Big Data & Self - ‘Faces of the War’

Links:

1. Video: https://ual.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=18e5ac11-e8d5-424f-8780-b33a00ce4529
2. Filtered Dataset: https://artslondon-my.sharepoint.com/:f:/g/personal/a_artemenko0820231_arts_ac_uk/EreH6LweF
3. GitHub project: https://github.com/AndriiArtemenko3/Faces-of-the-War

Introduction

My project is an exploration of how my home country, Ukraine, is affected by the ongoing Russian Invasion to Ukraine (2022-now), and aims to document the reality of the war from the perspective of regular people. In my dataset, I aimed to collect the publicly available data, that shows destructions and horrors of the war. The project has a personal significance for me, as I was born in Ukraine, Kyiv, and was living there for 16 years, until we moved with my parents to Europe in 2018. I still have a large part of my family living in Ukraine, who are in constant danger of being injured or killed. Some members of my family survived occupation in 2022, and still suffer from the PTSD from it. I have to admit that choosing this topic for a project is a big challenge for me, as throughout my education in London, I was avoiding the topic of the war, to preserve my mental health stability. However, due to an increased missile and drone attacks on the residential areas in 2025, and increased casualties among the civilians this year, I feel like the least I can do from my position is to spread awareness about the attacks and reality of leaving in a war-torn country, as unfortunately, it doesn’t get enough coverage in the Western press anymore, and many people are unaware.

For my project I used Reddit’s API to scrape through the major subreddits that cover the war in Ukraine. I also used MatPlotLib python library to make an artwork-collage from the scraped dataset.

Disclaimer - Sensitive Content

This project contains graphic and potentially distressing imagery.
The dataset includes photographs and visual material that document confirmed and alleged war crimes committed by the Russian Federation during its ongoing illegal invasion of Ukraine.

These materials may depict scenes of violence, injury, and loss of life. They are included solely for the purpose of accurate documentation, research integrity, and historical preservation.
I have chosen to retain this sensitive content out of respect for the victims and to ensure the truth of these events is preserved without censorship.

Viewers are advised to approach this material with caution.
The inclusion of these images does not aim to sensationalize violence, but rather to provide an unaltered record of human rights violations for academic and historical examination.

Ethical Statement

This project utilises a dataset of publicly available, scraped images that may contain graphic or sensitive content related to documented war crimes in Ukraine. The imagery was processed and incorporated into the final artwork with careful ethical consideration.

All identifiable personal information has been anonymised, and no individual subjects are named or singled out. The use of these images is solely for academic, historical, and artistic purposes, in line with the project brief.

The visual representation in the artwork has been designed to maintain respect for victims and affected communities, avoiding sensationalism or exploitation of traumatic events. The aim is to preserve the historical truth, raise awareness, and engage the viewer in a reflective, informed manner.

Research - Source of Data

I have considered a couple sources for the data for this project. I started by looking through large non-profit organisations like the UN, UkraineWarCrimes, Bellingcat, OSYNT, etc. What I noticed with most of them, is that they either don’t document all war crimes (for example, Bellingcat’s latest post regarding the war in

Ukraine is from 2024, Amnesty International has outdated information as well), or they only collect the data from citizens to pull them into the private archives, that will then be sent to the international court (mostly true for Ukraine government's websites like https://warcrimes.gov.ua/en , where you can submit a report, but no databases are available to the users like me).

The second step was to look at news outlets, like BBC or Guardian, Reuters, Axios, Kyiv Independent, etc.. While they do provide more information, I would say that they are more focused on the political events and broad overviews rather than going through each case of war crime in detail. Second consideration, is that they usually have much stricter policy for data scraping than other resources.

The third step was to look at social media, especially thread-based ones like Twitter/X. While I was able to find Twitter accounts that document Russia's war crime daily (Notably - Babel.ua: Ukraine at War), the restrictions of the free API allow me to only scrape the latest data, for the past 7 days, which will not be sufficient for my project. The 'pro' version for scraping X/Twitter is way out of my budget and besides, I don't want to give any money to this company.

Then I had to choose between the Telegram and Reddit. I decided to go with Reddit mainly because it is more 'reputable' in my opinion and I don't fully trust Telegram. Besides, from the research I have done on the Telegram API, it appears that scraping some channels without consent can lead to legal consequences. Reddit's sub-channels are public be default, and another pro-point is that Reddit's API was around much longer compared to Telegram's API, which means there are more learning resources for it, and its generally safer. All things considered, I decided to go with Reddit for my project.

Reddit's API set-up & testing

I started with the Reddit's API by writing a simple script to test if I can access the most basic information such as title, score & url of the posts in a specific sub-reddit:

