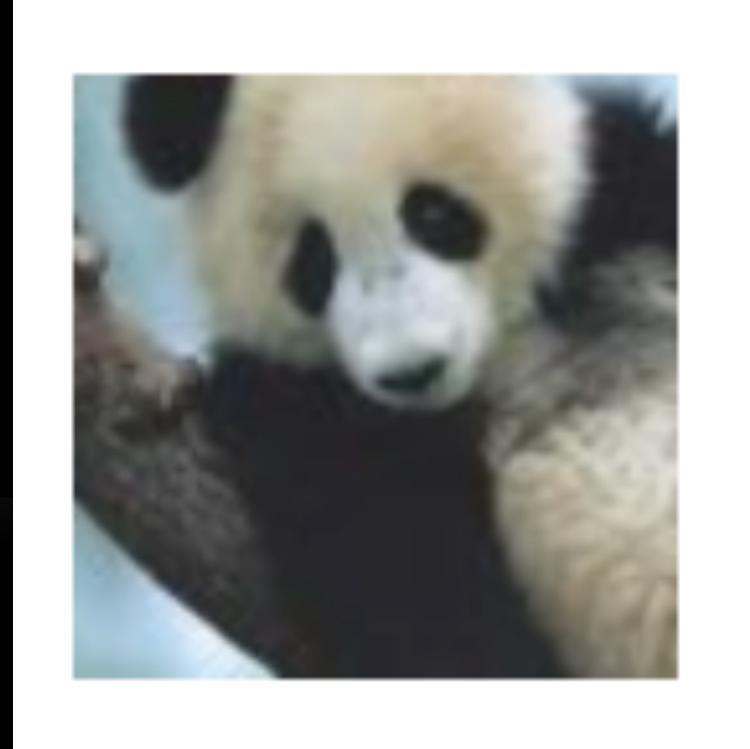
# How to exploit an AI image recognition system?



# The Famous Adversarial ML Example



What if we feed this famous  
adversarial example to  
commercial image  
recognition systems?

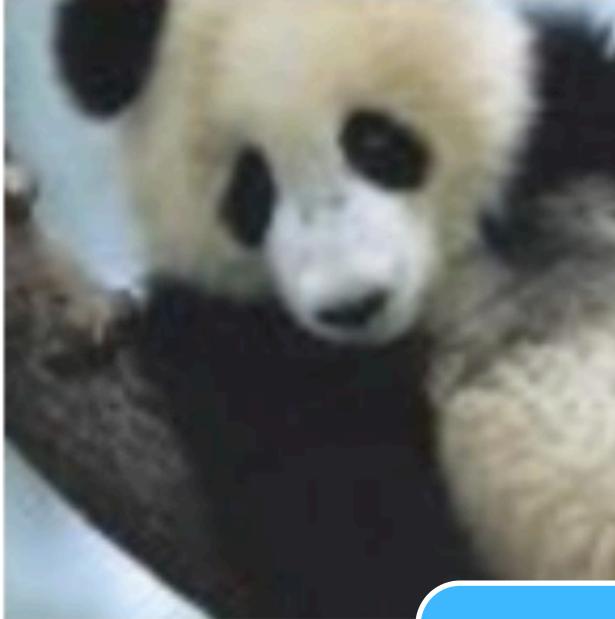


<https://ai.vision/>

Contact Sales: [Contact Sales](#) Search [Search](#) My account Portal Sign in

Overview ▾ Solutions Products Documentation Pricing Training Marketplace ▾ Partners ▾ Support ▾ Blog More ▾

**Products** account ▾



"a panda bear sitting up against a white background", "confidence": 0.7630678

"playing", "blue"], "captions": [ { "text": "a panda bear sitting up against a white background", "confidence": 0.7630678 } ] }

Tags

```
{ "name": "animal", "confidence": 0.9941491 }, { "name": "mammal", "confidence": 0.9867137 }, { "name": "indoor", "confidence": 0.8814776 }, { "name": "white", "confidence": 0.8659121 }, { "name": "giant panda", "confidence": 0.591491759 } ] }
```

# Other Examples from Adversarial ML Papers

Original Image  
(299x299)



'gorilla':  
0.96459390

Adversarial Examples  
(299x299)



'sloth\_bear':  
0.14774416

Theory



'cheeseburger':  
0.91612280



'balloon':  
0.99278200



'Dungeness\_crab':  
0.10425235



'parachute':  
0.52863985

# Hidden Assumption

## Adversarial Examples (299x299)



'sloth\_bear':  
0.14774416



'Dungeness\_crab':  
0.10425235



'parachute':  
0.52863985

# When Adversarial ML meet Reality

Adversarial Examples  
(299x299)

Reality



'sloth\_bear':  
0.14774416



'gorilla':  
0.91946507

Adversarial Examples  
Central Cropped to 87.5%  
(263x263)



'Dungeness\_crab':  
0.10425235



'cheeseburger':  
0.97768340



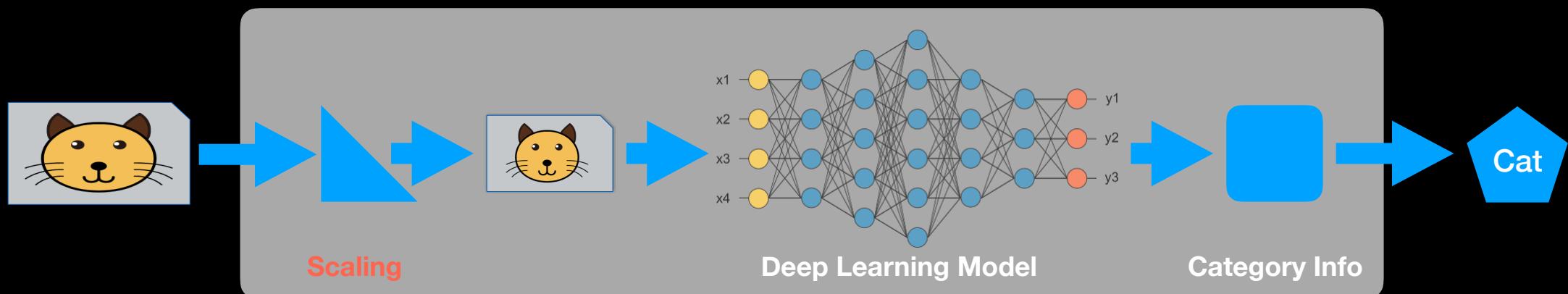
'parachute':  
0.52863985



'balloon':  
0.99310110

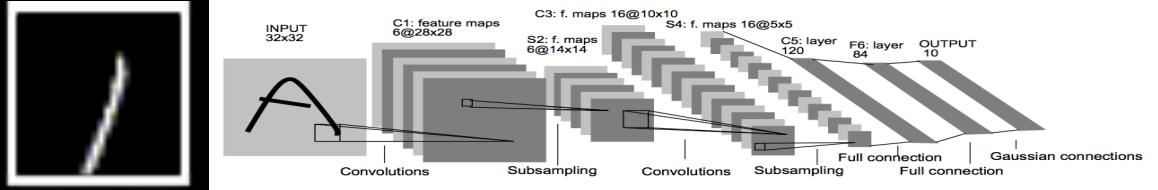
Adversarial samples failed with simple scaling!

