Homework #3

Deadline: November 18, 2025 @23:59

Submissions: (1) PDF version of this file

(2) .ipynb file (backup/support file only)

Only the PDF will be graded. The .ipynb file is for reference or verification only.

Download data in colab link below:

https://colab.research.google.com/drive/1YyMPZbfaJNlNaB5My1__ZYiVmGUcPBtr?usp=sharing

IMPORTANT! (1) Before submitting the python file, please make sure it can be successfully compiled and correctly in its format name

(2) The scores will be 0 for all students whose source codes/writing are very similar to each other.

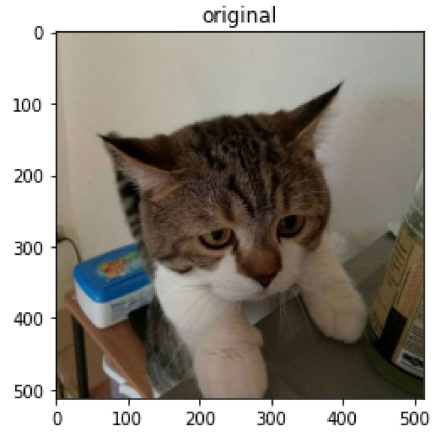
1. (10 points) Image Augmentation (no need to put in COLAB)

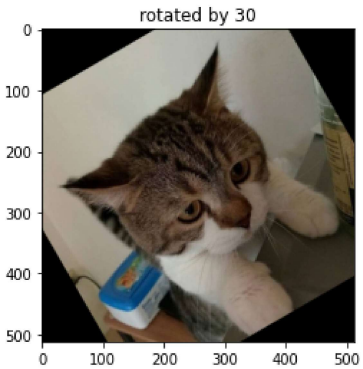
From image classification in object recognition lecture, the self-learning features are capable of learning object patterns using deep neural networks. However, you may require large number of datasets for a model to learn. Study image augmentation, which can be used to expand the size of the training dataset. **Design 5 modified versions of an original image** using knowledge from image processing class, so you can use them for model learning. **Implement 5 different image variations for general object classification task** using the example and the template below.

The example below is a modified version of an original image using rotation which can have variations in terms of orientation angle of an image from 0-360 degrees.

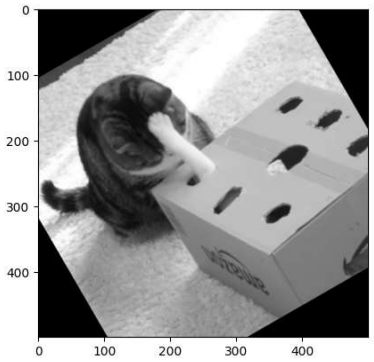
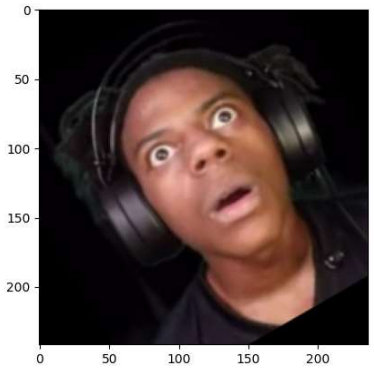
You should use a kitty image and one additional selected image and displayed in the table below. Also once call your python file, it

should show the results of your 5 different modified versions of the original image.

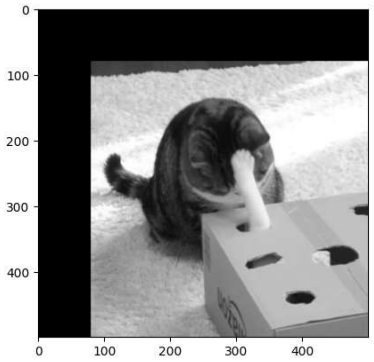
Original image	**Kitty.jpg**	**Your selected image**
		

