In [1]:
```
from segment_anything import SamAutomaticMaskGenerator, SamPredictor, sam_model
import cv2
import numpy as np
import matplotlib.pyplot as plt
import torch
import cv2
import os
import random
import scipy
```

```
/homes/e34960/anaconda3/envs/patchdiff/lib/python3.9/site-packages/tqdm/auto.p
y:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets.
See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
gt_images_folder = "/project/trinity/datasets/apricot/pub/apricot-mask/data_mas
gt_images_names = os.listdir(gt_images_folder)

print(len(gt_images_names))
```

```
873
```

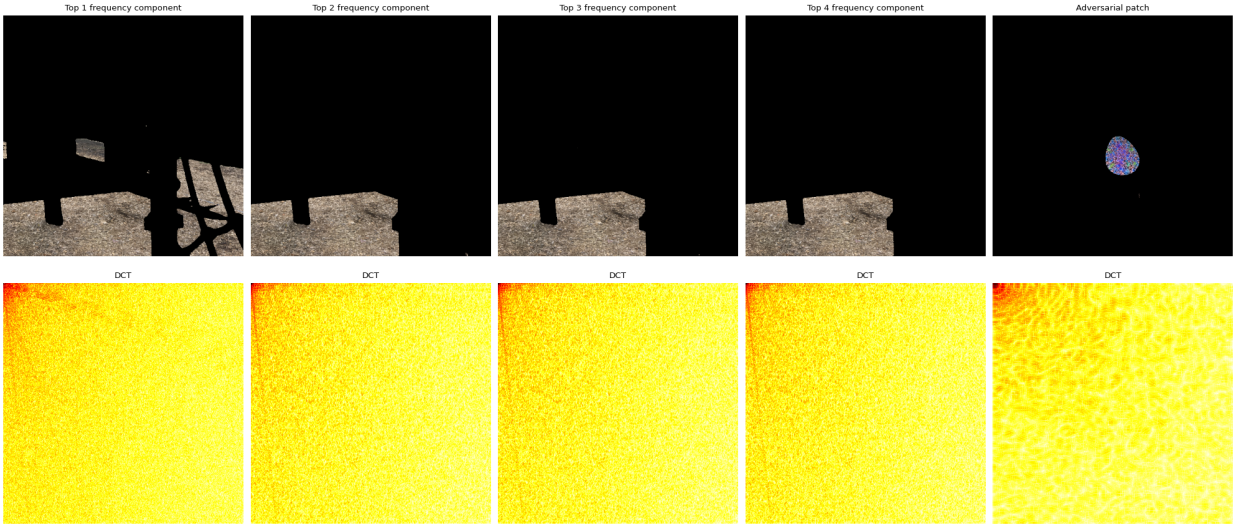
In [3]:
```
def compute_vectorized_iou(given_mask, masks):
    given_mask_expanded = np.expand_dims(given_mask, axis=0)
    masks_expanded = np.stack(masks)
    intersection = np.logical_and(masks_expanded > 0, given_mask_expanded > 0)
    union = np.logical_or(masks_expanded > 0, given_mask_expanded > 0)
    intersection_sum = np.sum(intersection, axis=(1, 2))
    union_sum = np.sum(union, axis=(1, 2))
    iou_values = intersection_sum / union_sum
    return iou_values

def getRandomImages(n=1):
    random_numbers = random.sample(range(0, len(gt_images_names)-1), n)

    imgs = []
    img_masks = []
    for i in range(n):
        # idx = random.randint(0,len(gt_images_names)-1)
        img_info = torch.load(os.path.join(gt_images_folder, gt_images_names[ra
        img = np.squeeze(img_info['Image'])
        img = np.uint8(img * 255.0)
        img = cv2.resize(img, (512,512), interpolation = cv2.INTER_AREA)
        imgs.append(img)

        img_mask = np.squeeze(img_info['Mask'])
        img_mask = np.uint8(img_mask * 255.0)
        img_mask = cv2.resize(img_mask, (512,512), interpolation = cv2.INTER_AR
        img_masks.append(img_mask)

    return imgs, img_masks
```

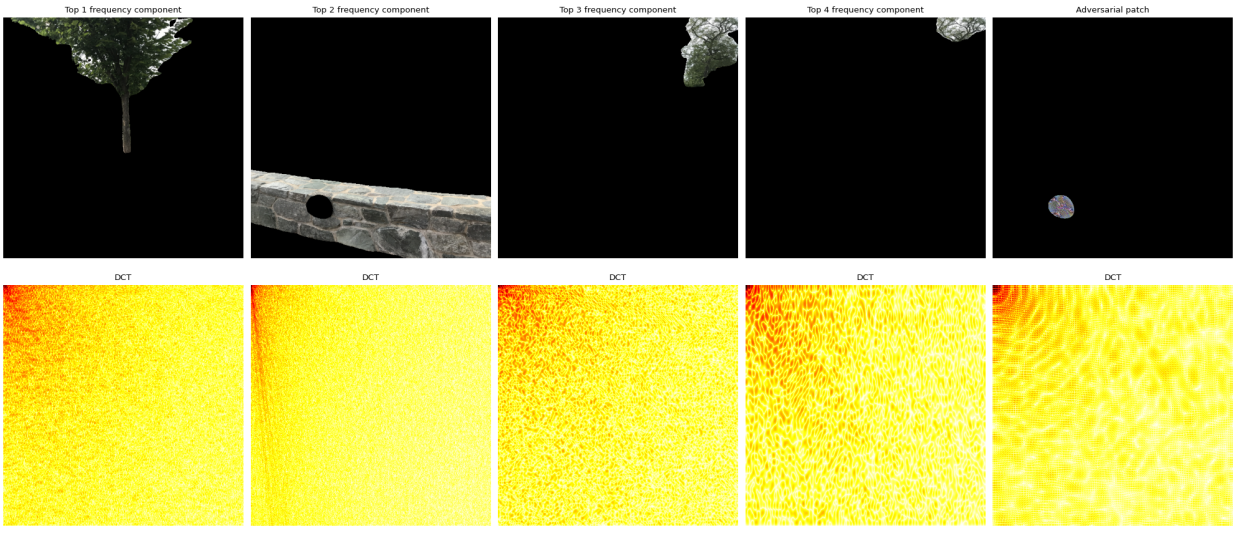
In [4]:
```
sam = sam_model_registry["vit_h"](checkpoint="/project/trinity/pretrained_model
sam.to(device='cuda')
mask_generator = SamAutomaticMaskGenerator(sam)
```