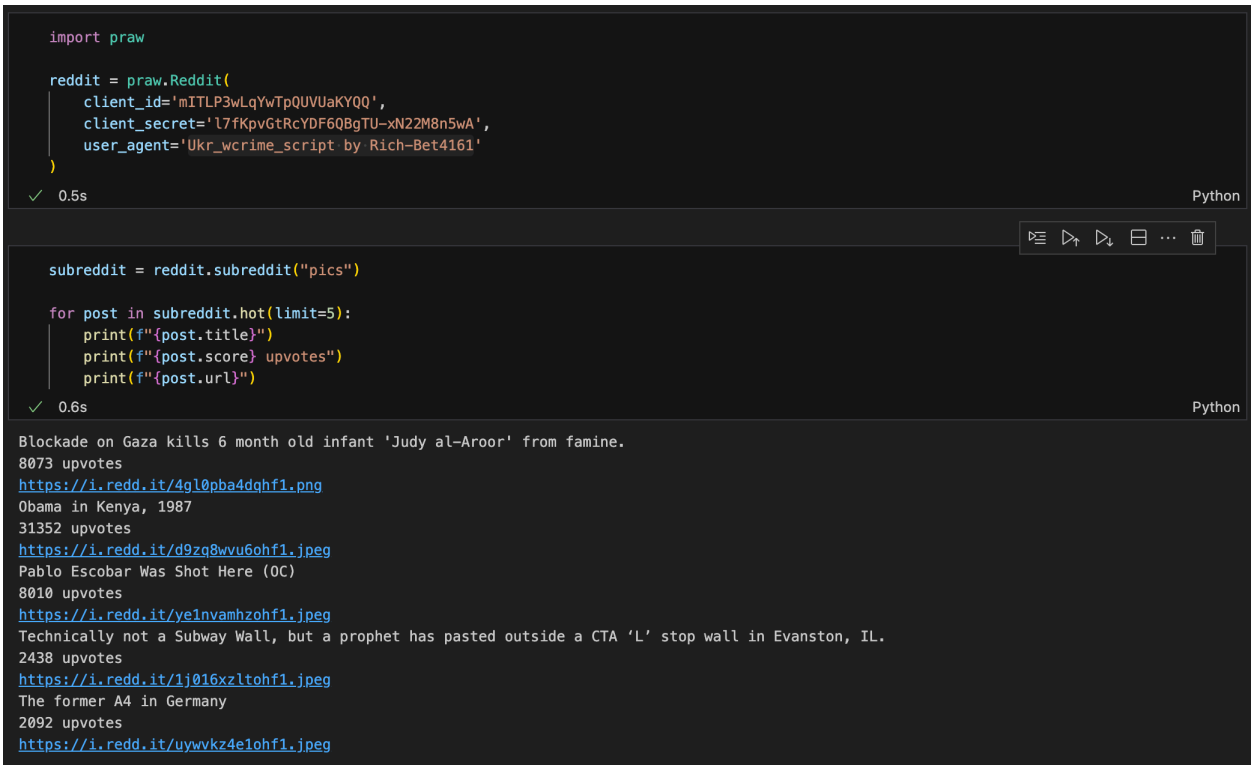


Fig. 1 Test script for Reddit API

But for my project I actually needed a script that would:
a) sort the posts by specific keywords
b) sort the posts by specific parameters e.g. min upvotes

c) save the images from the posts that match my criteria to a local folder

The first two are relatively easy to implement with the reddit API, but saving images locally was not something we covered in class, so I had to search through the open source projects on GitHub to find an example of a reddit scraper with the feature of saving images locally, to get insights into this method. I found a really cool project that provides a good foundation for what I had in mind. (https://www.youtube.com/watch?v=sEIv8UcR3Go) (https://github.com/ClarityCoders/RedditImageScraper).

I had to go through the process of installing the dependencies for this project to run it correctly (requirements.txt file). I also used a virtual environments for this project to avoid any conflicts with the versions of dependencies of other projects I might have on my local computer, and also as a standard recommended practice that we covered in class.

The sub_list.csv file contains names of the subreddits that I was scraping though. The SubDownload.py is the main file where I have my python scraper script.

The main variables I was manipulating with were post search amount, type of search (e.g. new, hot, top, best etc.), keywords, number of upvotes per post.

```
10
11  POST_SEARCH_AMOUNT = 900 # number of posts to search through
12  QUERY = [ # my keywords
13      "missile OR strike OR crime",
14      "military OR kids OR death",
15      "injured OR explosion OR dead",
16      "drone OR attack OR residential",
17      "terrorist OR terrorists OR Kyiv"
18  ]
19  IMAGE_RESIZE_DIMS = (224, 224)
```

Fig 2. Post Search Limit and Keywords

```
            if post.id not in seen_ids:
                seen_ids.add(post.id)
                results.append(post)

    try:
        add_unique(subreddit.search(query, sort="top", time_filter="year", limit=limit))
    except Exception as e:
        print(f"Search failed in {subreddit.display_name}: {e}")
```

Fig 3. Subreddit.search() method with key parameters

Reddit Scraper - Methods and Scraped Results

I first outlined what subreddits I will be scraping through to find the images I am looking for (war crimes, reports of civilian infrastructure being attacked). I quickly found that there are only 2 main subreddits that can provide accurate results → r/Ukraine and r/UkraineVideoReport. Other subreddits related to Ukraine either had different content (not war related), or were too small to actually provide good results in my opinion (they had sub 5000 followers, which is not a lot for reddit). Because I was limited by a certain amount of API calls/post requests, I usually only included one subreddit and then separately run my code for another subreddit, to scrape though as many images as I could in both of them.