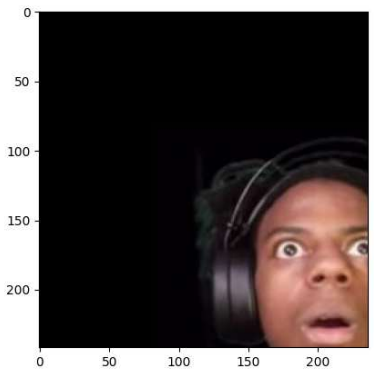
No.	**Modification Techniques & applied images (kitty.jpg and your selected image)**	**Purpose and key parameters**
0	Example: Rotation 	Rotation can imitate the real dataset that the object can be varied in orientations. **Variations**: angles in range of [-30,30]
	Code: // you can write your own designed augmentation techniques	

	`data_transform = transforms.Compose([`
	`transforms.RandomRotation((-30,30)),`
	`transforms.ToTensor(),`
	`])`	

1	 	Rotation to simulate viewpoint/orientation changes Key params variations: angle (degrees of rotation)

	Code: `rotate_image(image,30)`
	`def rotate_image(image, angle):`
	`  rotated_image = imutils.rotate(image, angle)`
	`  return rotated_image`	

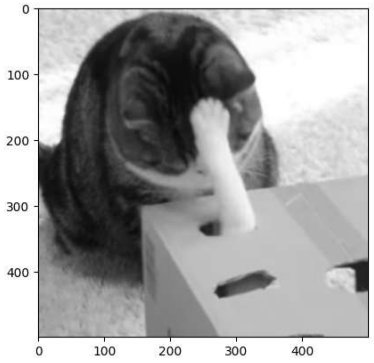
2		Translation to simulate object shift in frame (position change) Key params variations: x (horizontal shift), y (vertical shift), in pixels

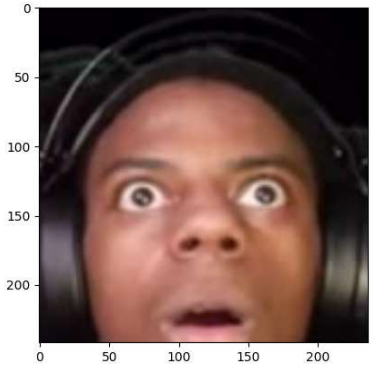


Code: `translate_image(image,80,80)`

```
def translate_image(image, x, y):
  translated_image = imutils.translate(image, x, y)
  return translated_image
```

3



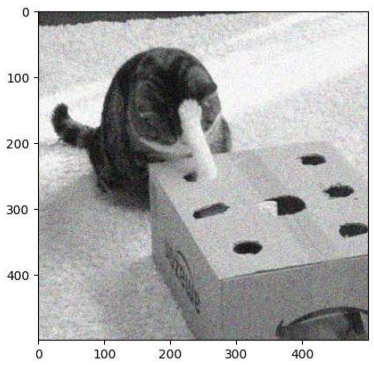


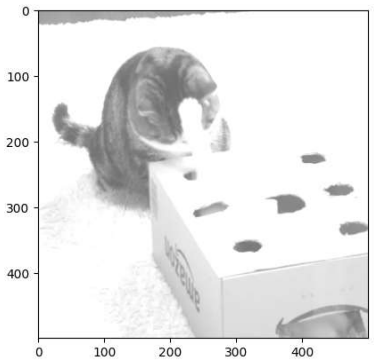
Crop to simulate zoom framing and improve scale robustness

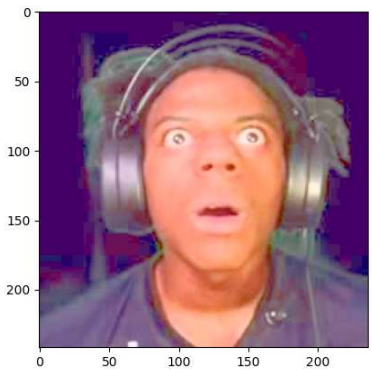
Key params variations: crop_frac : portion of original pixel kept

Code: `random_crop_and_resize(image, 0.6)`

```
def random_crop_and_resize(image, crop_frac = 0.8):
    h, w = image.shape[:2]
    new_h = int(h * crop_frac)
    new_w = int(w * crop_frac)
    top = random.randint(0, h - new_h)
    left = random.randint(0, w - new_w)
    crop = image[top:top + new_h, left:left + new_w]
    resized = cv2.resize(crop, (w, h), interpolation=cv2.INTER_LINEAR)
    return resized
```