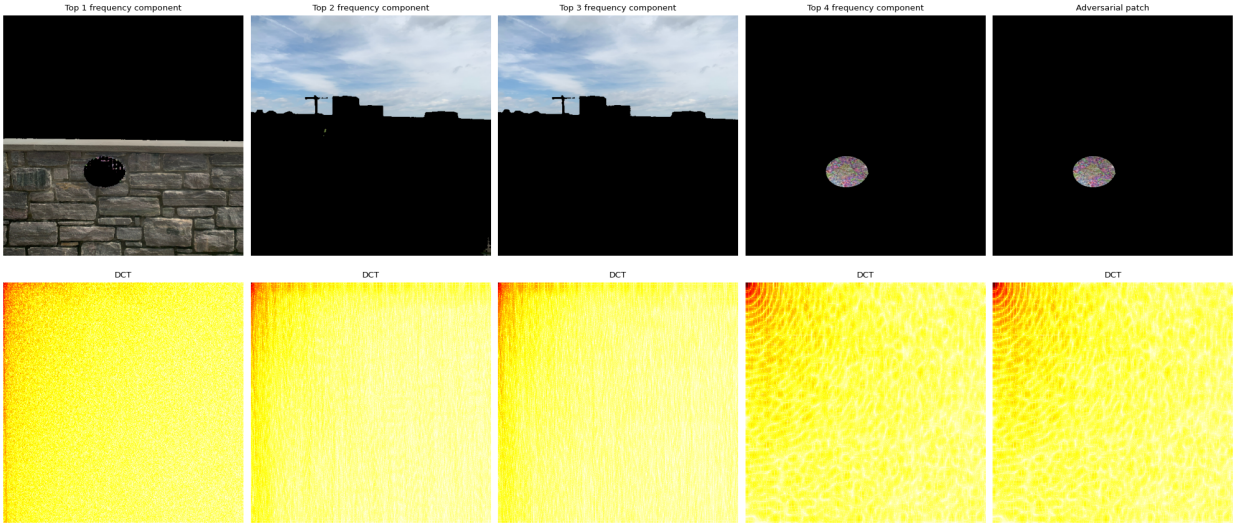
In [55]:
```
num_images = 20
img_list, mask_list = getRandomImages(num_images)
```

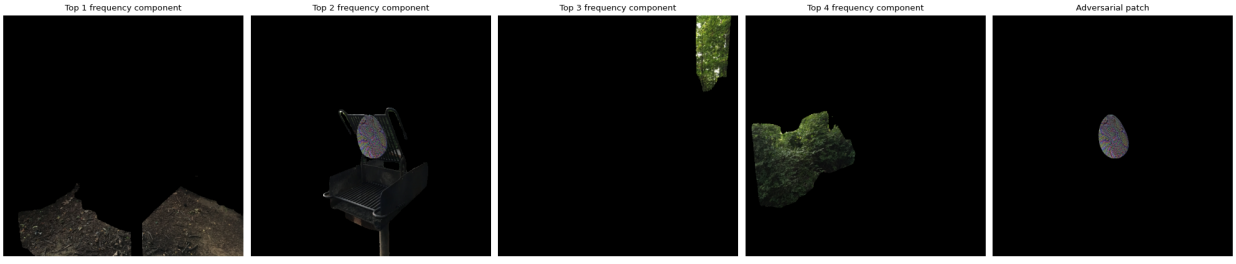
```
patch_indices = []

for i in range(num_images):
    masks = mask_generator.generate(img_list[i])
    all_masks = []
    for z, meta_info in enumerate(masks):
        all_masks.append(meta_info['segmentation'])

    iou_vals = compute_vectorized_iou(mask_list[i], all_masks)
    patch_mask_idx = np.argmax(iou_vals)

    # Count non-zero pixels in patch
    # patch_mask_count = np.sum(all_masks[patch_mask_idx] != 0)
    # print(patch_mask_count)

    max_frequencies_list = []
    dct_list = []
    masked_img_list = []

    for j in range(len(masks)):
        current_mask = masks[j]['segmentation']

        # Count non-zero pixels in mask
        # current_mask_count = np.sum(current_mask != 0)
        # print(current_mask_count)
        # if current_mask_count >= patch_mask_count/2:

        mask_3d = np.repeat(current_mask[:, :, np.newaxis], 3, axis=2)
        masked_img = img_list[i] * mask_3d

        gray_scale = cv2.cvtColor(masked_img, cv2.COLOR_BGR2GRAY)

        magnitude_spectrum = scipy.fftpack.dct( scipy.fftpack.dct(gray_scale, a

        # TODO Thresholding
        # magnitude_spectrum = magnitude_spectrum * (abs(magnitude_spectrum) >

        magnitude_spectrum = np.log1p(np.abs(magnitude_spectrum))
        dct_list.append(magnitude_spectrum)
        masked_img_list.append(masked_img)

        max_frequencies_list.append(np.sum(magnitude_spectrum > 3))


    dct_list = np.stack(dct_list, axis=-1)
    dct_list = 255.0 * (np.max(dct_list) - dct_list) / (np.max(dct_list) - np.m

    # Highest Frequency for the patch mask
    patch_mask_freq = max_frequencies_list[patch_mask_idx]

    max_frequencies_list = np.array(max_frequencies_list)
    sorted_max_frequencies_list = np.sort(max_frequencies_list)[::-1]
    sorted_indices = np.argsort(max_frequencies_list)[::-1]

    # print(patch_mask_freq)
    # print(sorted_max_frequencies_list)

    patch_indices.append(np.where(sorted_max_frequencies_list == patch_mask_fre
    plt.figure(figsize=(4,3))
```

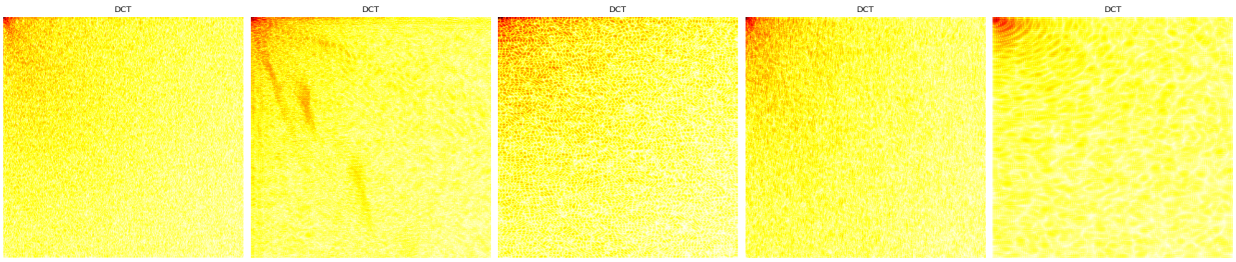
```
    plt.imshow(img_list[i])
    # plt.title('Original Image')
    plt.axis('off')
    plt.show()

    ###### Plot top 4 frequencies and the adversarial patch frequency (magnitud
    fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 9))
    for k in range(5):
        if k != 4:
            axes[0][k].imshow(masked_img_list[sorted_indices[k]], cmap='gray')
            axes[0][k].set_title('Top {} frequency component'.format(k+1))
            axes[0][k].axis('off')

            axes[1][k].imshow(dct_list[:,:,sorted_indices[k]], cmap='hot', vmax
            axes[1][k].set_title('DCT')
            axes[1][k].axis('off')

        # Adversarial patch case
        else:
            axes[0][k].imshow(masked_img_list[patch_mask_idx], cmap='gray')
            axes[0][k].set_title('Adversarial patch')
            axes[0][k].axis('off')

            axes[1][k].imshow(dct_list[:,:,patch_mask_idx], cmap='hot', vmax=np
            axes[1][k].set_title('DCT')
            axes[1][k].axis('off')

    plt.tight_layout()
    plt.show()
```