Second consideration is the amounts of posts I searched through, and from the information I found online (consulted ChatGPT on this matter and double checked

via google search), the upper limit for it is about 1000 posts per request. So for my **Post_search_amount** variable I didn't go beyond 900, because I tried to run it on 1000 one time and got this message:

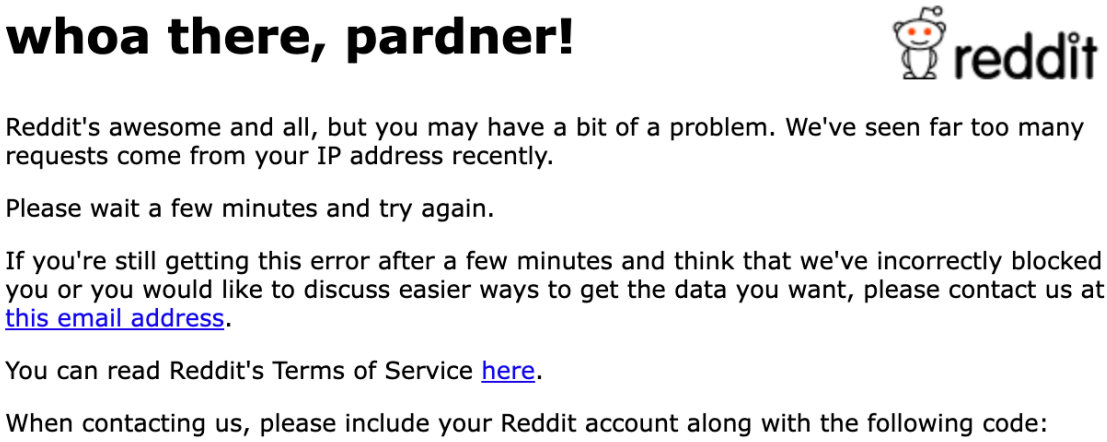


Fig. 4 Warning Message

For the keywords I did a couple things - I included the keywords that I think are reasonable and can return the results that I am looking for (so I was guessing at first - going with words like 'civilians' or 'explosion'). Then I actually opened those subreddits manually, looked through the posts that show the images that I would want to include in my dataset, and observed what keywords people use for the description/titles. Then I would find some common keywords that actually make sense and are used for at least a couple posts, and I would add them to my keyword list, and run the script again. I did it at least a couple times, changing the keywords to get more variety in my results. The limitation of this method is that some posts either don't have a title or description, or it is so unique that it can't be replicated for a mass search through the subreddit.

For extra variety I also modified my **QUERY** variable after a few iterations. I started by writing keywords in a single string, separated by the spaced in between (e.g. **QUERY = 'missile attack children civilians'**). This method uses AND operator, and would only return posts that have all 4 of these keywords in the title. I then added OR operator in between my keywords to make the search broader, as with the OR operator the results will return posts if they contain at least one of those 4 keywords in my example. The results I got were much broader and maybe less precise but it definitely made the search more effective in scraping through more

images.
Then I also expanded my QUERY variable in 5 string versus one, to get even more results, as the code has to loop through each string separately, which provides higher output and applied post search limit (900) to each string separately.

Then I applied some restrictions to my search parameters, to exclude posts that are clearly spam / not relevant. I added a condition **if submission.score < x** to the **for submission in results** loop, to sort the posts via the amount of upvotes. I found that the number around 25-50 is best for hot/top results, but for new results its best to go with the number around 10, as otherwise the results will be too limited.

I also added some rate limits to ensure I am not over reaching the reddit's API restriction on requests per minute. I run a couple iterations and determined the optimal ranges of delay, however I do think that increasing them is the best practice. I just didn't want to waste more hours on scraping, as it was still quite time consuming and for some runs I was waiting 4+ hours to finish.

```
for query in QUERY:
    print(f"Running query batch: '{query}'")
    query_results = fetch_all_posts(subreddit, query, POST_SEARCH_AMOUNT)
    results.extend(query_results)
    time.sleep(0.5) #i run on 1, takes ages. 0.5s seems better
    print(f"Total fetched: {len(results)} posts from r/{sub}")
    print(f"Total fetched: {len(results)} posts from r/{sub}")
```

Fig. 5 delay in Query loop

```
for submission in results:
    time.sleep(0.2) #0.2 is near optimal but might actually remove it altoge
    if submission.score < 10: #for top/hot i run on 50, returns higher quali
        continue
    url = submission.url.lower()
    if "jpg" in url or "png" in url:
        try:
```

Fig. 6 delay in submission loop

Then in my **fetch_all_posts** function I run a Reddit's API fetch method **subreddit.search()**. I did a couple iterations where I changed the **sort=""** parameter to new, top and hot to include a variety of results in my dataset. I also

played with **time_filter=""** parameter to include older results, so I used **time_filter="all"** from time to time.

```
try:
    add_unique(subreddit.search(query, sort="top", time_filter="year", limit=limit))
except Exception as e:
    print(f"Search failed in {subreddit.display_name}: {e}")
```

Fig. 7 Subredit.search() method, top / year parameters

In case there are not enough results for these set parameters, I also included a fallback function that would run the fetching function and include different parameters, merging new, hot and top posts together to find more relevant results.

```
try:
    add_unique(subreddit.top(limit=limit))
    add_unique(subreddit.new(limit=limit))
    add_unique(subreddit.hot(limit=limit))
except Exception as e:
    print(f"Fallback fetch failed in {subreddit.display_name}: {e}")

return results
```

Fig. 8 Old 'fallback' function

As a result of these iterations, I was able to initially scrape around 500 images from two subreddits: r/Ukraine and r/UkraineWarVideoReport. However, after I went through the process of manual filtration, I only got 85 images in my cleared dataset. It was not enough for my artwork, so I made a few more improvement to my scraper which made it more effective.

