In [1]:
```
# This file is a modified version of the original:
# https://www.kaggle.com/code/leonidkulyk/eda-vc-id-volume-layers-animation
```

🌋 Vesuvius Challenge - 📜 Ink Detection - Exploratory Data Analysis

Resurrect an ancient library from the ashes of a volcano

(ಠ_ಠ) Overview

◯ Join the Vesuvius Challenge to resurrect an ancient library from the ashes of a volcano. In this competition you are tasked with detecting ink from 3D X-ray scans and reading the contents. Thousands of scrolls were part of a library located in a Roman villa in Herculaneum, a town next to Pompeii. This villa was buried by the Vesuvius eruption nearly 2000 years ago. Due to the heat of the volcano, the scrolls were carbonized, and are now impossible to open without breaking them. These scrolls were discovered a few hundred years ago and have been waiting to be read using modern techniques..

◯ The competition offers a **grand prize of 150,000 USD to the first team that can read these scrolls from a 3D X-ray scan**. The competition hosts the Ink Detection progress prize, which is about the sub-problem of detecting ink from 3D X-ray scans of fragments of papyrus that became detached from some of the excavated scrolls.

◯ The ink used in the Herculaneum scrolls does not show up readily in X-ray scans, but machine learning models can detect it. The dataset contains 3D X-ray scans of four fragments at 4µm resolution, made using a particle accelerator, as well as infrared photographs of the surface of the fragments showing visible ink.

◯ Hand-labeled binary masks indicating the presence of ink in the photographs are also provided.

Table of contents

- 0. Import all dependencies
- 1. Overview directories
- 2. Overview the 1st volume of the train/ directory
  - 2.1 Consider mask.png image
  - 2.2 Consider ir.png image
  - 2.3 Consider inklabels.png image
  - 2.4 Consider inklabels_rle.csv file
  - 2.5 Consider surface_volumes/ directory
- 3. Overview the 2nd volume of the train/ directory
  - 3.1 Consider mask.png image
  - 3.2 Consider ir.png image
  - 3.3 Consider inklabels.png image
  - 3.4 Consider inklabels_rle.csv file
  - 3.5 Consider surface_volumes/ directory
- 4. Overview the 3rd volume of the train/ directory
  - 4.1 Consider mask.png image
  - 4.2 Consider ir.png image
  - 3.3 Consider inklabels.png image
  - 4.4 Consider inklabels_rle.csv file
  - 4.5 Consider surface_volumes/ directory
- 5. Overview volumes of the test/ directory
  - 5.1 Consider mask.png images

0. Install & Import all dependencies

In [2]:
```
!pip install celluloid -q
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system packag
e manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv (https://pip.pypa.i
o/warnings/venv)
```

In [3]:
```
import os
import gc
import matplotlib.pyplot as plt
from typing import Union, Optional
from PIL import Image
from IPython.display import HTML, display

import numpy as np
import pandas as pd
import plotly.express as px
from tqdm import tqdm
from celluloid import Camera
```

In [4]:
```
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

In [5]:
```
plt.rcParams['figure.dpi'] = 350
plt.style.use('dark_background')
```

In [6]:
```
class CFG:
    layers_count: int = 65
    train_volumes_path_template: str = "/kaggle/input/vesuvius-challenge-ink-detection/train/{}/surface_volume"
    test_volumes_path_template: str = "/kaggle/input/vesuvius-challenge-ink-detection/test/{}/surface_volume"
    tif_template: str = "{:02d}.tif"
```

In [7]:
```
def load_volume(volume_path: str, disable_tqdm: bool = False) -> np.ndarray:
    volume = []
    for i in tqdm(range(CFG.layers_count), disable=disable_tqdm):
        img = Image.open(f"{volume_path}/{CFG.tif_template.format(i)}")
        arr = np.array(img)
        volume.append(arr)

    return volume
```

In [8]:
```python
def animate_volume(
    volume: Union[np.ndarray, str],
    ir_photo: Optional[np.ndarray] = None,
    inklabels: Optional[np.ndarray] = None,
) -> None:

    plt.rcParams['figure.dpi'] = 350
    plt.style.use('dark_background')

    if isinstance(volume, str):
        volume = load_volume(volume, disable_tqdm=True)