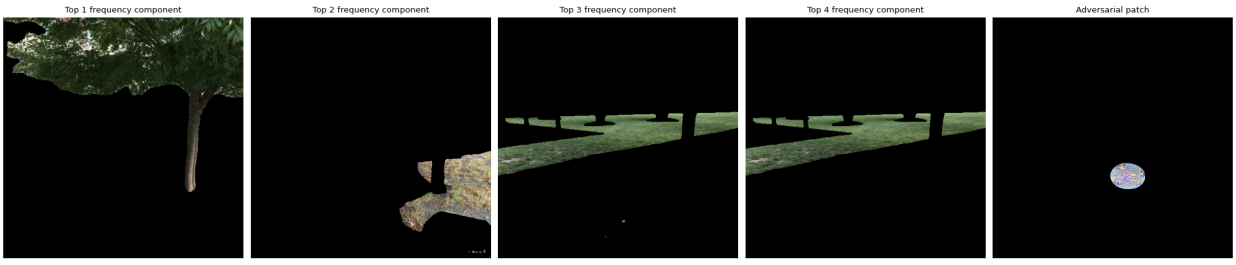

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

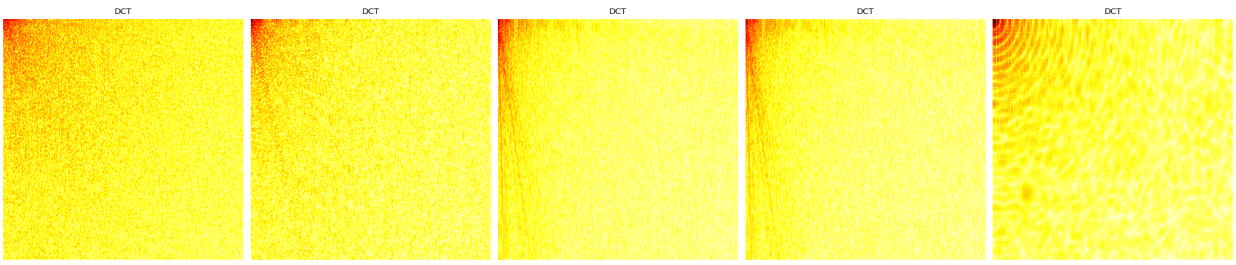

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT


Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

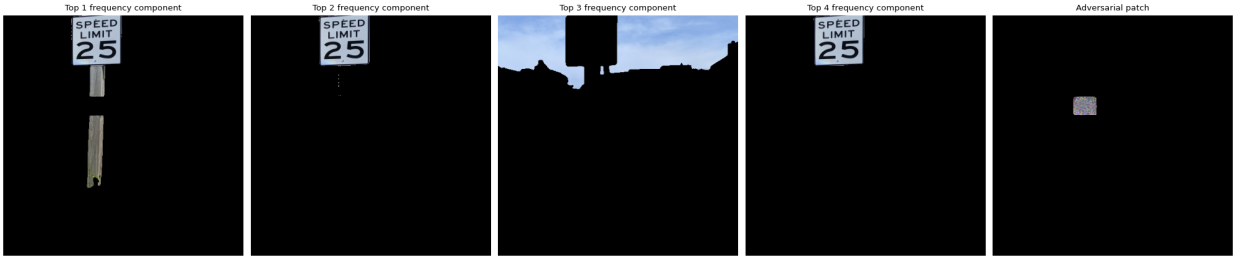

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

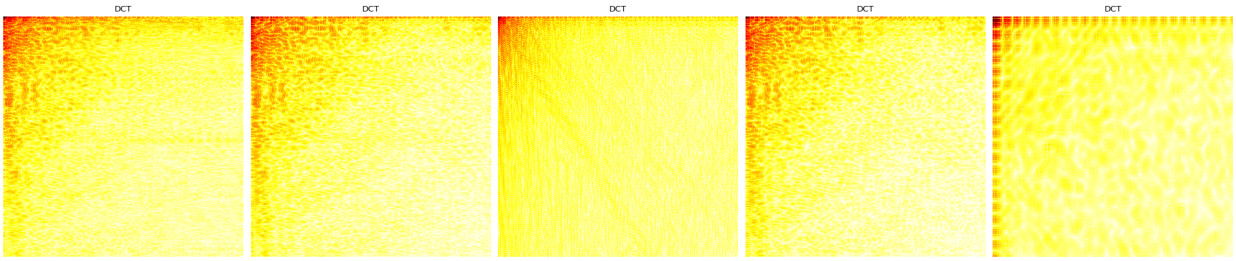

DCT
DCT
DCT
DCT
DCT

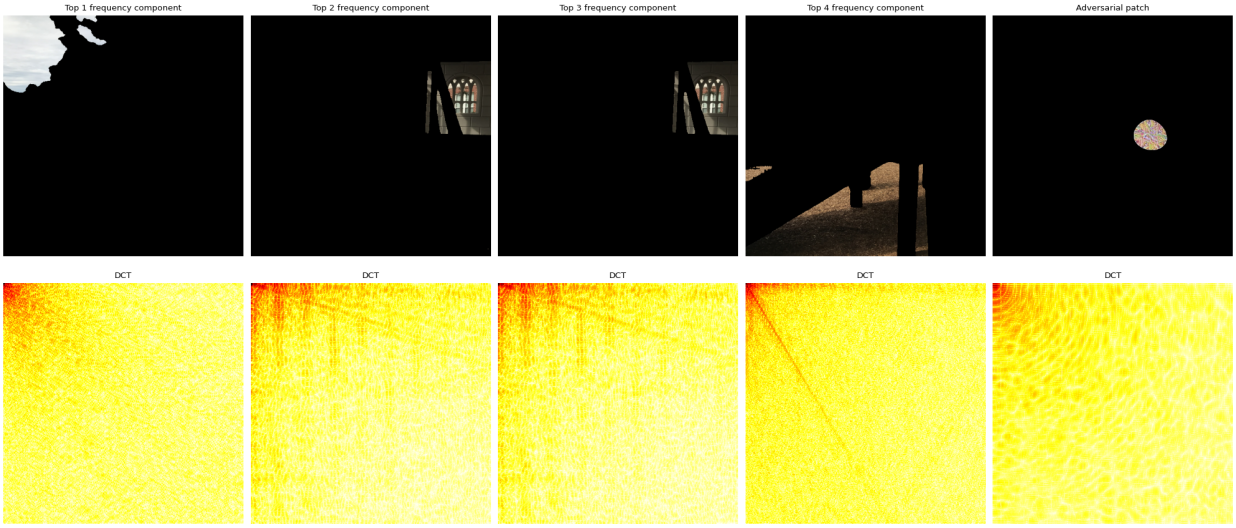