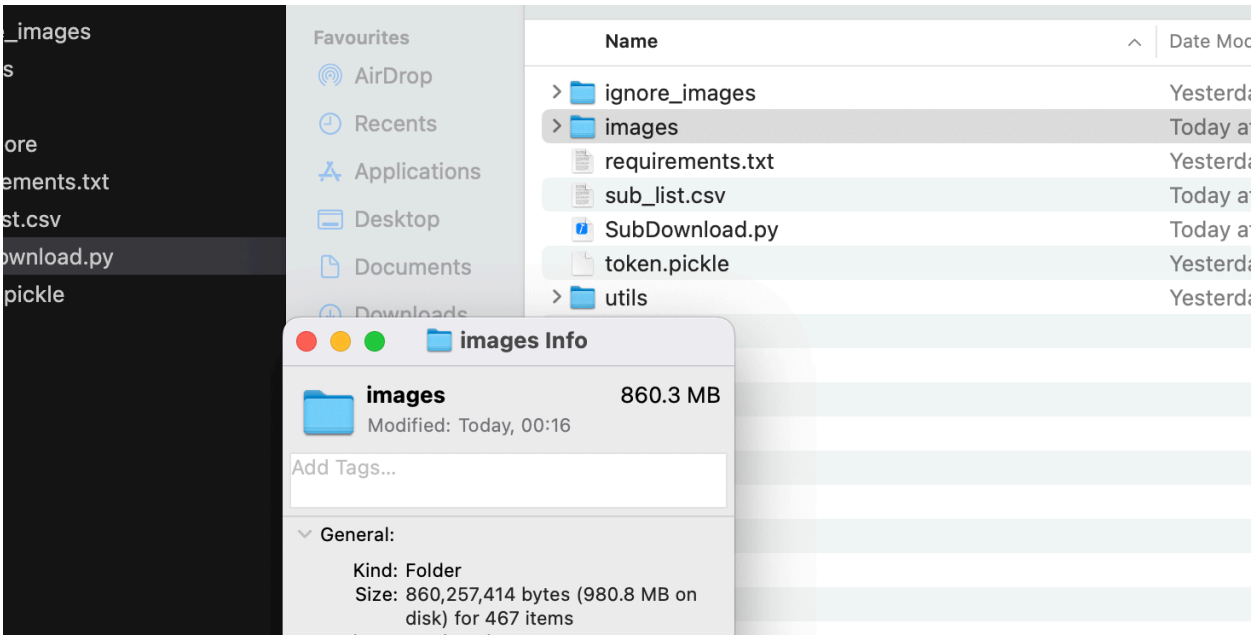


Fig. 9 Scraped results from the first script iteration

So to improve the output and quality of the results I revised my code and fixed a few things. Firstly, I fixed my syntax for the 'fallback' part of the **subreddit.search()** method to actually include the keywords (query parameter) for it as well, and get more specific results to my search.

```
try:
    add_unique(subreddit.search(query, sort="new", time_filter="month", syntax="lucene", limit
    add_unique(subreddit.search(query, sort="hot", time_filter="all", syntax="lucene", limit=l
except Exception as e:
    print(f"Fallback fetch failed in {subreddit.display_name}: {e}")

return results
```

Fig. 10 Fixed subreddit.search() method for my fallback function

I included the **syntax="lucene"** parameter to broaden my keyword search, because it allowed me to use wildcards, and search keywords specifically in the title section:

```
POST_SEARCH_AMOUNT = 800 # number of posts to search through
QUERY = [ # my keywords
    "kill* OR injure*",
    "civilian* OR murder*",
    "terror* OR drone*",
]
IMAGE_RESIZE_DIMS = (224, 224)
```

Fig. 11 Keywords with * extension for broader reach

the * symbol after the word means that words that derive from it (e.g. kill* → killed, killers, killing etc.) will also be included in the search

I also used **title:** syntax a few times but I find that the results I got were not much different

```
POST_SEARCH_AMOUNT = 800 # number of posts to search through
QUERY = [ # my keywords
    "title:(kill* OR injure*)",
    "title:(civilian* OR murder*)",
    "title:(terror* OR drone*)",
]
IMAGE_RESIZE_DIMS = (224, 224)
```

Fig. 12 Keywords with title: extension

Another very important parameter that I added to my **reddit.search()** method is **params={"include_over_18": "on"}** which lets me scrape through NSFW posts. It's quite important for my specific project, because a lot of graphic content which shows the aftermath of Russia's attacks is filtered for 18+ users only, and I think that before I added this parameter my scraper was omitting these posts. I definitely got more 'graphic' images after I included it.

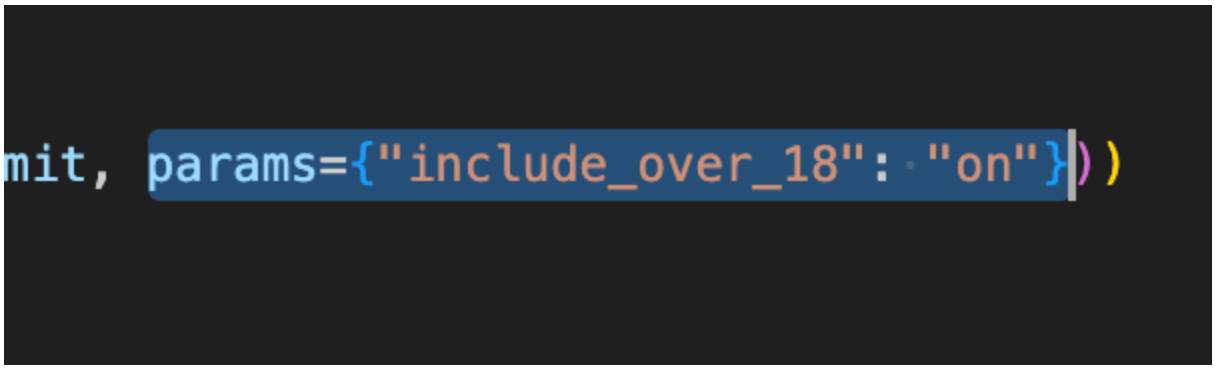


Fig. 13 Adding NSFW parameter to the search

Another thing I did to add more variety to my search was simply including more subreddits. I scraped through other Ukraine and war related subreddits: r/UkraineConflict, r/RussiaUkraineWar2022, r/WarinUkraine, r/UkraineInvasionVideos.
I also run a specific search by the keywords such as Kyiv, Kherson, Kharkiv (Ukrainian cities) in the subreddits r/Europe and r/Worldnews, because the most horrific attacks were also shared there frequently.