    # creating figure subplot
    if ir_photo and inklabels:
        fig, (ax_ir, ax, ax_il) = plt.subplots(1, 3)
    elif ir_photo:
        fig, (ax, ax_ir) = plt.subplots(1, 2)
    elif inklabels:
        fig, (ax, ax_il) = plt.subplots(1, 2)
    else:
        fig, ax = plt.subplots()

    camera = Camera(fig) # define the camera that gets the fig we'll plot
    for i in range(CFG.layers_count):
        ax.axis('off')
        ax.text(
            0.5, 1.08, f"Layer {i+1}/{CFG.layers_count}", fontweight='bold', fontsize=18,
            transform=ax.transAxes, horizontalalignment='center'
        )
        ax.imshow(volume[0], cmap='gray') # plot volume layer

        if ir_photo:
            ax_ir.imshow(ir_photo, cmap='gray')
            ax_ir.axis('off')
            ax_ir.text(
                0.5, -0.13, "Infrared photo", style='italic', fontsize=15,
                transform=ax_ir.transAxes, horizontalalignment='center'
            )

        if inklabels:
            ax_il.imshow(inklabels, cmap='gray')
            ax_il.axis('off')
            ax_il.text(
                0.5, -0.13, "Ink labels", style='italic', fontsize=15,
                transform=ax_il.transAxes, horizontalalignment='center'
            )


        camera.snap() # the camera takes a snapshot of the plot

        del volume[0]
        gc.collect()

    plt.close(fig) # close figure

    animation = camera.animate() # get plt animation

    fix_video_adjust = '<style> video {margin: 0px; padding: 0px; width:100%; height:auto;} </style>'
    display(
        HTML(fix_video_adjust + animation.to_html5_video())
    ) # displaying the animation