<table>
<tr>
<td>4</td>
<td> 
 </td>
<td>Gaussian noise to simulate sensor noise to increase noise robustness
Key params variations: mean (center of noise), sigma (noise intensity)</td>
</tr>
<tr>
<td></td>
<td colspan="2">Code: <code>add_gaussian_noise(image,5.0,30.0)</code>

```
def add_gaussian_noise(image, mean = 0.0, sigma= 10.0):
    if image.dtype != np.float32 and image.dtype != np.float64:
        img = image.astype(np.float32)
    else:
        img = image.copy()
    noise = np.random.normal(mean, sigma, img.shape).astype(np.float32)
    noisy = img + noise
    noisy = np.clip(noisy, 0, 255).astype(np.uint8)
    return noisy
```
</td>
</tr>
<tr>
<td>5</td>
<td>  </td>
<td>Increase brightness to simulate lighting variations (brighter scene)
Key params variations: value (brightness increment in HSV V channel)</td>
</tr>
</table>



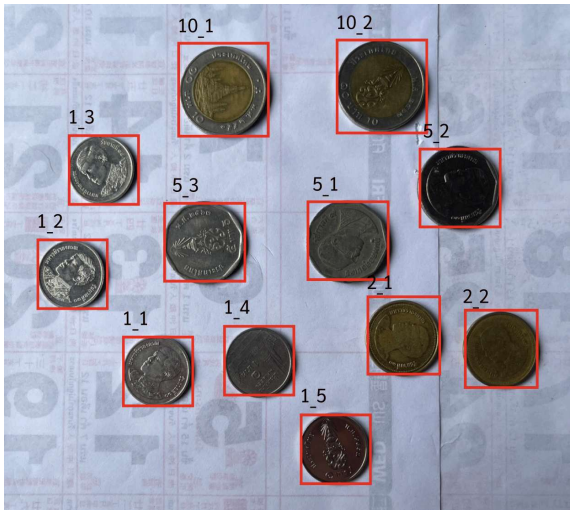
Code: `increase_brightness(image, 100)`

```python
def increase_brightness(image, value):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v, value)
    v = np.clip(v, 0, 255)
    final_hsv = cv2.merge((h, s, v))
    brighthen_img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    return brighthen_img
```

2. Thailand’s coin currency consists of several denominations: 1, 2, 5, and 10 Baht coins. Each coin has a unique size and color that can be used for identification. Specifically, the 1 Baht coin is small and silver in color, the 2 Baht coin is medium-sized and gold, the 5 Baht coin is larger and silver, and the 10 Baht coin is the largest, featuring a bicolor design with a silver rim and a gold center. To assist in automating the coin counting process, you are required to develop an image processing algorithm capable of detecting and counting Thai coins from a given image. The program should be able to identify the denomination of each coin based on its size and color, count the number of coins for each denomination, and calculate both the total number of coins and the total monetary value in Baht. The output should include the number of coins for each denomination, the total number of coins, and the total value in Baht. In addition, a resulting image should be

generated to visually present the detected coins, annotated with bounding circles or boxes and labeled text using functions as the example shown below

Note: your algorithm does not have to be 100% accurate; you should explain your results.