As a result of these improvements and continuous iterations / tweaking the keywords and search parameters, I scraped 1335 images total.

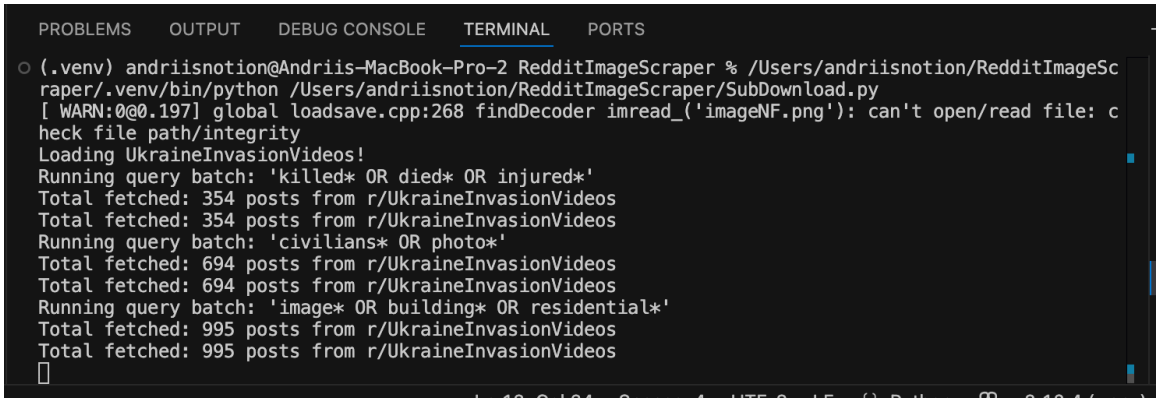


Fig. 14 Script runs in the terminal

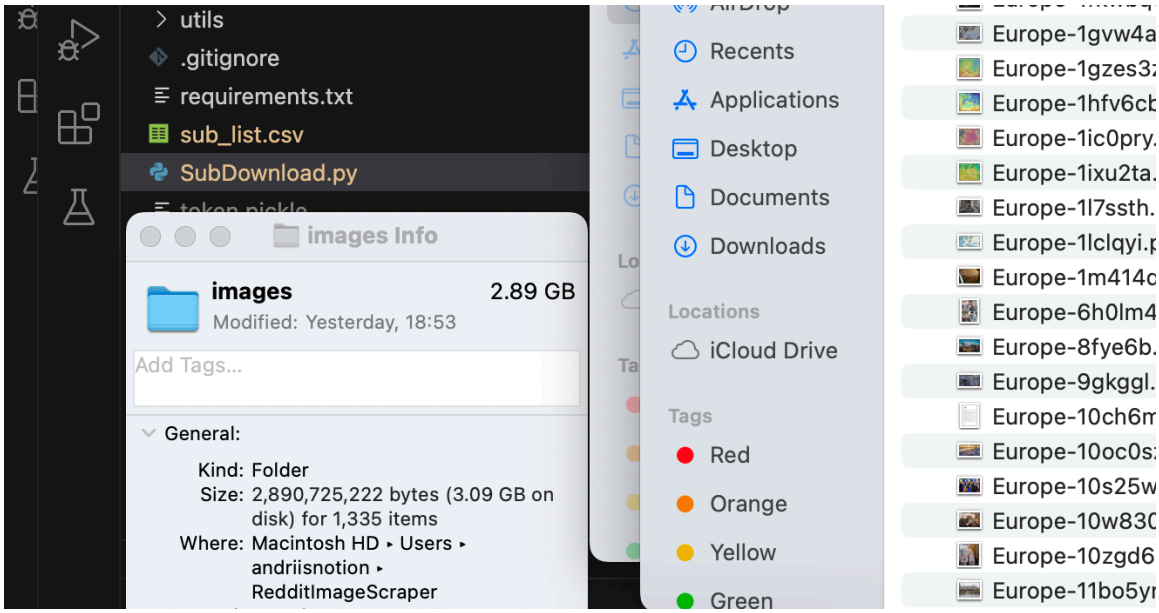


Fig. 15 Final scraped results - 1335 photos total

It's important to consider the limitations of the scraper I use. For instance, I was only scraping images with **jpg** or **png** format, and most posts I saw in the subreddits with the keywords I was looking for, were videos. Another point is that I am limited by the API's capabilities and search limit, and I also didn't want to push it to the max due to my concerns with my ip getting banned or my account getting frozen.

When it comes to keywords, I think that I covered a very broad range - from the popular keywords that show Russia's crimes or attacks, to location specific words that can target the missile and drone attacks. Same goes for the subreddits that I scraped through - I included around 10 different subreddits that are directly or indirectly related to the war in Ukraine, and run at least a few iterations of my script for each of them. For the larger subreddits, like r/Ukraine - I run over 20-30 iterations of the script with slightly different keywords, at the different stages of my scraper's development.