    del camera
    del animation
    gc.collect()
```

In [9]:
```python
def get_concat_v(im1: Image, im2: Image) -> Image:
    dst = Image.new('RGB', (im1.width, im1.height + im2.height))
    dst.paste(im1, (0, 0))
    dst.paste(im2, (0, im1.height))
    return dst
```

1. Overview directories

◯ `[train/test]/[fragment_id]/surface_volume/[image_id].tif` slices from the 3d x-ray surface volume. Each file contains a greyscale slice in the z-direction. Each fragment contains **65 slices**. Combined this image stack gives us `width * height * 65` number of voxels per fragment. The image stack is an array of 16-bit grayscale images. You can expect two fragments in the hidden test set, which together are roughly the same as the first training fragment. The sample slices available to download in the test folders are simply copied from training fragment one.

◯ train/ directory has **3** 3d x-ray surface volumes.

In [10]:
```
sorted(os.listdir("/kaggle/input/vesuvius-challenge-ink-detection/train"))
```

Out[10]:
```
['1', '2', '3']
```

◯ test/ directory has **3** 3d x-ray surface volumes. Which together are roughly the same as the **1st training fragment**

In [11]:
```
sorted(os.listdir("/kaggle/input/vesuvius-challenge-ink-detection/test"))
```

Out[11]:
```
['a', 'b']
```

2. Overview the 1st volume of the *train/* directory

◯ Each of train volumes has next files:

- mask.png — a binary mask of which pixels contain data.
- ir.png — the infrared photo on which the binary mask is based.
- inklabels.png — a binary mask of the ink vs no-ink labels.
- inklabels_rle.csv — a run-length-encoded version of the labels, generated using this script. This is the same format as you should make your submission in.

In [12]:
```
os.listdir("/kaggle/input/vesuvius-challenge-ink-detection/train/1")
```

Out[12]:
```
['ir.png', 'inklabels_rle.csv', 'inklabels.png', 'mask.png', 'surface_volume']
```

2.1 Consider *mask.png* image

◯ Let's load the mask

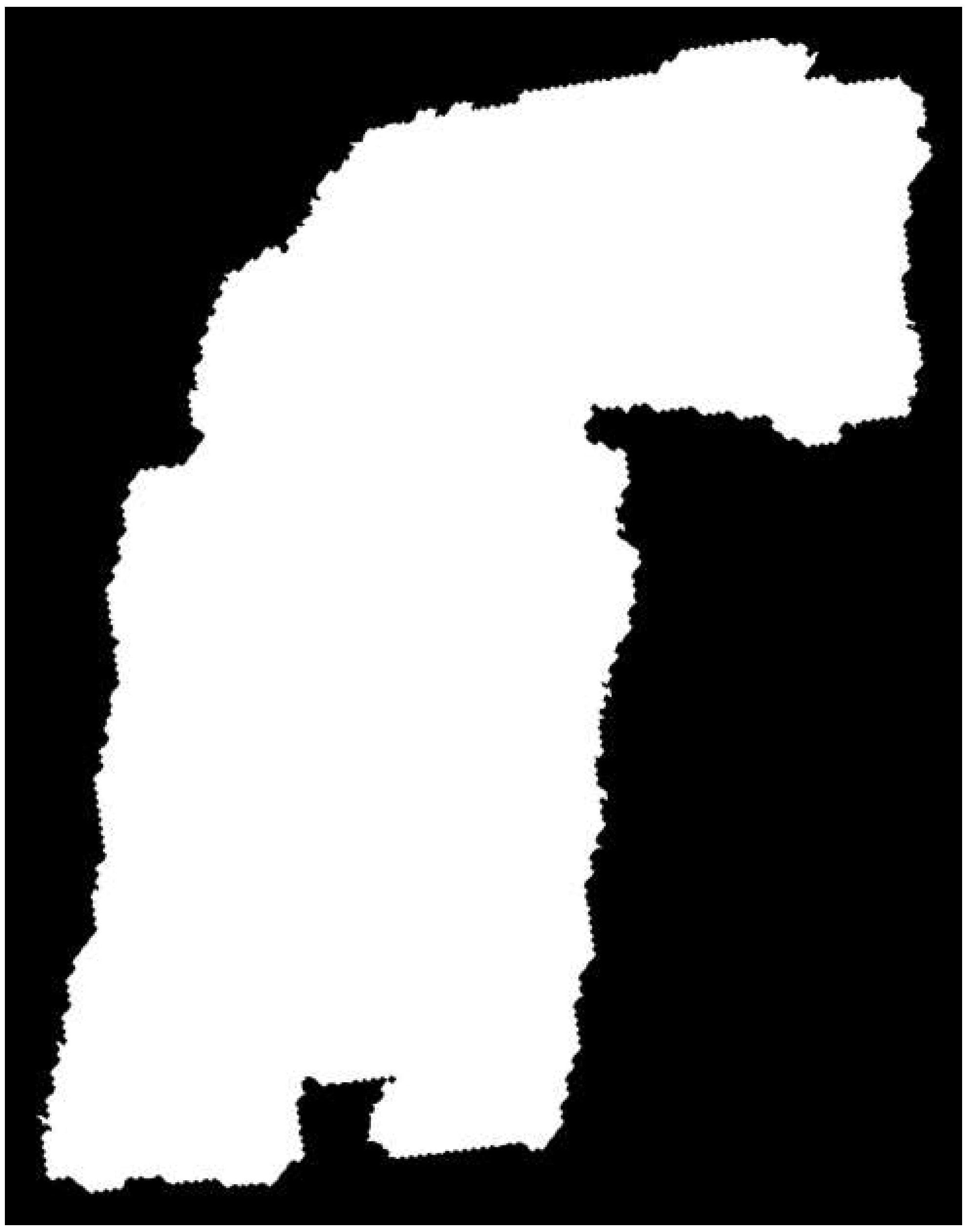
In [13]:
```
train_1_mask = Image.open("/kaggle/input/vesuvius-challenge-ink-detection/train/1/mask.png")
```

In [14]:
```
print(f"Mask image size: {color.BOLD}{color.CYAN}{train_1_mask.size}{color.END}")
```