# Data Flow in Deep Learning Image Applications



# A Hidden Assumption of Deep Learning Applications

MNIST

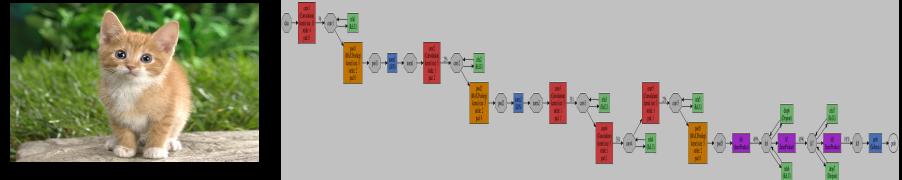


<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

Deep Learning Model Input Requirement  
(pixel x pixel)

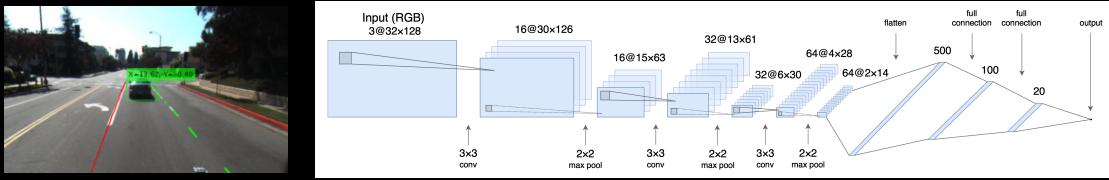
MNIST	28x28
ImageNet	
AlexNet	227x227
GoogleNet, VGG	224x224
ResNet	224x224
NVIDIA	
DAVE-2 Self-Driving	200x66

ImageNet



[https://github.com/BVLC/caffe/tree/master/examples/cpp\\_classification](https://github.com/BVLC/caffe/tree/master/examples/cpp_classification)

NVIDIA PX DAVE-2



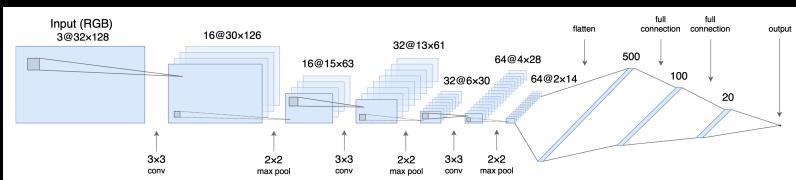
<https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf>



**What if the input size does not match model scale?**

# NVIDIA Self-Driving Models and Input Scales

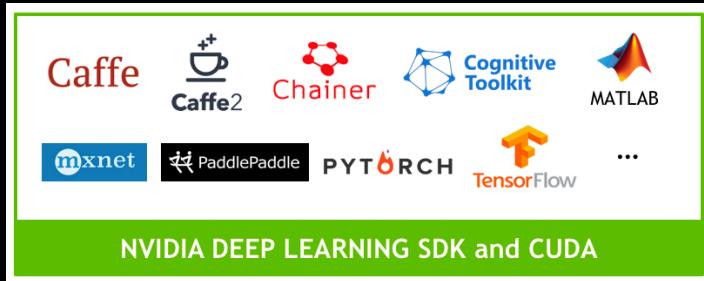
NVIDIA Sample Self-Driving Models



NVIDIA Recommended Ecosystem Camera Vendors



NVIDIA PX2	
DAVE-2 Self-Driving	200x66



Filr	
A310	320x240
A615	640x480
Leopard	
LI-AR0231	1920x1208
SEKONIX	
SF3326-100	1920x1208

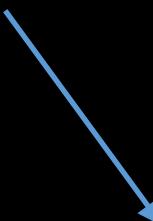


# **Scaling Function might be hidden from Developers**

# Scaling Functions Provided by Frameworks

## TensorFlow Example

```
1 def read_tensor_from_image_file(file_name, input_height=299, input_width=299,
2                                 input_mean=0, input_std=255):
3     input_name = "file_reader"
4     output_name = "normalized"
5     file_reader = tf.read_file(file_name, input_name)
6     if file_name.endswith(".png"):
7         image_reader = tf.image.decode_png(file_reader, channels = 3,
8                                           name='png_reader')
9     elif file_name.endswith(".gif"):
10        image_reader = tf.squeeze(tf.image.decode_gif(file_reader,
11                                  name='gif_reader'))
12    elif file_name.endswith(".bmp"):
13        image_reader = tf.image.decode_bmp(file_reader, name='bmp_reader')
14    else:
15        image_reader = tf.image.decode_jpeg(file_reader, channels = 3,
16                                            name='jpeg_reader')
17    float_caster = tf.cast(image_reader, tf.float32)
18    dims_expander = tf.expand_dims(float_caster, 0);
19    resized = tf.image.resize_bilinear(dims_expander, [input_height, input_width])
20    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
21    sess = tf.Session()
22    result = sess.run(normalized)
23
24    return result
```



`tf.image.resize_bilinear(dims_expander, [input_height, input_width])`

# Scaling Functions Provided by Frameworks

## DeepDetect Example

```
1 int read_file(const std::string &fname)
2 {
3     cv::Mat img = cv::imread(fname,_bw ? CV_LOAD_IMAGE_GRAYSCALE :
4                               CV_LOAD_IMAGE_COLOR);
5     if (img.empty())
6     {
7         LOG(ERROR) << "empty image";
8         return -1;
9     }
10    _imgs_size.push_back(std::pair<int,int>(img.rows,img.cols));
11    cv::Size size(_width,_height);
12    cv::Mat rimg;
13    cv::resize(img,rimg,size,0,0,CV_INTER_CUBIC);
14    _imgs.push_back(rimg);
15    return 0;
16 }
```

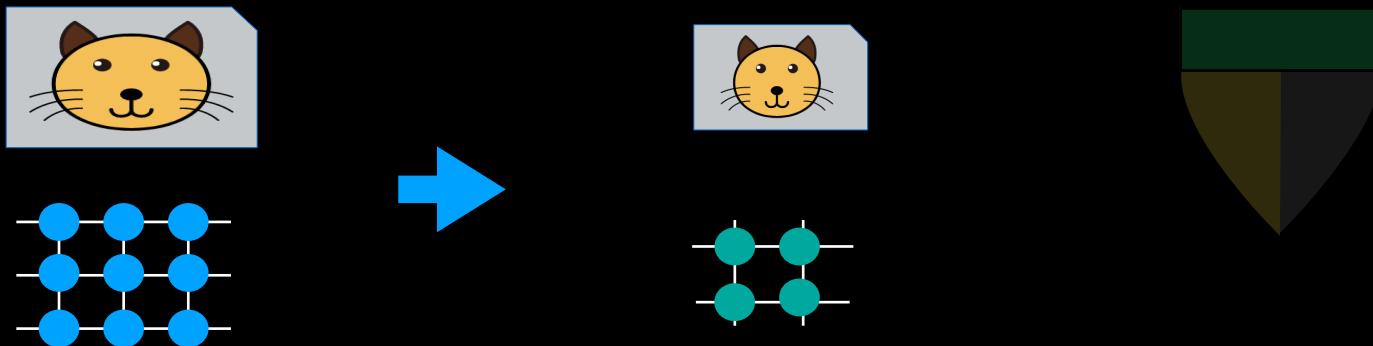
resize(img,rimg,size,0,0,CV\_INTER\_CUBIC);