Dataset filtration and curation

Once I was done with scraping, I had to manually go through the scraped images to filter out the ones I don't need for my project. Mostly, those were screenshots of maps and social media posts, pictures of politicians, memes, arts, and other unrelated content.

I left the images of the infrastructure and residential images being destroyed or damaged, pictures of captured Ukrainian soldiers that returned from the captivity, reports from the social media regarding the recorded war crimes committed by the russians, photos of people grieving and other images that show the suffering of common people. Some images were very graphic and honestly traumatising, showing dead bodies, blood and heavily injured people. I decided to leave them in my filtered dataset, to preserve the memory about the victims, and as an act of paying my respect to people who lost their lives.

I also left some images that are highly symbolic, showing large pro-Ukrainian demonstrations, or symbols of resistance. Some photos that I decided to leave as well are neutral in terms of their emotional value, showing soldiers on the battlefield, civilians standing with the rifles (preparing to defend their city from the enemy, which was very common in 2022 and 2023), photos of explosions and military equipment. I decided to keep these kind of photos to show how life looks like in a war-torn country, where every mundane part of life is affected by the war.

Also, there are some photos of civilians (families and children) with no captions or additional information. Unfortunately, in most cases, these photos show people who died in the terror attacks on the residential infrastructure.

Overall, I left 569 photos from the scraped dataset of 1335 photos. It is a very big improvement in the relative quality of the scraped images, as initially I left only 85 photos from the dataset of around 500 images (I went from 20% successfully filtered to 40%). So, it suggests that the improvements I made in the code, as well as my strategy, worked well, and I was able to target the keywords more effectively with the new iterations.

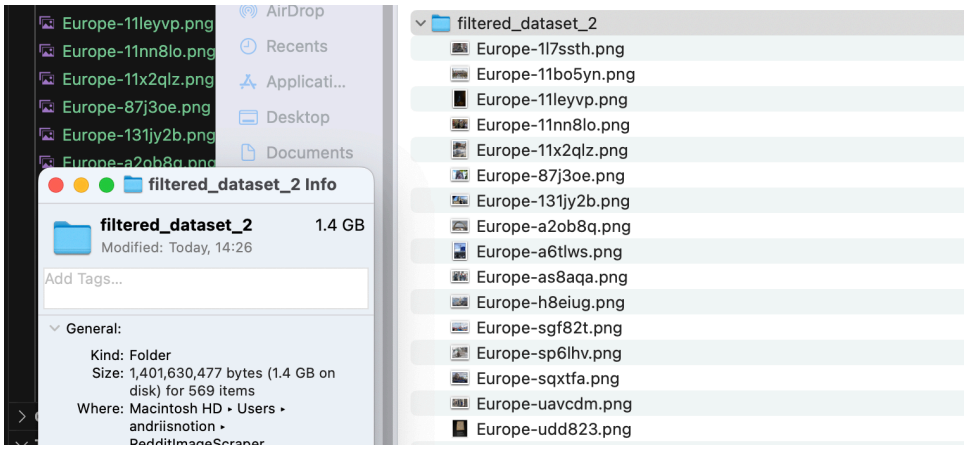


Fig. 16 Filtered dataset results - 569 photos

Fig. 17 Example photo from the dataset 1 - destroyed residential building

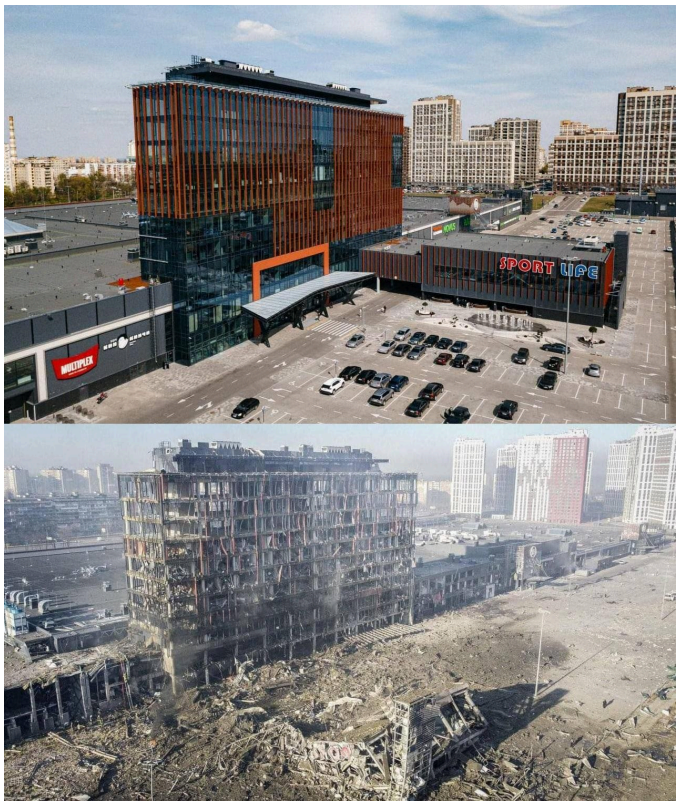


Fig. 18 Example photo from the dataset 2 - destroyed mall with a ballistic missile

Making the Artwork from the Dataset

I decided to use MatPlotLib Python library for my artwork, because I have more experience working with it, and because I wanted to make a collage from the images I scraped and this library was a good option to do it.