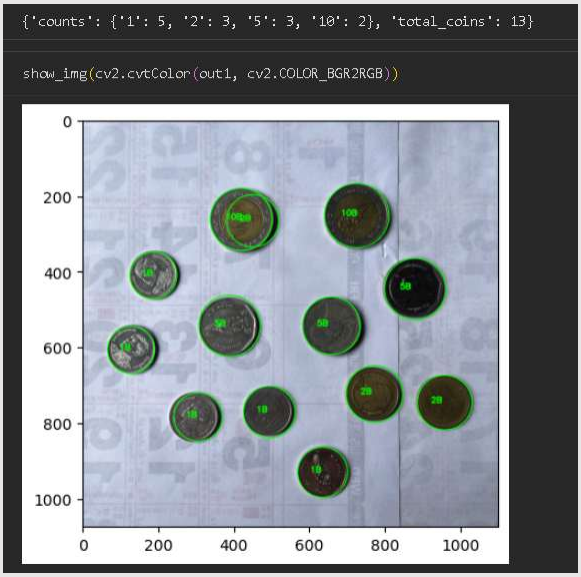
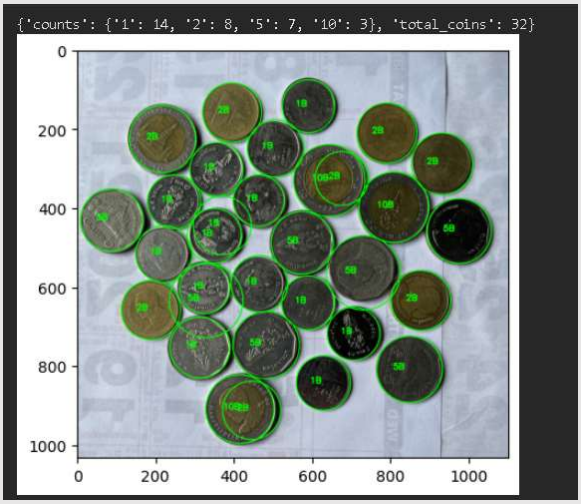
Original image	Your results / number of counted spores
 Coin_1.png	**EXAMPLE**  **Result:** **1 Bath = 5 Coins, 2 Bath = 2 Coins** **5 Bath = 2 Coins, 10 Bath = 2 Coins** **Total 11 Coins, 39 Baths**

2.1) Describe steps of your algorithm

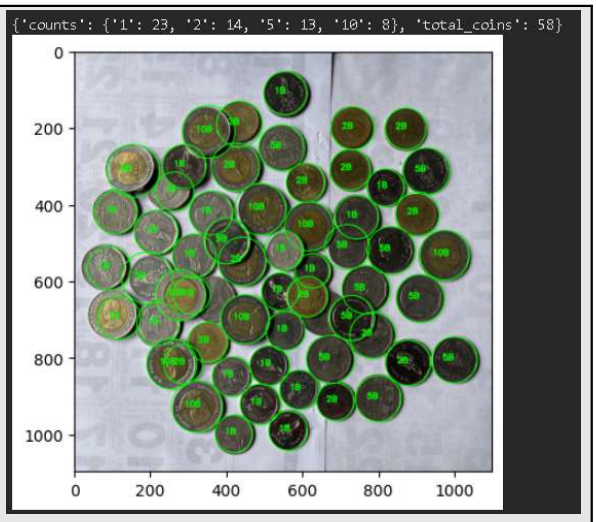
Steps	Description and purposes
1	Remove back ground with gray-scale threshold because I found back ground is much brighter than foreground

2	Mix original foreground with cleaned background because we have to use color for identify 2 & 10 bath coin
3	Use Hough circle to get a circle like shape -> coin
4	Use 2nd biggest coin (avoid outlier) to generalize size, and saturation of circles center to identify color -> get coin type

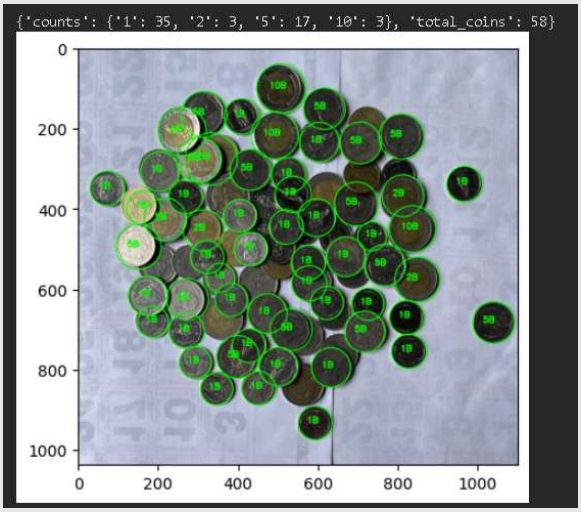
2.2) Results

Original image	**Your results / number of counted spores**
 Coin_1.png	
 Coin_2.png	