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

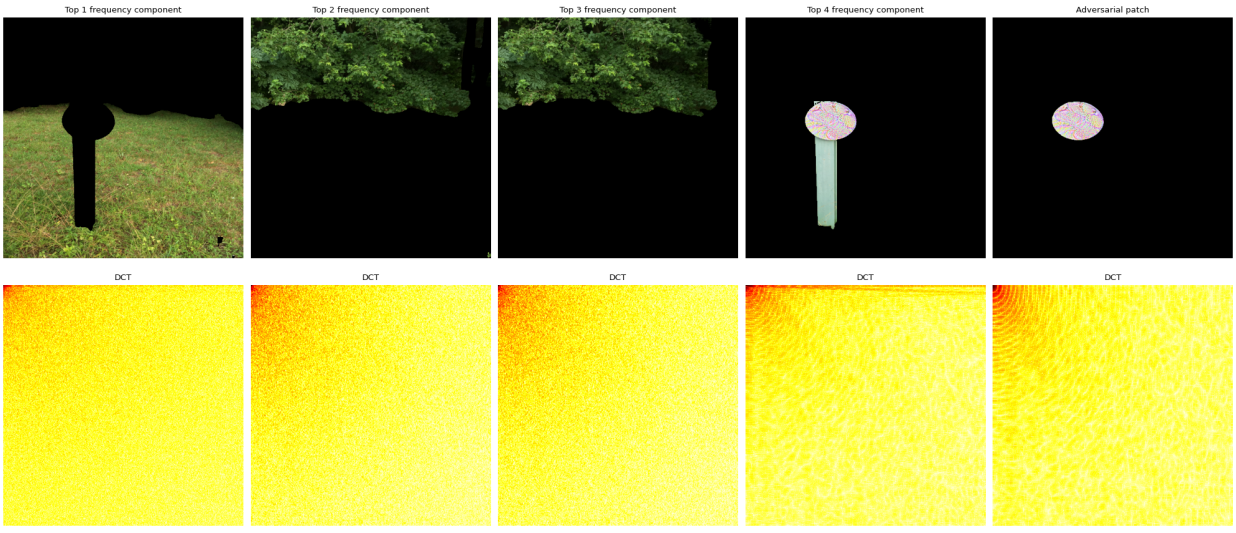

DCT
DCT
DCT
DCT
DCT

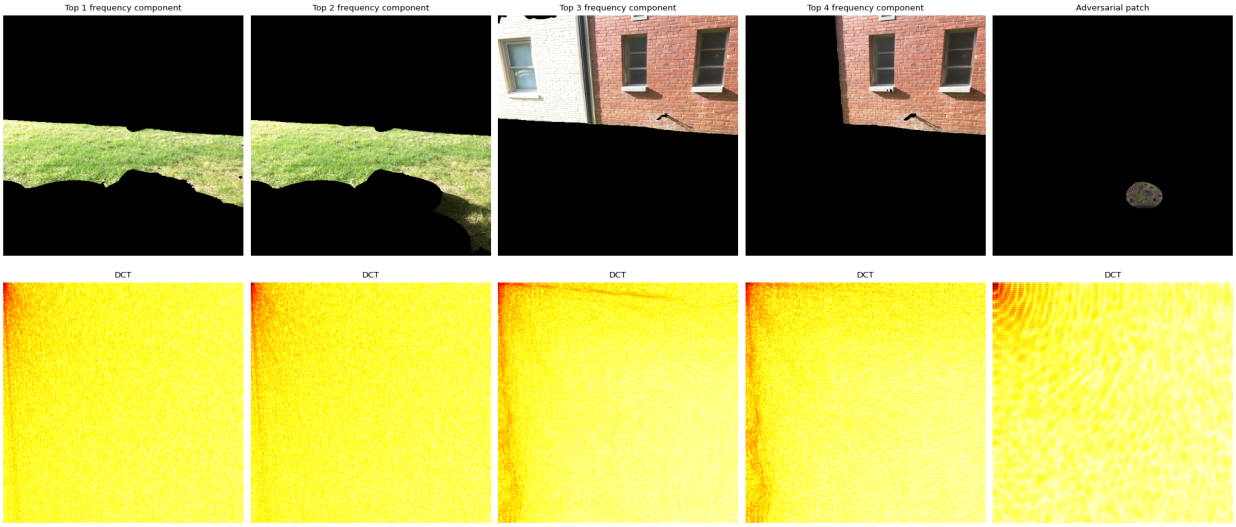

SPEED
LIMIT
25

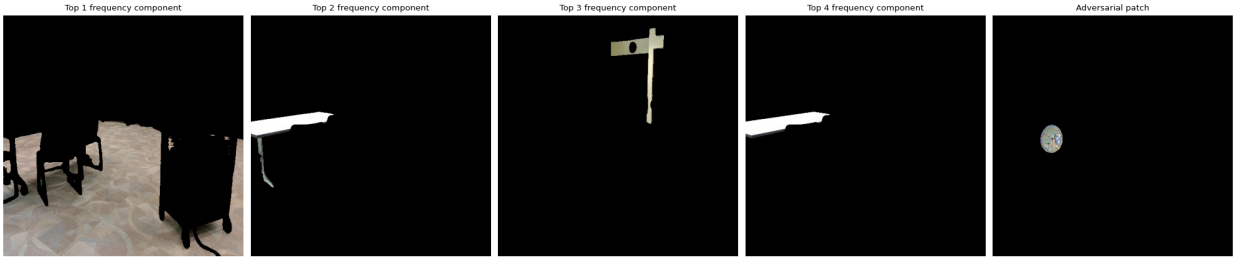

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

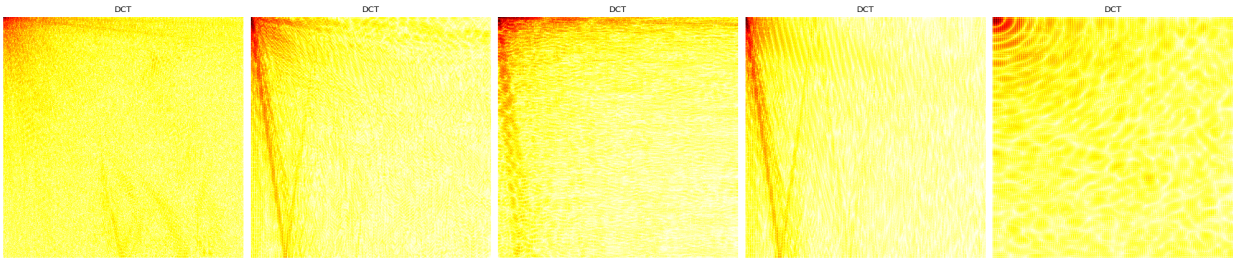

DCT
DCT
DCT
DCT
DCT

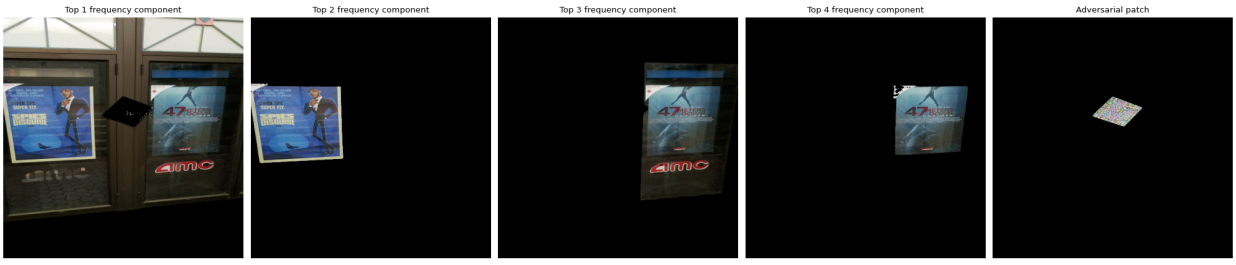

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

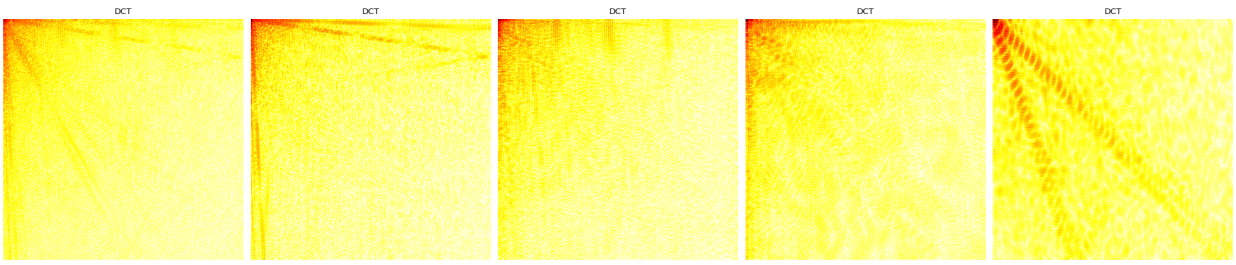