# Common Scaling Algorithms and Scaling Attacks

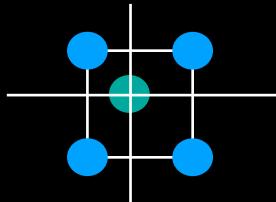
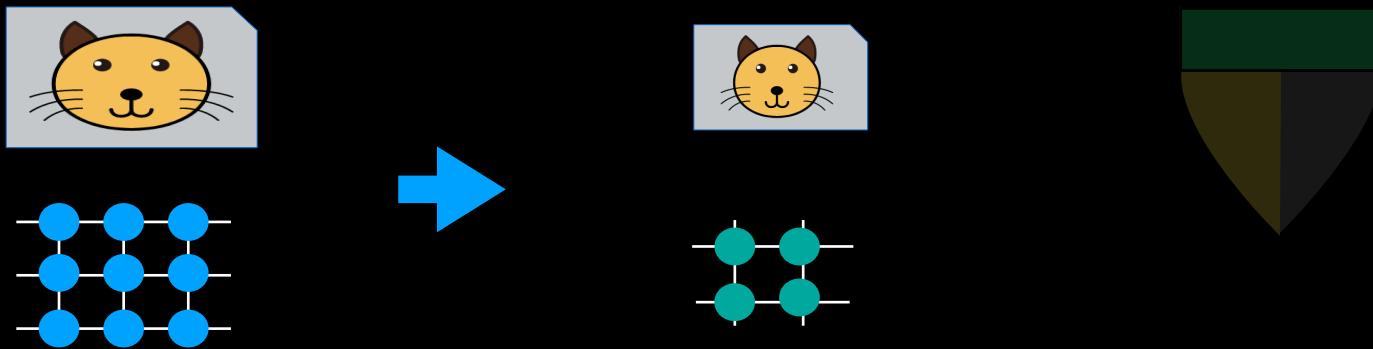


# Scaling and Interpolation Algorithms

Scaling is supposed to preserve the visual features of an image and thus does not change its **semantic** meaning.



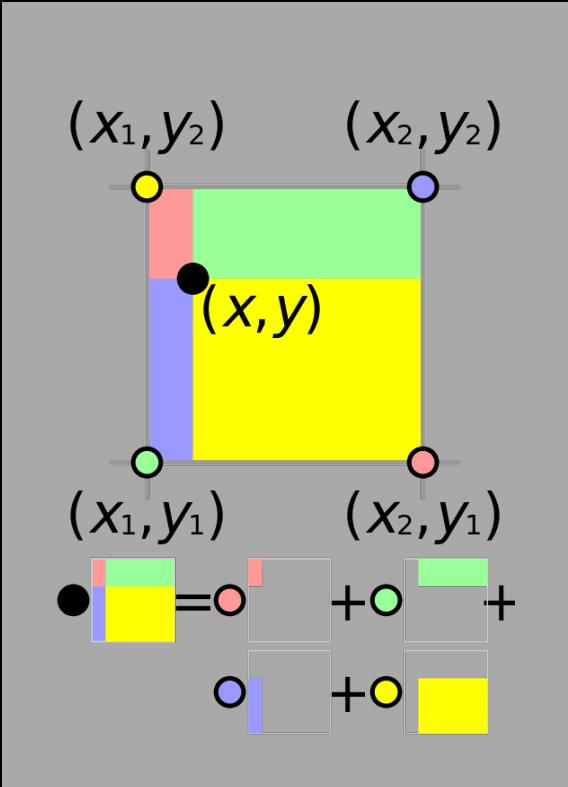
# Scaling and Interpolation Algorithms



**Interpolation:** infer the pixel value at each missing point

Goal: to preserve visual features (and hopefully the semantic meanings)

# Popular Scaling Algorithms



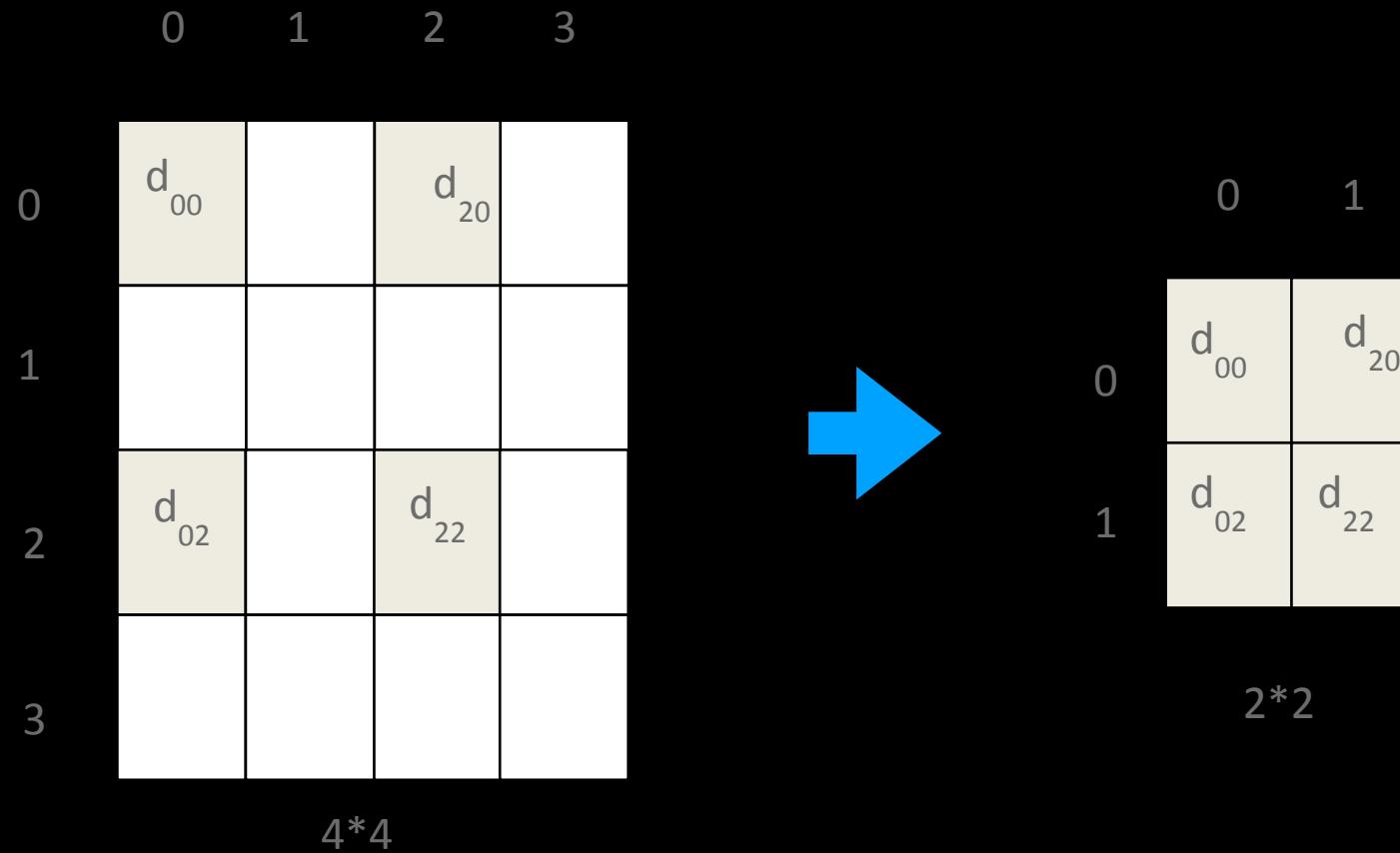
## Bilinear Interpolation:

Value at  $(x, y) =$   
Sum of the value at each spot multiplied by  
the area of the rectangle divided by the total  
area of all four rectangles

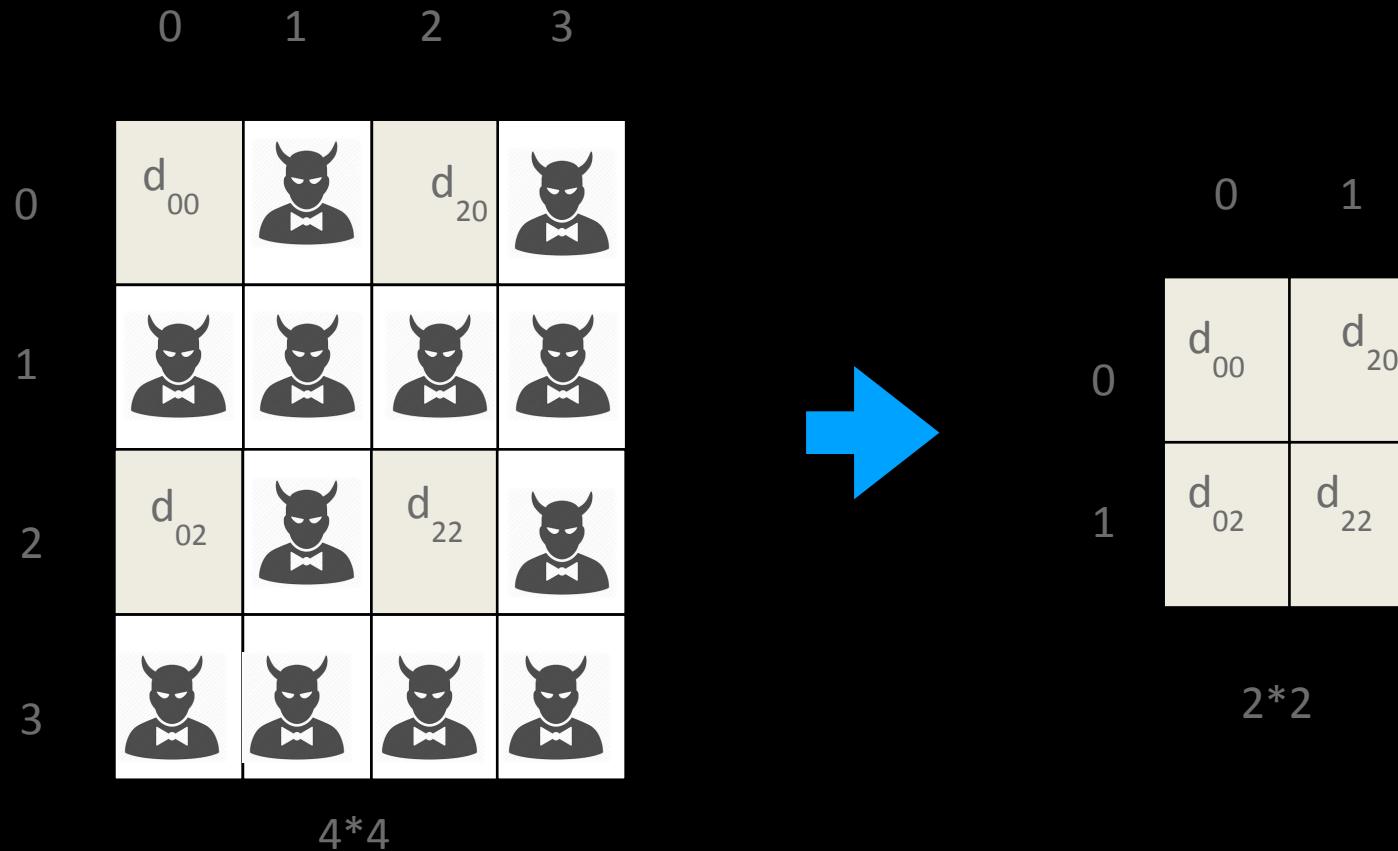
[https://en.wikipedia.org/wiki/Bilinear\\_interpolation](https://en.wikipedia.org/wiki/Bilinear_interpolation)

$$f(i+u, j+v) = (1-u)(1-v)f(i, j) + (1-u)v f(i, j+1) + u(1-v)f(i+1, j) + uv f(i+1, j+1)$$