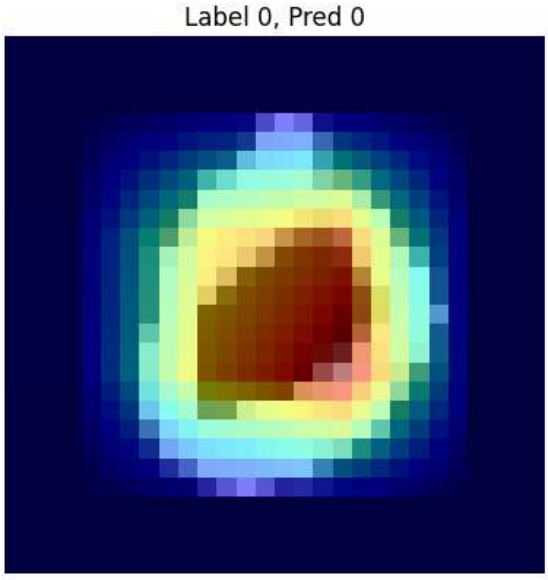
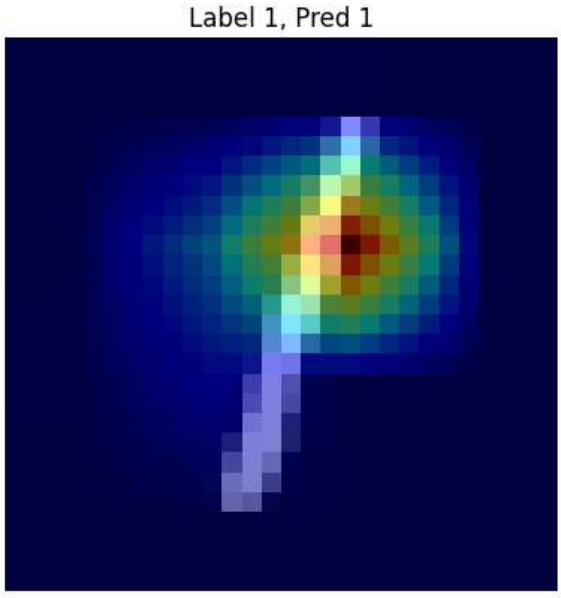
Coin_3.png

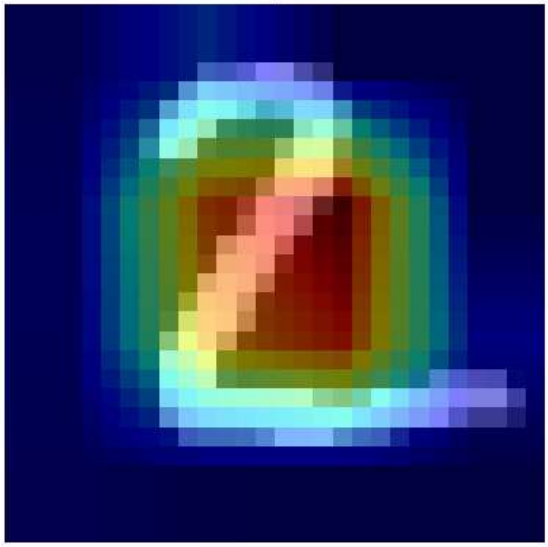
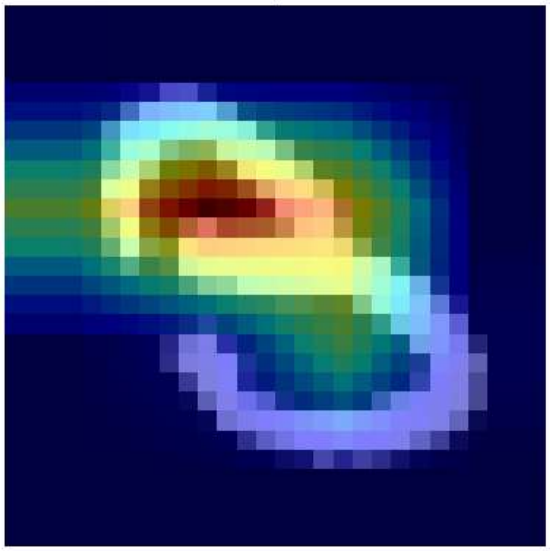
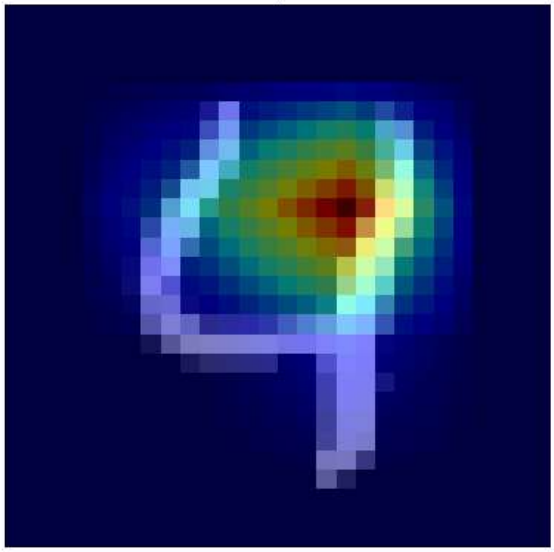