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

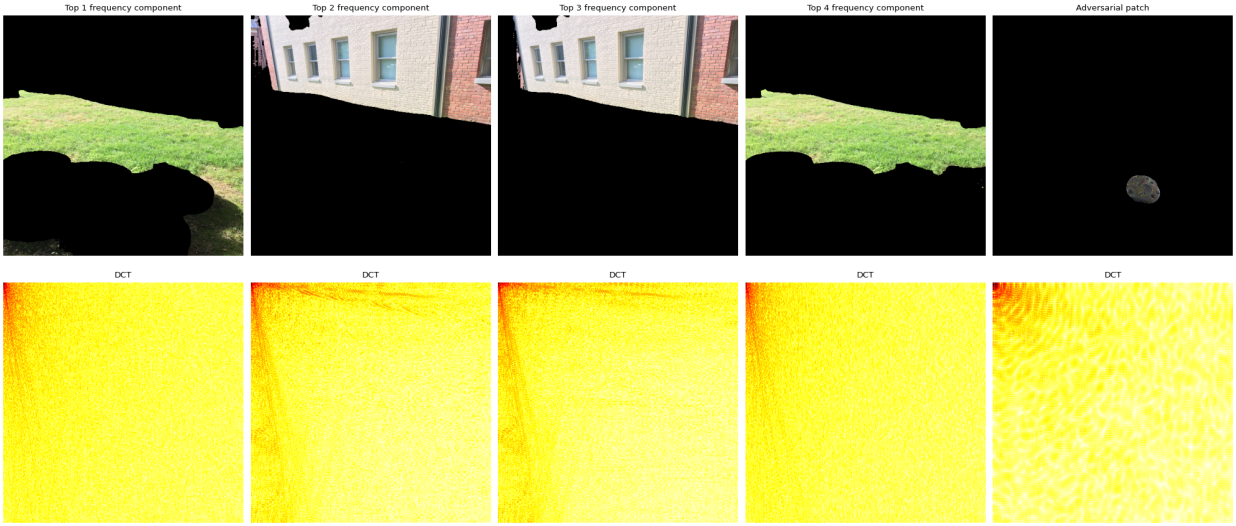

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

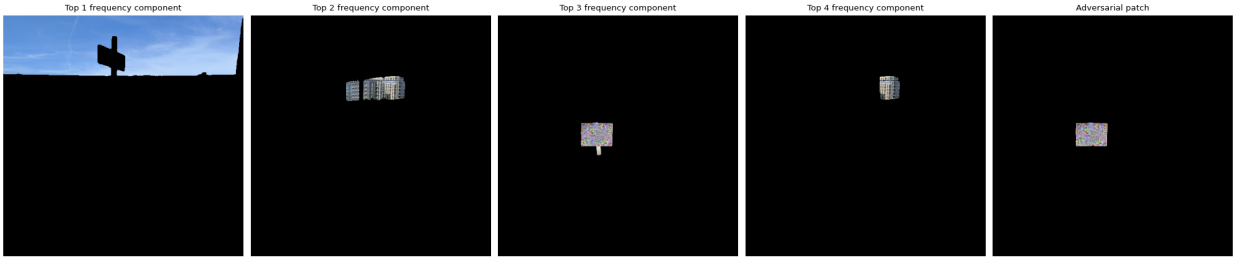

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

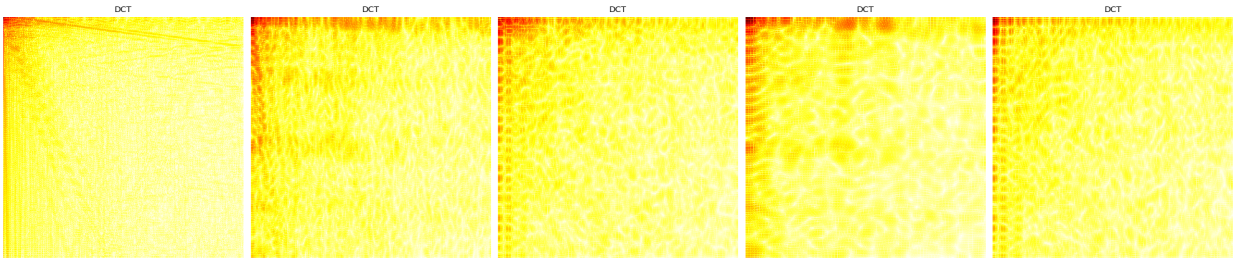

DCT
DCT
DCT
DCT
DCT

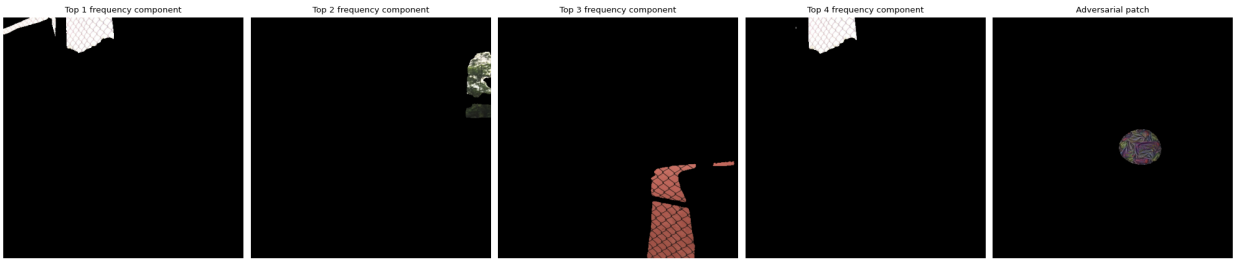

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

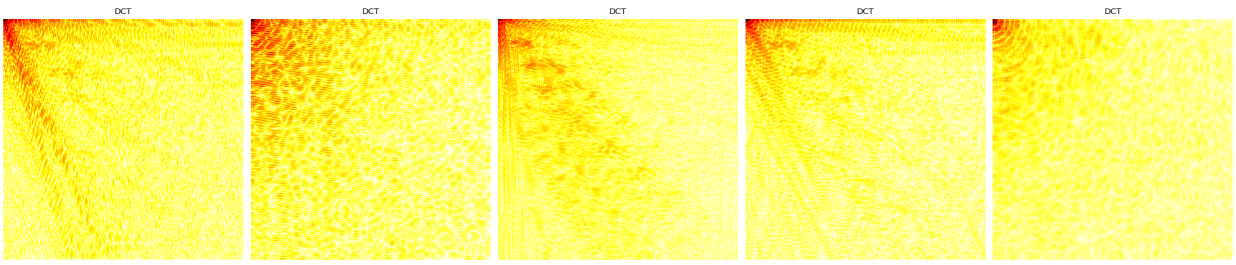

DCT
DCT
DCT
DCT
DCT