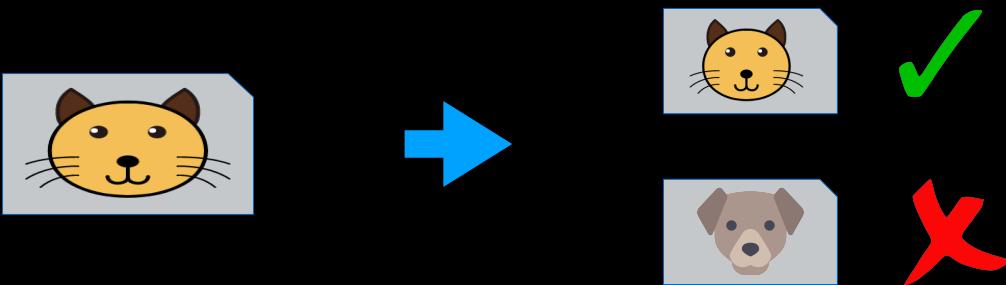
# Nearest Neighbor Scaling Algorithm



# Consequence of Scaling

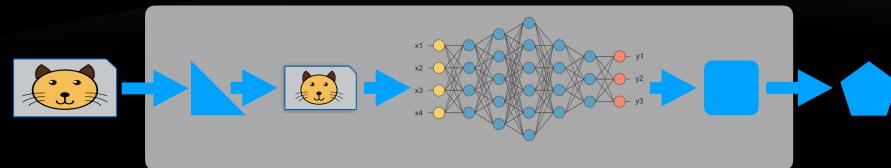


# Examples of Scaling Effect

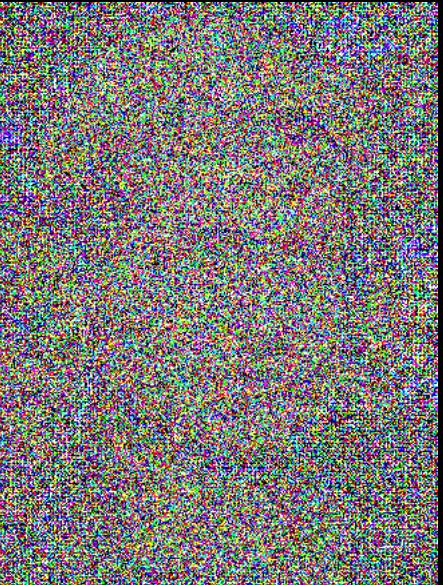


Scaling is not supposed to change the **semantic** meaning of the input image

If we know the scaling algorithms and sizes ...

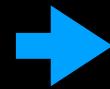


# Attack Leveraging the Scaling Effect (prior work)

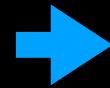


Data Scaling Attacks in Deep Learning Applications  
<https://www.defcon.org/html/defcon-china/dc-cn-speakers.html#LiKang>

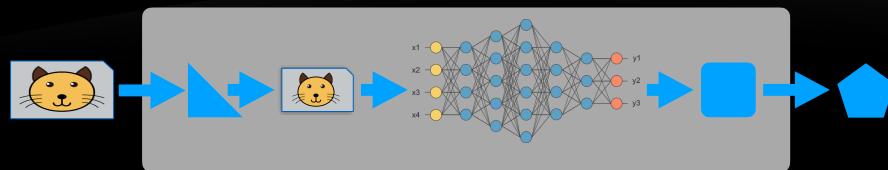
# Attack Sample #2 (Traffic Sign)



# Attack Sample #2 (Traffic Sign)



# How to infer the scaling factor in cloud services?



# Inferring Preprocessing Parameters



inferring parameters by sending queries  
and observe responses

attacker

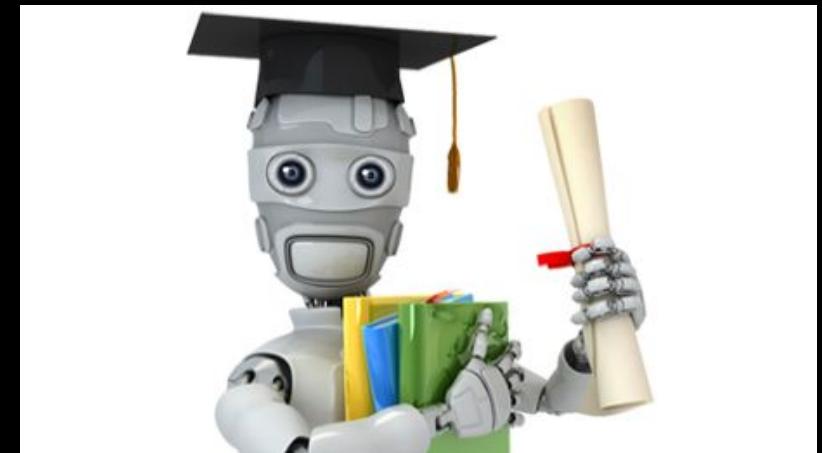


image recognition service

# Inferring Preprocessing Parameters

***"What can you see?"***



attacker



***{probElm<sub>1</sub>}***

(1024->200,bilinear)

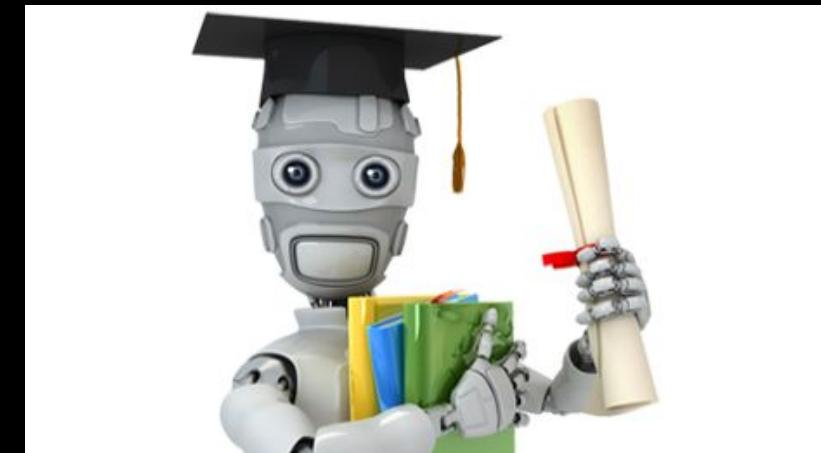


image recognition service

Using specially crafted images:  
meaningful if scaling with the appropriate parameters

# Inferring Preprocessing Parameters

***"What can you see?"***

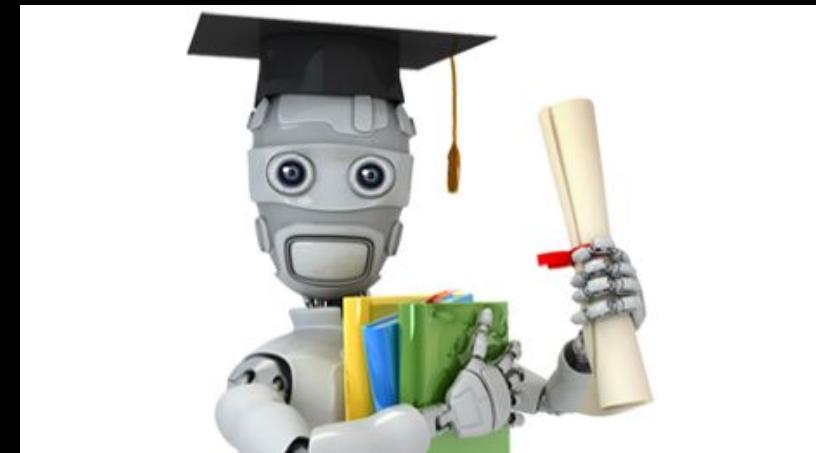


***{probImg<sub>1</sub>}***

(1024->200,bilinear)



***"A meaningless image."***



attacker

image recognition service

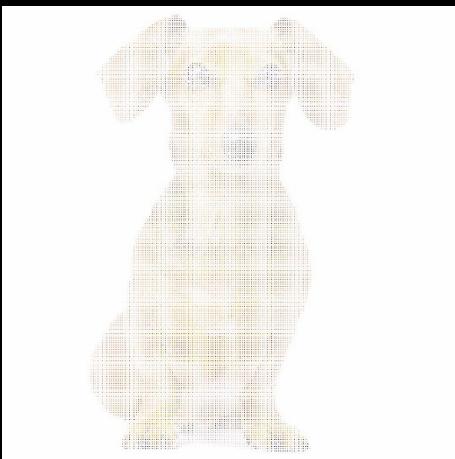
Using specially crafted images:  
meaningful if scaling with the appropriate parameters