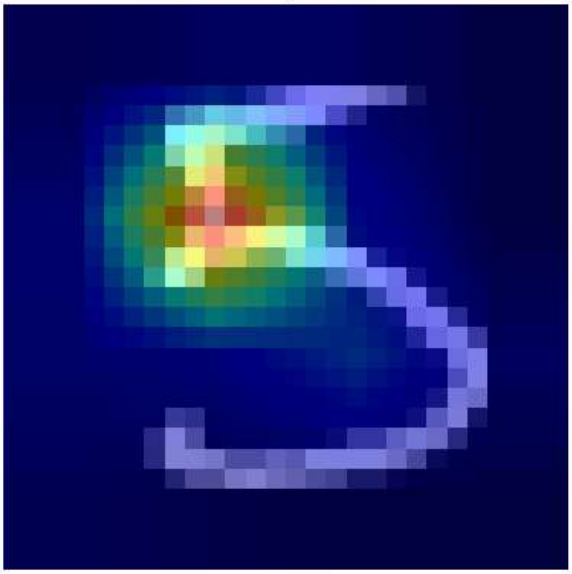
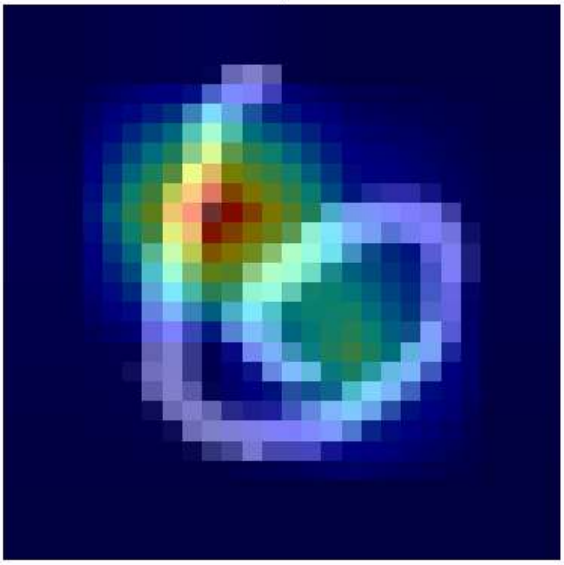
Coin_4.png