For my first iteration / artwork 1 I made this collage

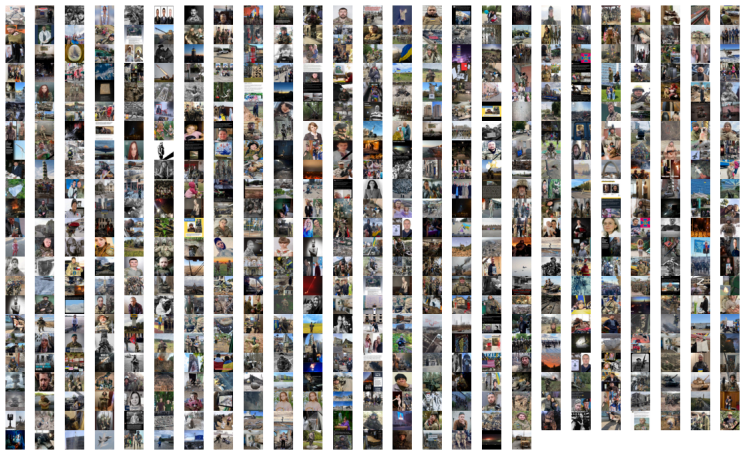


Fig. 19 ‘Faces of the War’ collage - iteration one

artwork1.py > ...

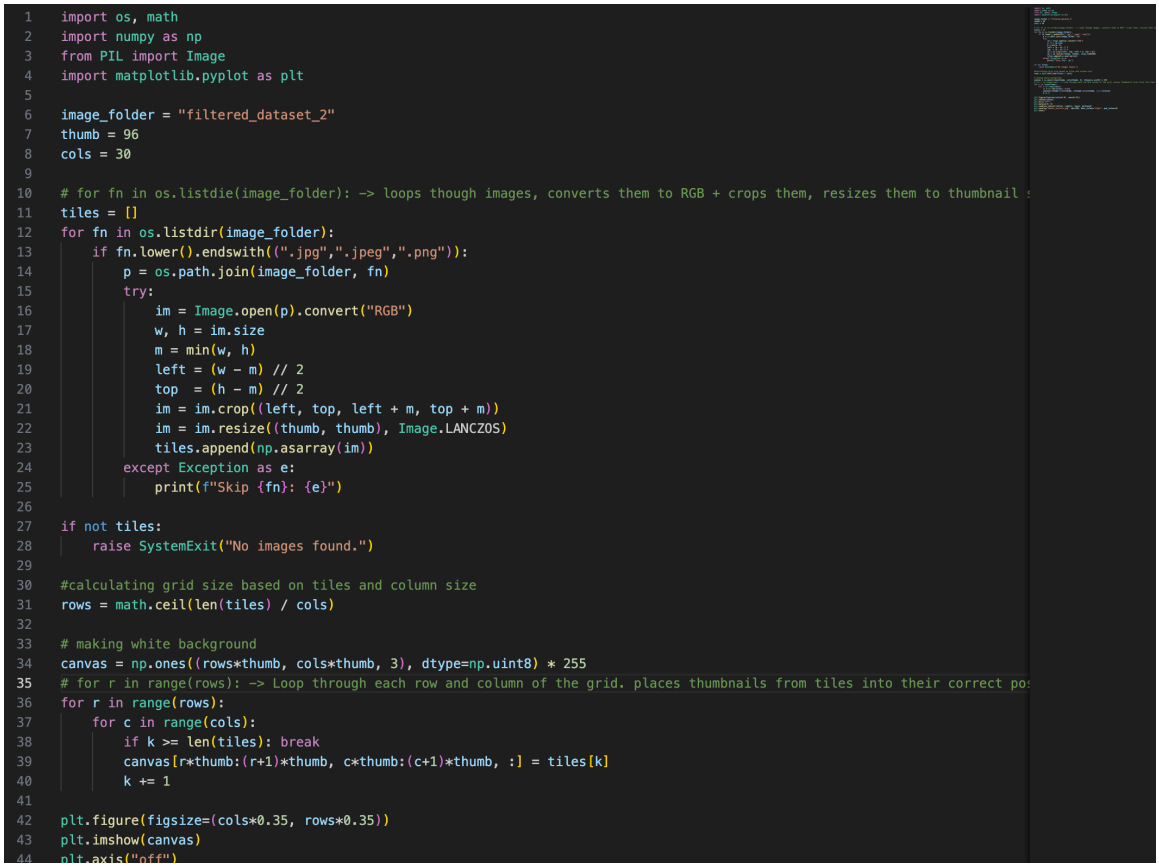
```python
import os
import matplotlib.pyplot as plt
from PIL import Image

image_folder = "filtered_dataset_2"

images_per_row = 25
thumb_size = (64, 64)

all_images = []
for filename in os.listdir(image_folder):
    if filename.lower().endswith((".png", ".jpg", ".jpeg")):
        img_path = os.path.join(image_folder, filename)
        # opens image from the source folder, convert to RGB, and resize to defined thumbnai
        try:
            img = Image.open(img_path).convert("RGB")
            img = img.resize(thumb_size)
            all_images.append(img)
        except Exception as e:
            print(f"Error loading {filename}: {e}")

# calculates number of rows for the artwork. total images in the source foulder divided by i
rows = (len(all_images) // images_per_row) + 1
# makes a matplotlib grid for the artwork
fig, axes = plt.subplots(rows, images_per_row, figsize=(images_per_row, rows))
# set the spaces between the images
fig.subplots_adjust(wspace=0, hspace=0)

# for r in range(rows): -> loops through each row on the grid and displays images accordingl
idx = 0
for r in range(rows):
    for c in range(images_per_row):
        ax = axes[r, c] if rows > 1 else axes[c]
        ax.axis("off")
        if idx < len(all_images):
            ax.imshow(all_images[idx])
            idx += 1

plt.show()
```

Fig 20. Source code (i1)

However, I wasn’t satisfied with the outcome because of the gaps between the columns.