# Inferring Preprocessing Parameters

*"What can you see?"*



attacker



**{probImg<sub>1</sub>}**

(1024->200, bilinear)



**{probImg<sub>k</sub>}**

(1024->201, bilinear)

**{probImg<sub>k</sub>}**

(1024->202, bilinear)

*"A meaningless image."*

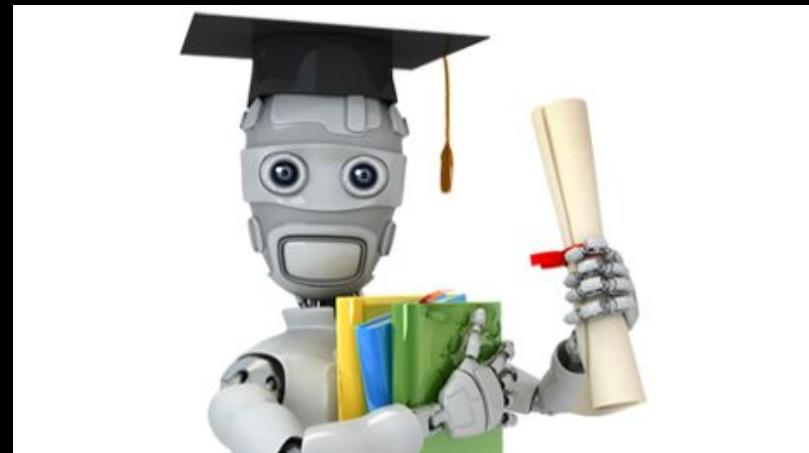
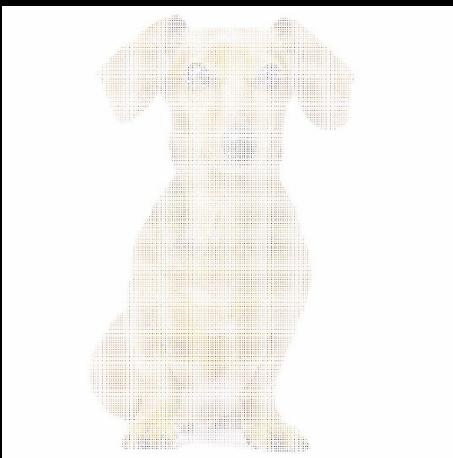


image recognition service

# Inferring Preprocessing Parameters

***“What can you see?”***

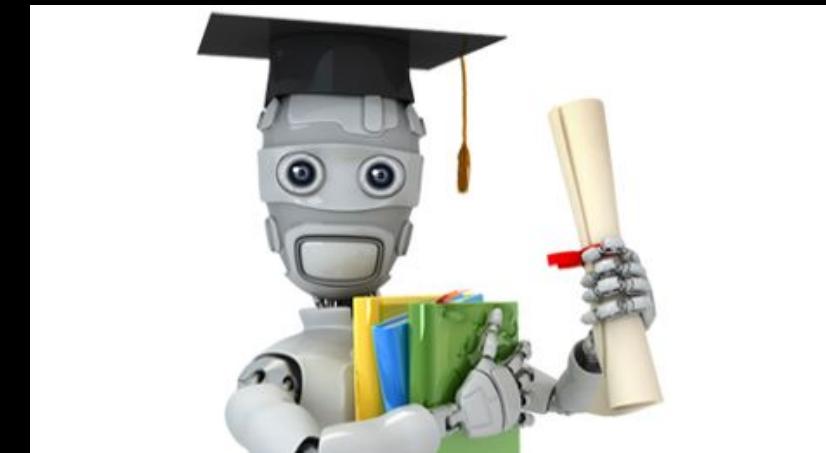


***{probImg<sub>k</sub>}***

(1024->202, bilinear)



***“A dog.”***



attacker

image recognition service

# Inferring Preprocessing Parameters

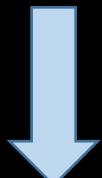
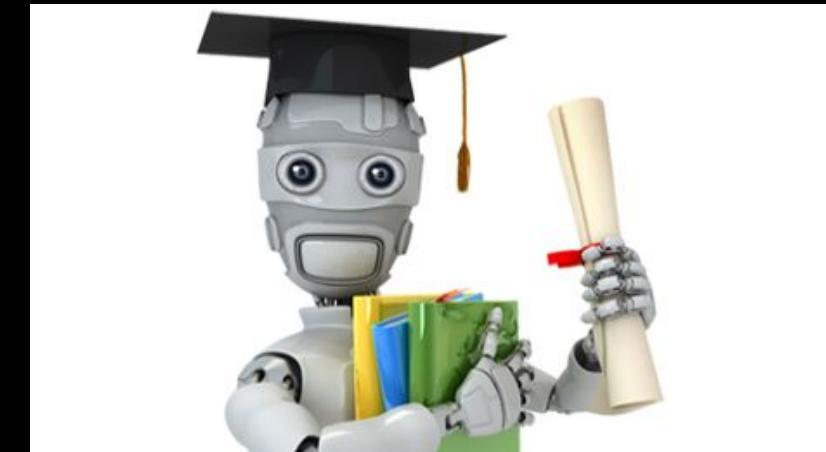
***“What can you see?”***



***“A dog.”***

***{probImg<sub>k</sub>}***

(1024->202, bilinear)