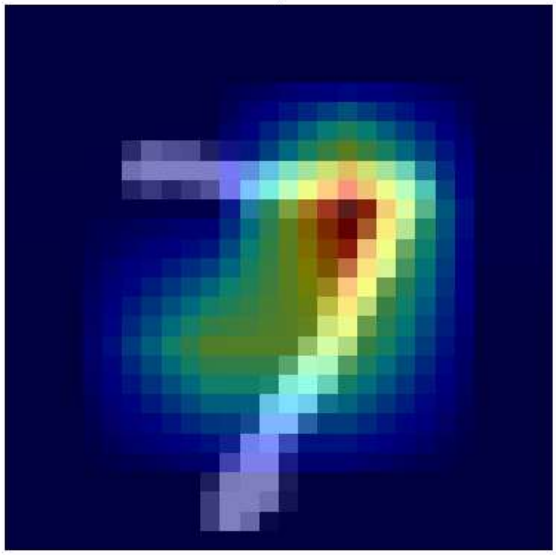
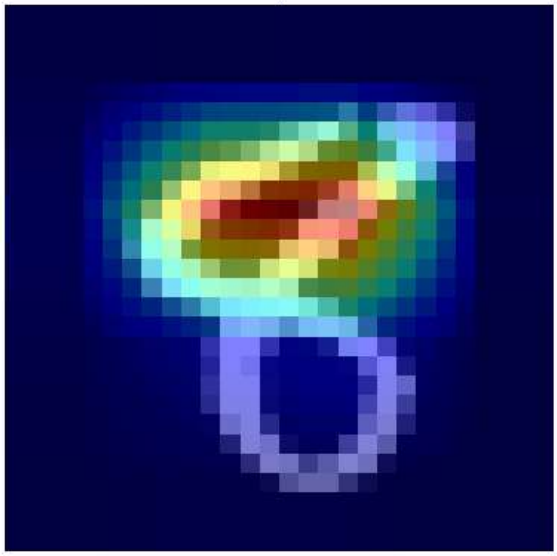
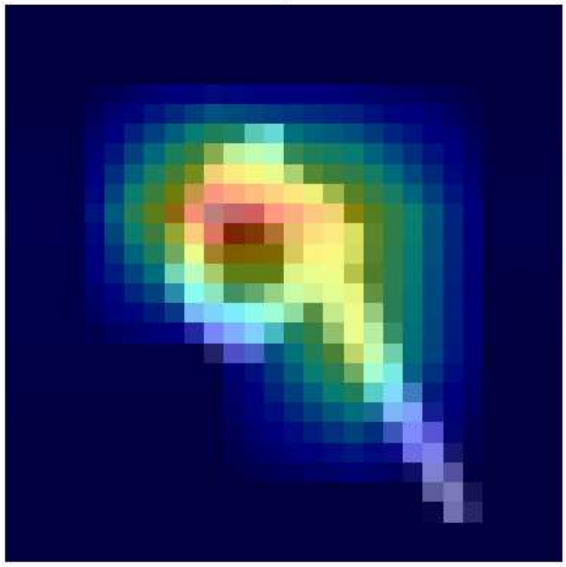


3. In the Colab link above, you have the ViT model code from our class.
3.1) Write code to **visualize** which parts of the image the model focuses on (e.g., attention map or heatmap). You may overlay the attention on the original image. Show at least one example image for each digit from 0 to 9 in the table provided below.

Number	Heatmap / Attention map
0	
1	

2	Label 2, Pred 2 
3	Label 3, Pred 3 
4	Label 4, Pred 4 

5	Label 5, Pred 5 
6	Label 6, Pred 6 

7	Label 7, Pred 7 
8	Label 8, Pred 8 
9	Label 9, Pred 9 

3.2) Write a short explanation in your own words (English or Thai). Describe what your visualization shows, which parts the model attends to, and what you learned.

Pure AI-generated text (e.g., directly copied from ChatGPT or other tools) will not be accepted.

Code: Model added

```python
class MultiHeadSelfAttention(nn.Module):
    def __init__(self, dim, num_heads=4):
        super().__init__()
        self.num_heads = num_heads
        self.head_dim = dim // num_heads
        self.scale = self.head_dim ** -0.5
        self.qkv = nn.Linear(dim, dim * 3)
        self.proj = nn.Linear(dim, dim)
        self.last_attn = None # store attn

    def forward(self, x):
        B, N, D = x.shape
        qkv = self.qkv(x).reshape(B, N, 3, self.num_heads, self.head_dim)
        q, k, v = qkv[:, :, 0], qkv[:, :, 1], qkv[:, :, 2]
        q, k, v = q.transpose(1, 2), k.transpose(1, 2), v.transpose(1, 2)
        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = F.softmax(attn, dim=-1)
        self.last_attn = attn # save attn
        out = (attn @ v).transpose(1, 2).reshape(B, N, D)
        return self.proj(out)
```

```python
def visualize_attention_on_image(img, attn_grid, label, pred, save_path=None):
    # move image to CPU and to (H, W)
    img_np = img.squeeze().detach().cpu().numpy()

    # upsample (GxG) → (HxW) using bilinear
    attn = attn_grid.unsqueeze(0).unsqueeze(0)     # (1,1,G,G)
    H, W = img_np.shape
    attn_up = F.interpolate(attn, size=(H, W), mode="bilinear", align_corners=False)[0, 0]

    # normalize to [0,1]
    attn_up = attn_up.detach().cpu()
    attn_up = attn_up - attn_up.min()
    attn_up = attn_up / (attn_up.max() + 1e-6)

    plt.figure()
    plt.imshow(img_np, cmap="gray")
    plt.imshow(attn_up.numpy(), cmap="jet", alpha=0.5)
    plt.title(f"Label {label}, Pred {pred}")
    plt.axis("off")

    if save_path is not None:
        plt.savefig(save_path, bbox_inches="tight", dpi=150)
    plt.show()
```

```python
with torch.no_grad():
    for imgs, labels in testloader:
        imgs, labels = imgs.to(device), labels.to(device)
        preds = model(imgs)
        # get attention from last transformer block
        attn = model.blocks[-1].attn.last_attn
        B, Hh, N, _ = attn.shape
        # average over heads -> (B, N, N)
        attn_mean = attn.mean(dim=1)
        # CLS token is index 0; take its attention to all tokens
        cls_attn = attn_mean[:, 0, :]  # CLS row
        # drop CLS->CLS itself, keep only patch tokens
        cls_to_patches = cls_attn[:, 1:]
        num_patches = model.num_patches
        grid_size = int(math.sqrt(num_patches))
        cls_to_patches = cls_to_patches.reshape(B, grid_size, grid_size)
        for i in range(B):
            label = labels[i].item()
            pred = preds[i].argmax(dim=0).item()
            # only use correctly classified examples
            if pred == label and not picked[label]:
                attn_grid = cls_to_patches[i]
                img       = imgs[i]
                visualize_attention_on_image(img, attn_grid, label, pred)
                picked[label] = True
                if all(picked.values()):
                    break
        if all(picked.values()):
            break
```

Explanation:

เรา save attention matrix อันสุดท้ายไว้เพื่อดูว่าโมเดลให้ความสนใจยังไง, ดึง cls token เพราะ cls token เป็นตัวแทนของภาพ, เนื่องจากมีการ patch เลยต้องทำการรวมกลับไปด้วยจากนั้นก็ bilinear ให้มีขนาดเท่าภาพก่อน overlay ลงบนภาพแบบ heatmap

Explanation to visualization :
0 ดูตรงกลางว่ากลมไหม ซึ่งน่าจะเช็คไปพร้อมกันว่ามีกี่วง
1 ดูตรงหัวว่ามีการโค้งงอหรือไม่ ไม่มีคือ 1 เพราะเลขอื่นมีโค้งกันหมด
2 ดูทรง S ที่กลับด้าน
3 ดูส่วนข้างบน ค่อนข้าง surprise คิดว่า วงบนที่ไม่ปิดคือ 3 ถ้าปิดคือ 8
4 ดูรูป -| นี้โดยที่ข้างบนไม่มีเส้นแนวนอน
5 ดู |= ส่วนบนซ้าย ค่อนข้าง surprise
6 ค่อนข้างคล้ายกับ 5 แต่มีการเช็ควงกลมข้างล่าง
7 ดูหักมุมขวาบน
8 ดูวงกลมบนว่าปิดไหม คล้าย 3
9วงกลม + เส้นตรงลากลง