```
Mask image size: (6330, 8181)
```

◯ And plot it

In [15]:
```
fig, ax = plt.subplots(dpi=160)
ax.axis('off')
ax.imshow(train_1_mask);
```



○ OK there is nothing interesting in the mask, so let's take a look at ir.png image.

2.2 Consider *ir.png* image

○ Load the infrared image

In [16]:
```
train_1_ir = Image.open("/kaggle/input/vesuvius-challenge-ink-detection/train/1/ir.png")
```

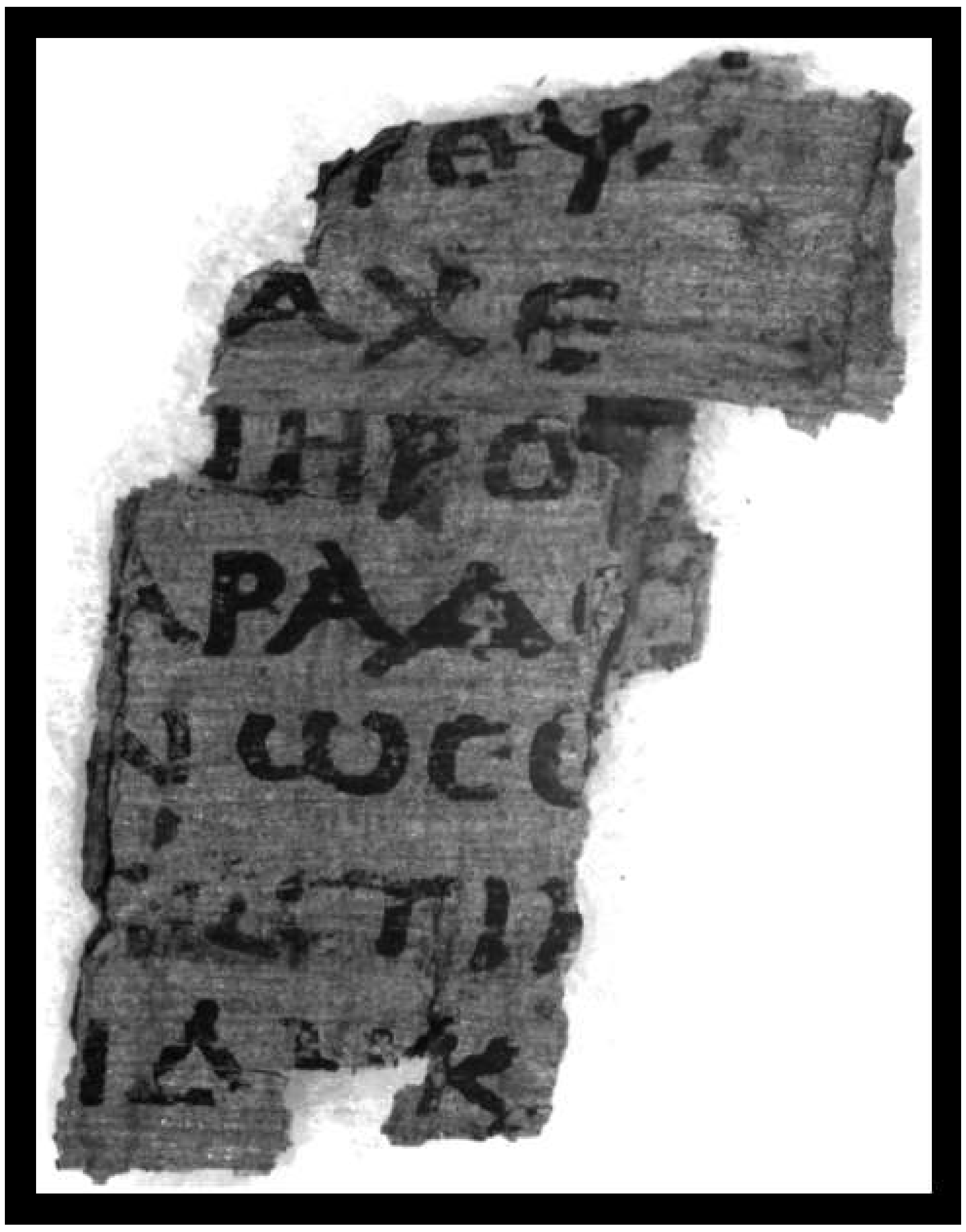
In [17]:
```
print(f"Infrared image size: {color.BOLD}{color.CYAN}{train_1_ir.size}{color.END}")
```

```
Infrared image size: (6330, 8181)
```

○ As you can see in the image there are some fragments of words. The approximate number of letters on the fragment is about 20.

● It is important to note that there is no infrared image in the test dataset, only a mask and 65 volume layers from a 3d x-ray.

In [18]:
```python
fig, ax = plt.subplots(dpi=160)
ax.axis('off')
ax.imshow(train_1_ir, cmap='gray');
```



2.3 Consider *inklabels.png* image

◯ Load inklabels.png image.

In [19]:
```python
train_1_inklabels = Image.open("/kaggle/input/vesuvius-challenge-ink-detection/train/1/inklabels.png")
```

In [20]:
```python
print(f"Inklabels image size: {color.BOLD}{color.CYAN}{train_1_inklabels.size}{color.END}")
```