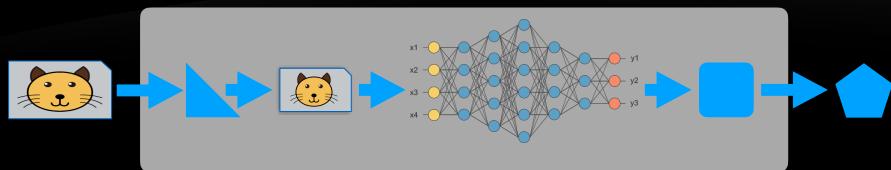


***“Scaling method: bilinear  
Model Input Size: 202x202”***

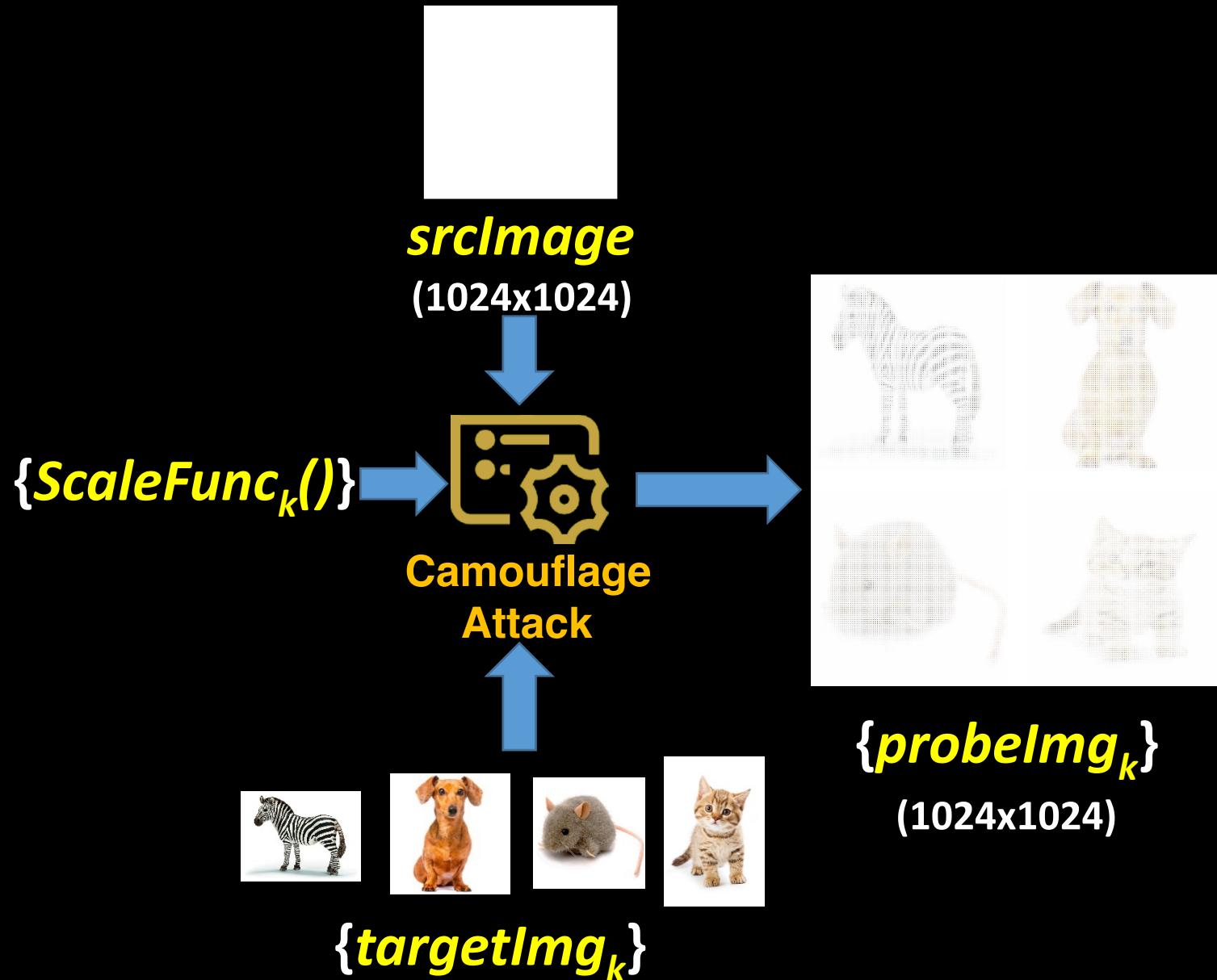
**Brute-force Scaling Attack**



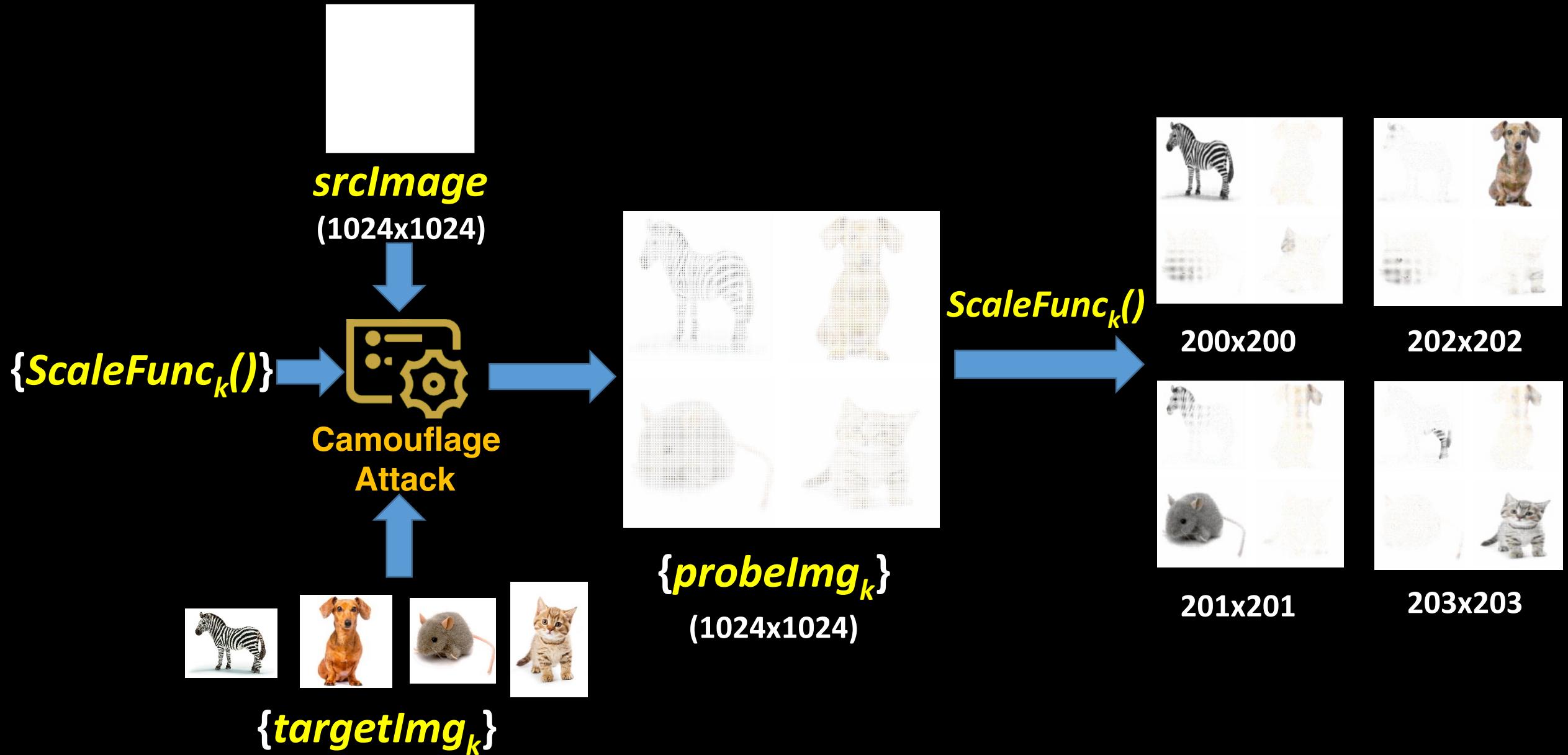
# The efficiency of brute-force inference is low