I made a second iteration of the collage which has slightly larger thumbnail/tile size and has no gaps between the photos.

Fig. 21 Collage-artwork (iteration 2). Compressed for documentation purposes



```python
import os, math
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

image_folder = "filtered_dataset_2"
thumb = 96
cols = 30

# for fn in os.listdie(image_folder): -> loops though images, converts them to RGB + crops them, resizes them to thumbnail s
tiles = []
for fn in os.listdir(image_folder):
    if fn.lower().endswith((".jpg",".jpeg",".png")):
        p = os.path.join(image_folder, fn)
        try:
            im = Image.open(p).convert("RGB")
            w, h = im.size
            m = min(w, h)
            left = (w - m) // 2
            top  = (h - m) // 2
            im = im.crop((left, top, left + m, top + m))
            im = im.resize((thumb, thumb), Image.LANCZOS)
            tiles.append(np.asarray(im))
        except Exception as e:
            print(f"Skip {fn}: {e}")

if not tiles:
    raise SystemExit("No images found.")

#calculating grid size based on tiles and column size
rows = math.ceil(len(tiles) / cols)

# making white background
canvas = np.ones((rows*thumb, cols*thumb, 3), dtype=np.uint8) * 255
# for r in range(rows): -> Loop through each row and column of the grid. places thumbnails from tiles into their correct pos
for r in range(rows):
    for c in range(cols):
        if k >= len(tiles): break
        canvas[r*thumb:(r+1)*thumb, c*thumb:(c+1)*thumb, :] = tiles[k]
        k += 1

plt.figure(figsize=(cols*0.35, rows*0.35))
plt.imshow(canvas)
plt.axis("off")
```

Fig. 22 Source code (iteration 2)

It was already a good result and matched with what I wanted to do for my final artwork. But I felt like I can make the artwork more symbolic and add a meaning/visual context to it, so I made a third iteration with the mask of the Ukrainian map.

I found a source image of the country’s silhouette on a white background to use as my mask source.

Fig. 23 Ukraine source image for mask

In the code I define the black color as the ‘inside’ portion of the mask. First I load the mask into greyscale to give each pixel a value from 0 to 255 (black to white).

```
# loads the mask image in grayscale and calculates its aspect rati
mask_img = Image.open(MASK_PATH).convert("L")
mask_w, mask_h = mask_img.size
aspect = mask_h / mask_w
```

Fig. 24 Loading the mask in greyscale

Then I make a condition that if the pixel is lower than 128 (255/2 - in the middle between white and black colors) it is counted as ‘black’ and is part of the ‘inside’ part of the mask.

```
# resizes mask to match canvas dimensions and convert to NumPy arr
mask = mask_img.resize((canvas_w, canvas_h), Image.BILINEAR)
mask_np = np.asarray(mask)

# decides that only black pixels will count as inside_bool
if FILL_WHERE.lower() == "black":
    inside_bool = mask_np < 128
else:
    inside_bool = mask_np >= 128
```

Fig. 25 Mask detection condition based on pixel value

Then I loop through the my row and column cells to validate if the cells are inside or outside my mask shape.

```
# gets grid cell coordinates that fall inside the mask shape
inside_cells = []
for r in range(rows):
    for c in range(COLS):
        y = r*TILE + TILE//2
        x = c*TILE + TILE//2
        if inside_bool[y, x]:
            inside_cells.append((r, c))

canvas = np.full((rows*TILE, COLS*TILE, 3), BG_COLOR, dtype=np.uint8)
```

Fig. 26 Looping though the cells to validate if they are inside the mask shape

And finally I populate the tiles(images) only in the cells that fall inside the mask.

```
# places tiles into the cells inside the mask
k = 0
n_tiles = len(tiles)
for (r, c) in inside_cells:
    tile = tiles[k % n_tiles]
    canvas[r*TILE:(r+1)*TILE, c*TILE:(c+1)*TILE] = tile
    k += 1
```

Fig. 27 Populating the cells inside mask

My final artwork:

Fig. 28 Final artwork collage (compressed for better quality)

Reference List / Learning Resources

1. **Reddit *API documentation***, *no date*. Available at: https://www.reddit.com/dev/api/ (Accessed: 14 August 2025).
2. **Clarity Coders** (2020) Reddit Image Scraper using Python (2020), [online video]. Available at: https://www.youtube.com/watch?v=sEIv8UcR3Go&t=293s (Accessed: 14 August 2025).
3. **ClarityCoders** (no date) *RedditImageScraper* [online]. GitHub repository. Available at: https://github.com/ClarityCoders/RedditImageScraper (Accessed: 14 August 2025).
4. **Shunankana, S.** (2021) 'Very Basic Photo Collage in Python using NumPy and Matplotlib', *Hashnode*, 24 January. Available at: https://sumanshunankana.hashnode.dev/very-basic-photo-collage-in-python-using-numpy-and-matplotlib (Accessed: 14 August 2025).
5. **Koca, D.** (2023) 'Create a Photo Collage with Python PIL', *Danyel Koca*, 30 December. Available at: https://www.danyelkoca.com/en/blog/make-photo-collage-with-python (Accessed: 14 August 2025).

AI Assistance Statement

Throughout the project, I used *ChatGPT 4-o* (OpenAI) to assist in researching and understanding Reddit's API capabilities and MatPlotLib masking logic, as well as for debugging and code review. AI was used as a consultative tool to support my problem-solving process, and all code was written, tested, and validated by me.