```
Inklabels image size: (6330, 8181)
```

◯ You are supposed to use it as a prediction reference. Thus, you predict a binary semantic mask from 3D X-ray 65 volume layers and check its intersection with the inklabels mask.

In [21]:
```python
fig, ax = plt.subplots(dpi=160)
ax.axis('off')
ax.imshow(train_1_inklabels, cmap='gray');
```

◯ Let's see the distribution of ink on the first training instance. To calculate the distribution, I consider it optimal to first take a mask, and only after that calculate the distribution, since it is in this way that it is assumed that training and inference will take place.

In [51]:
```python
# conver to np.ndarray
train_1_inklabels_np = np.array(train_1_inklabels)
# np.unique(train_1_inklabels_np) => [0 1]
# np.unique(train_1_mask) => [0 1]
train_1_mask_np = np.array(train_1_mask, dtype=np.bool8)
# np.unique(train_1_mask_np) => [False  True]

# reassign new value for empty space
train_1_inklabels_np[np.logical_not(train_1_mask_np)] = 2

# get counts for each label
# return_counts => if True, also return the number of times each unique item appears in array.
train_1_pix_counts = np.unique(
    train_1_inklabels_np,
    return_counts=True
)[1]

# np.unique(train_1_inklabels_np, return_counts=True) =>
# (array([0, 1, 2], dtype=uint8), array([23803478,  5339362, 22642890]))
```

```
(array([0, 1, 2], dtype=uint8), array([23803478,  5339362, 22642890]))
```

In [ ]:

◯ On visualization you will see next 3 labels:

- No ink – is inside the mask, but is not ink.
- Ink – is inside the mask and is ink.
- Empty space – is outside the mask.

In [23]:
```python
fig = px.bar(
    x=["No ink", "Ink", "Empty space"], y=train_1_pix_counts,
    color_discrete_sequence=['darkgoldenrod']
)

fig.update_layout(
    xaxis_title="Label", yaxis_title="Pixel count",
    title={
        'text': "Distribution of Ink on the 1st train instance",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

2.4 Consider *inklabels_rle.csv* file

◯ There is nothing interesting in this file. It's just a run-length-encoded version of the inklabels.png image, generated using **this script** (https://gist.github.com/janpaul123/ca3477c1db6de4346affca37e0e3d5b0).

◯ It's important to note that this is in the same format as you should make your **submission in** (https://www.kaggle.com/competitions/vesuvius-challenge-ink-detection/overview/evaluation).

In [24]:
```python
pd.read_csv("/kaggle/input/vesuvius-challenge-ink-detection/train/1/inklabels_rle.csv")
```

Out[24]:

	Id	Predicted
0	1	606211 19 612538 26 618867 39 625196 44 631525...

2.5 Consider *surface_volumes/* directory

◯ As mentioned above these are slices from the 3d x-ray surface volume. Each file contains a **greyscale slice in the z-direction**. Each fragment contains **65 slices**. Combined this image stack gives us 65 * height * width number of voxels per fragment.

In [25]:
```python
first_train_volume = CFG.train_volumes_path_template.format(1)
```

In [26]:
```python
train_volume_1 = load_volume(first_train_volume)
```

```
100%|██████████| 65/65 [01:43<00:00,  1.59s/it]
```

In [27]:
```
print(f"1st train volume size: {color.BOLD}{color.CYAN}{train_volume_1[0].shape}{color.END}")
```

```
1st train volume size: (8181, 6330)
```

◯ The whole volume is huge, let's see exactly how.

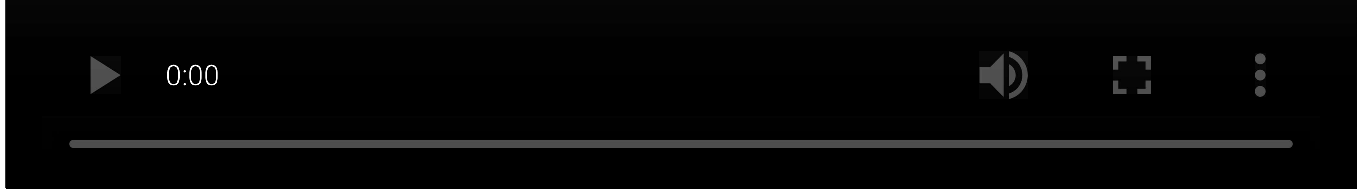
In [28]:
```
bytes_size = CFG.layers_count * train_volume_1[0].size * train_volume_1[0].itemsize
print(f"Memory size of 1st train volume in bytes: {color.BOLD}{color.PURPLE}{bytes_size}{color.END} B")
print(f"Memory size of 1st train volume in gigabytes: {color.BOLD}{color.PURPLE}{bytes_size / 1024**3}{color.END} GB")
```

```
Memory size of 1st train volume in bytes: 6732144900 B
Memory size of 1st train volume in gigabytes: 6.269798520952463 GB
```