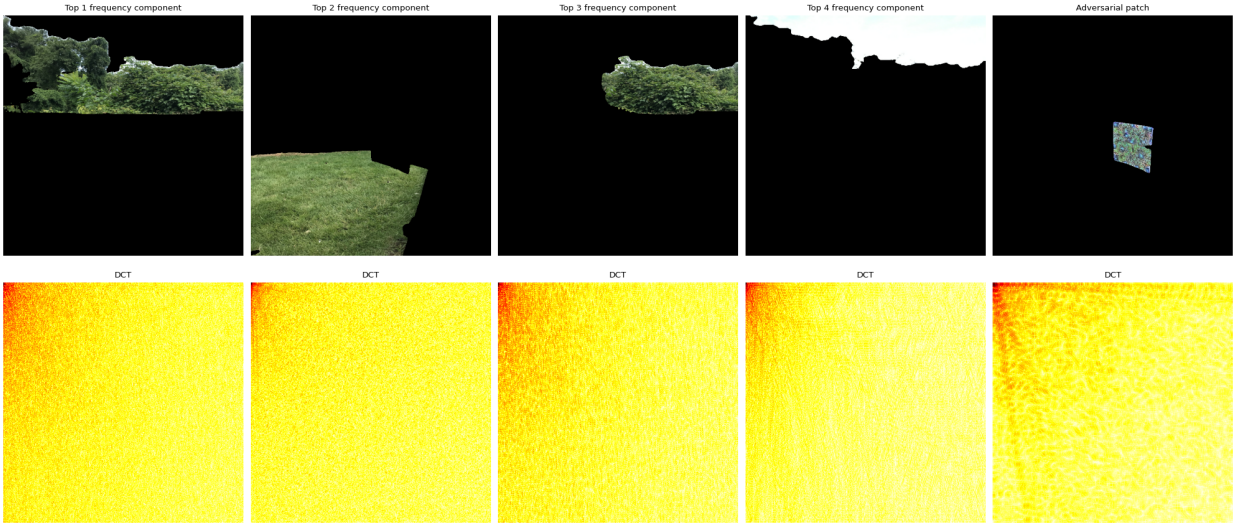

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

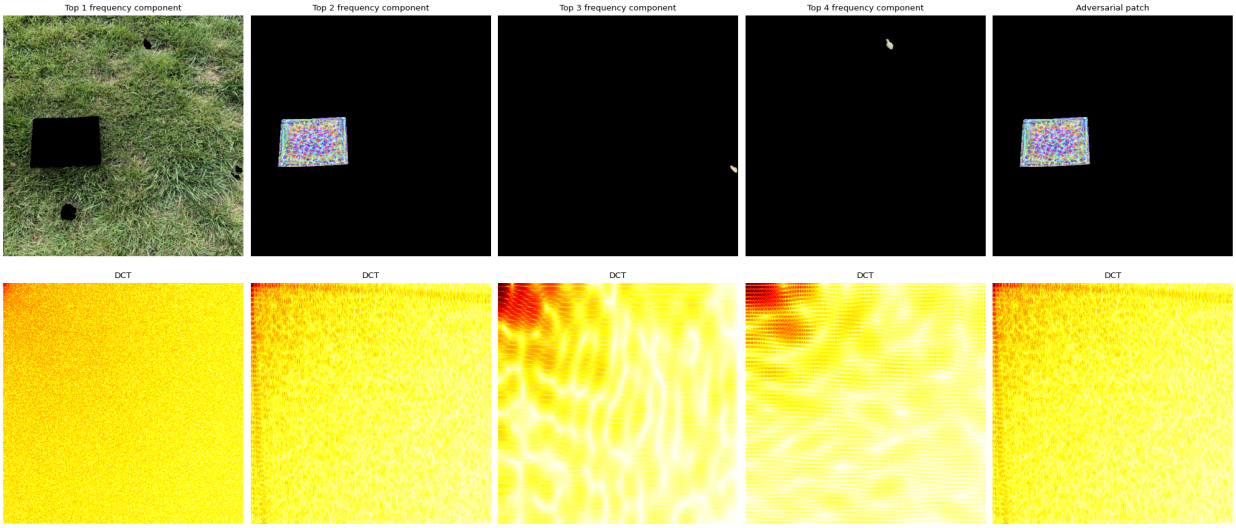

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

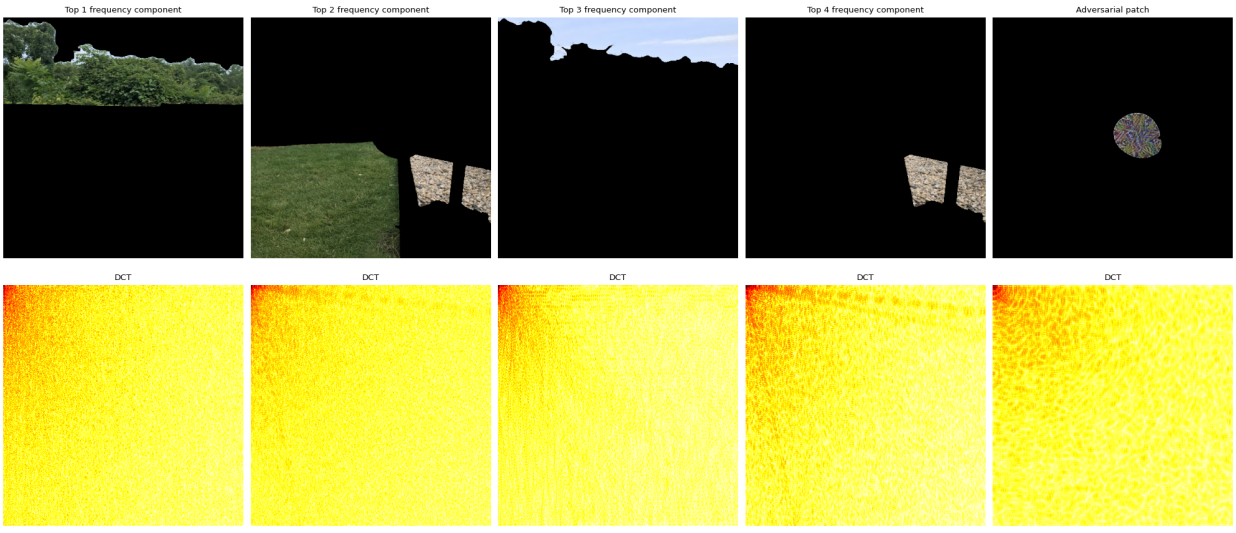

DCT
DCT
DCT
DCT
DCT

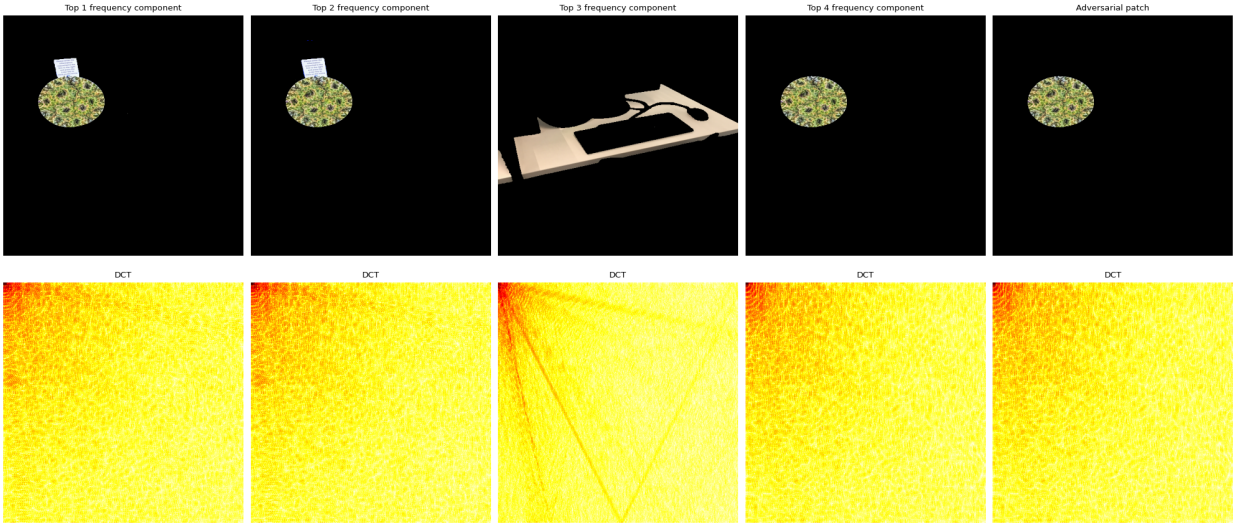

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch

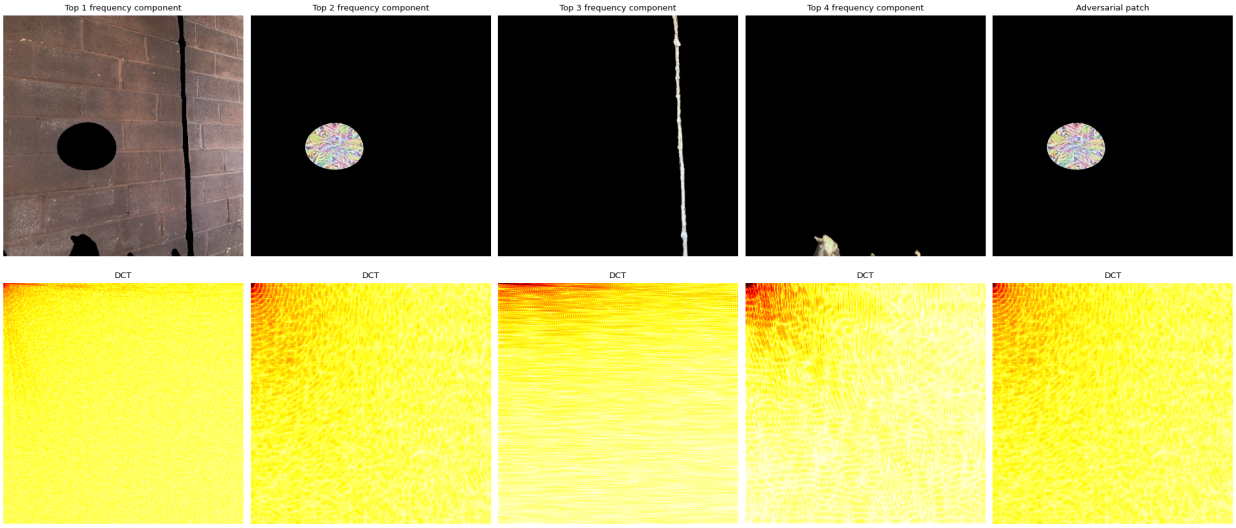

DCT
DCT
DCT
DCT
DCT

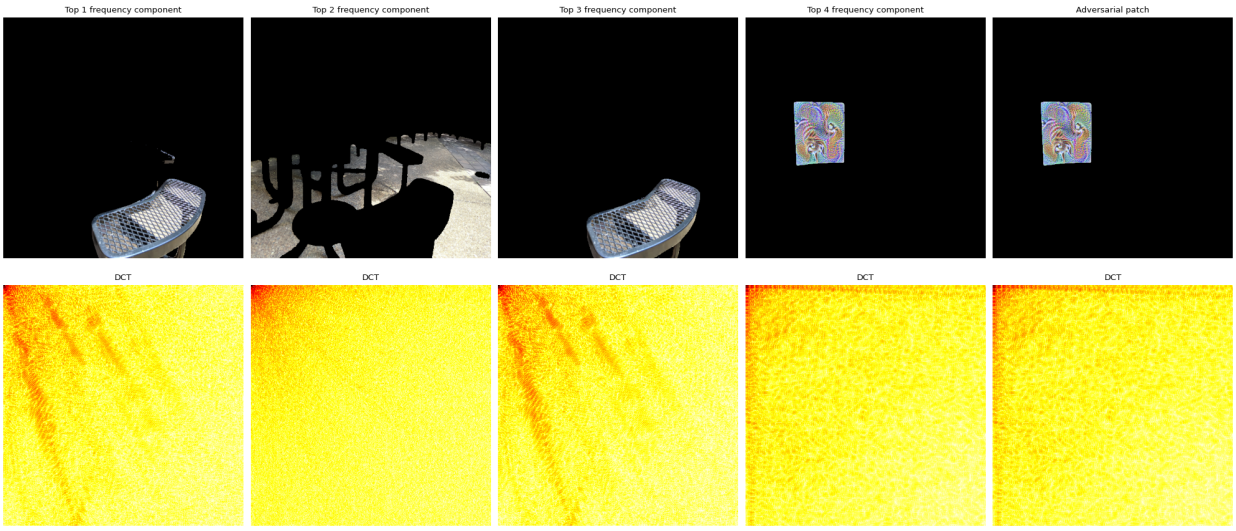