# Improving Efficiency by Overlay Scaling Attacks

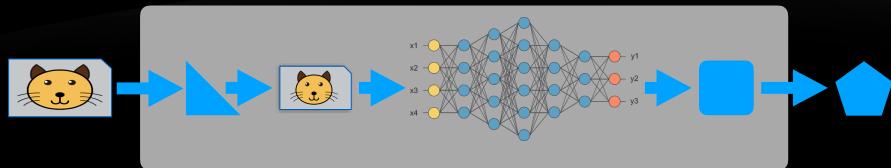


# Improving Efficiency by Overlay Scaling Attacks

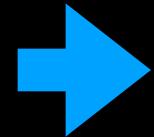




# Once we know the preprocessing parameters

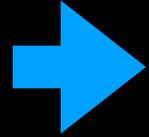


# Attack Effect Examples

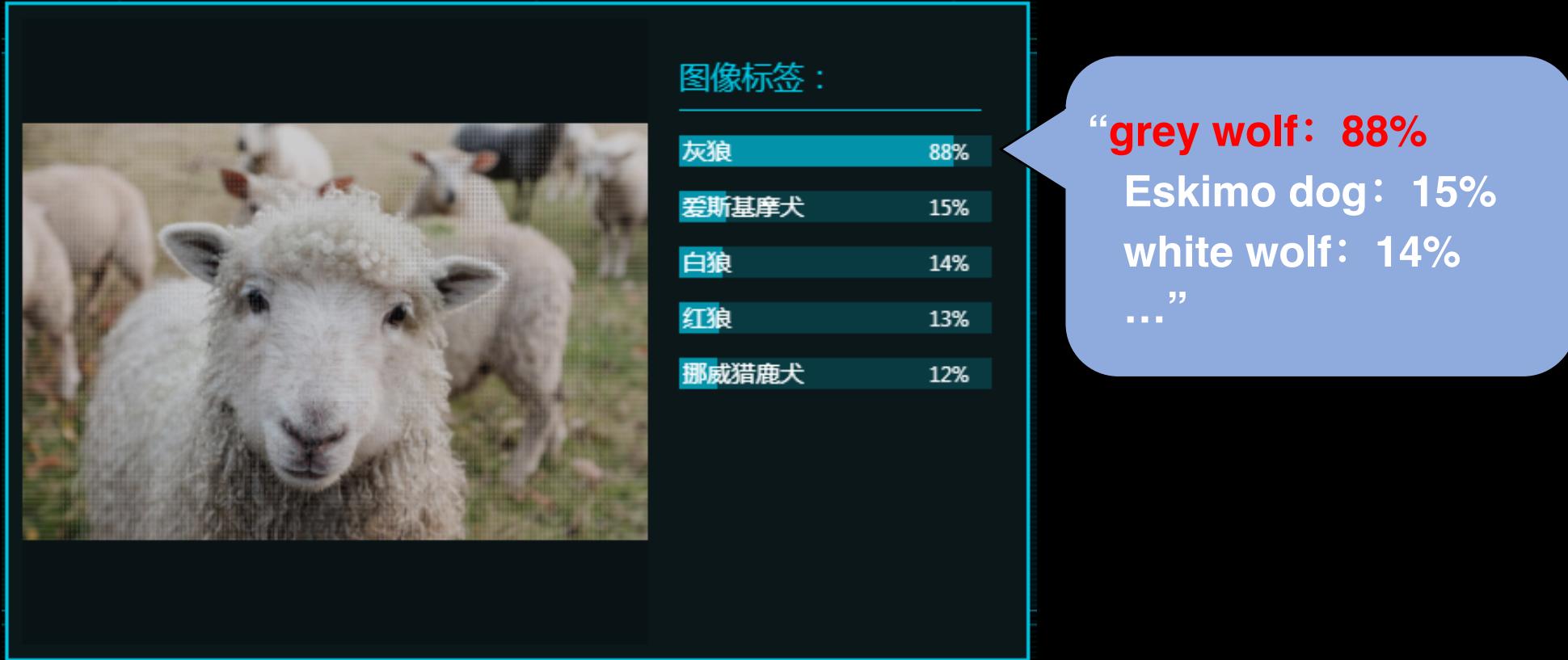


**Result from AI Image  
Recognition Services ?**

# Attack Effect Examples



# Attack Effect Example (Vendor A)



# Attack Effect Example (Vendor B)

A close-up photograph of a white sheep's face. A semi-transparent blue overlay displays the recognition results. The results show "灰狼" (Grey Wolf) at 0.939, "墨西哥狼" (Mexican Wolf) at 0.015, "北极狼" (Arctic Wolf) at 0.011, "哈士奇犬" (Siberian Husky) at 0.009, and "郊狼" (Coyote) at 0.005.

识别结果

灰狼	0.939
墨西哥狼	0.015
北极狼	0.011
哈士奇犬	0.009
郊狼	0.005

请输入网络图片URL      检测      或      上传图片

“grey wolf: 0.939  
mexican wolf: 0.015  
arctic wolf: 0.011  
...”

# Attack Effect Example (Vendor M)



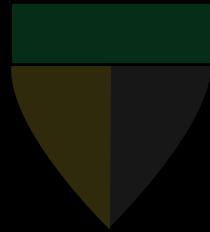
FEATURE	VALUE
NAME:	
Description	{ "tags": [ "animal", "mammal", "wolf", "looking" ], "captions": [ { "text": "a close up of a wolf", "confidence": 0.707954049 } ] }
Tags	[ { "name": "animal", "confidence": 0.9989328 }, { "name": "mammal", "confidence": 0.9908992 }, { "name": "wolf", "confidence": 0.981169641 } ]
Image format	"Jpeg"
Image dimensions	1024 x 1024
Clip art	0

- ✓ Description: { "tags": [ "animal", "mammal", **wolf**, "looking" ], "captions": [ { "text": "a close up of a wolf", "confidence": 0.707954049 } ] }
- ✓ Tags: [ ..., { "name": "wolf", "confidence": 0.981169641 } ]

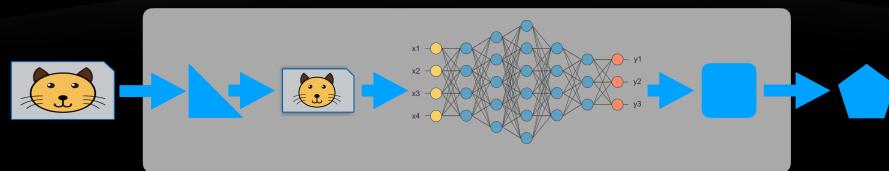
# Attack Effect Example (Vendor T)



“white wolf: 98.52%  
gray wolf: 0.50%  
arctic fox: 0.40%  
...”



# Image Scaling Attack Toolkit



# Summary

- AI-based image recognition services are getting increasingly popular
- Samples from adversarial ML **fails** to fool commercial image API
- Image pre-processing is widely and often implicitly used in AI-based image recognition services
- Attackers can infer image recognition service parameters and launch effective evading attacks