◯ For a better understanding of how the layers relate to each other, I made an **animation with the transition from one layer to another**. And so for all 65 layers.

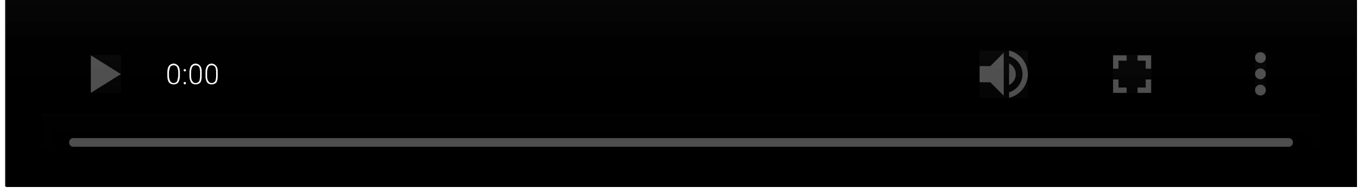
◯ As noted in the description of this competition, it is visually almost impossible to notice any ink on these images. But together they form a coherent picture, which, I hope, can be picked up very easily by a neural network.

In [29]:
```
# animate_volume(train_volume_1)
```



- Let's animate volume right next to its infrared image. So you can look at some features on the fly!

In [30]:
```
# animate_volume(first_train_volume, ir_photo=train_1_ir)
```



◯ And let's animate volume next to its ink labels mask.

```
In [31]: # animate_volume(first_train_volume, inklabels=train_1_inklabels)
```

```
In [32]: # animate_volume(first_train_volume, train_1_ir, train_1_inklabels)
```

5. Overview volumes of the *test/* directory

5.1 Consider *mask.png* images

◯ Let's load masks

```
In [ ]: test_a_mask = Image.open("/kaggle/input/vesuvius-challenge-ink-detection/test/a/mask.png")
        test_b_mask = Image.open("/kaggle/input/vesuvius-challenge-ink-detection/test/b/mask.png")
```

```
In [ ]: print(f"Mask a image size: {color.BOLD}{color.CYAN}{test_a_mask.size}{color.END}")
        print(f"Mask b image size: {color.BOLD}{color.CYAN}{test_b_mask.size}{color.END}")
```

◯ And plot them

```
In [ ]: fig, ax = plt.subplots(dpi=160)
        ax.axis('off')
        ax.imshow(test_a_mask);
```

```
In [ ]: fig, ax = plt.subplots(dpi=160)
        ax.axis('off')
        ax.imshow(test_b_mask);
```

◯ Let's combine them and you'll see that it's just divided 1st train volume.

```
In [ ]: concatenated_test_mask = get_concat_v(test_a_mask, test_b_mask)
```

```
In [ ]: fig, axarr = plt.subplots(2, dpi=160)

        axarr[0].imshow(concatenated_test_mask)
        axarr[0].axis('off')
        axarr[0].set_title("Concatenated test mask")

        axarr[1].imshow(train_1_mask)
        axarr[1].set_title("1st train volume mask")
        axarr[1].axis('off');
```

```
In [ ]: # ( ˘̩̩̩ᴗ˘̩̩̩) WORK STILL IN PROGRESS
```

༼ つ ◕_◕ ༽つ Thank You!

📩 Thank you for taking the time to read through my notebook. I hope you found it interesting and informative. If you have any feedback or suggestions for improvement, please don't hesitate to let me know in the comments.

🚀 If you liked this notebook, please consider upvoting it so that others can discover it too. Your support means a lot to me, and it helps to motivate me to create more content in the future.

❤️ Once again, thank you for your support, and I hope to see you again soon!