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT


Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT


Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT

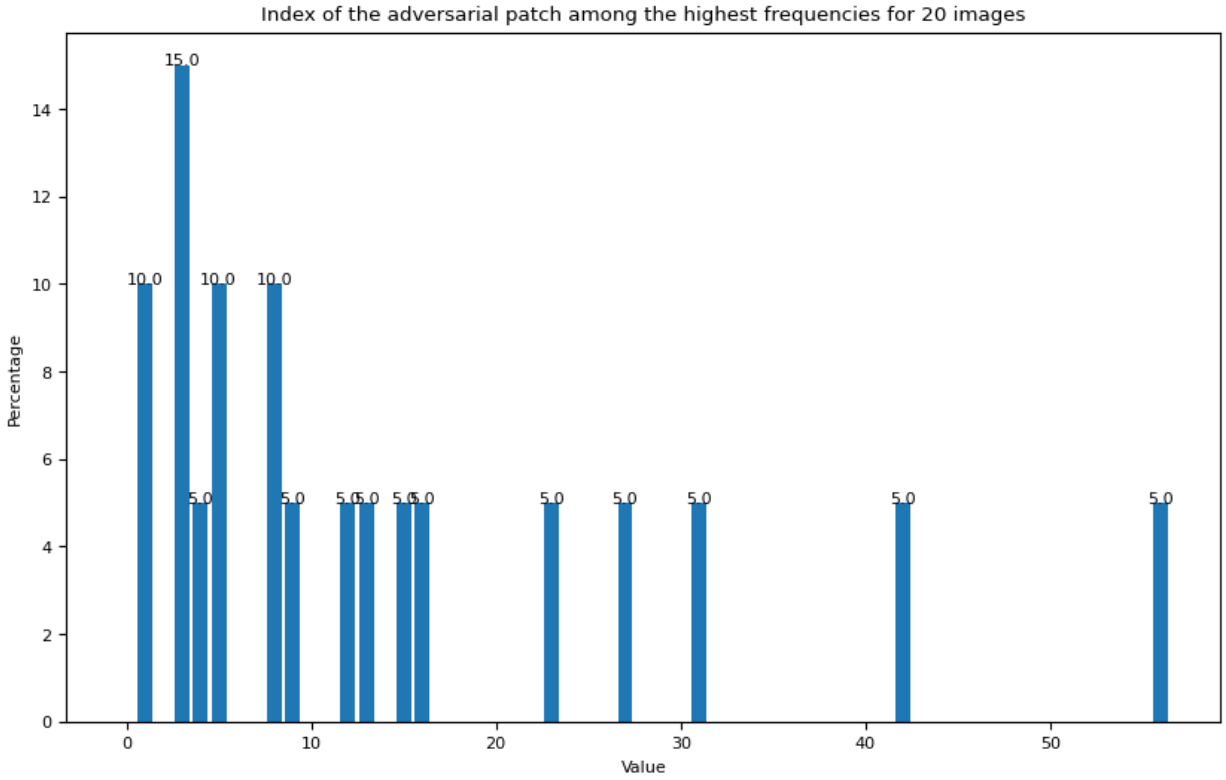

Top 1 frequency component
Top 2 frequency component
Top 3 frequency component
Top 4 frequency component
Adversarial patch
DCT
DCT
DCT
DCT
DCT





In [56]:
```
hist, bins = np.histogram(patch_indices, bins=np.arange(np.max(patch_indices) +

plt.figure(figsize=(10,6))
plt.rc('font', size=8)

percentage = hist / len(patch_indices) * 100

# Plot the percentage-wise histogram
plt.bar(bins[:-1], percentage, align='center', width=0.8)
plt.xlabel('Value')
```

```
plt.ylabel('Percentage')
plt.title('Index of the adversarial patch among the highest frequencies for {}

for i in range(len(percentage)):
    if percentage[i] != 0.0:
        plt.text(bins[i], percentage[i], f'{percentage[i]:.1f}', ha='center')

plt.show()
```



In [ ]: