Gathering carpet image data

I will be using the Bing image downloader script, a custom-built library to quickly scrap available images in the Bing browser. To gather as much data as possible, I will be using various variations of "Azerbaijani carpets" terms and making sure we do not have any duplicates. After the first batch of the gathered images, I manually checked each shot to ensure no copies and removed irrelevant photos, as shown below.

In [2]:
```
# Ignore the Warning Messages
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
from glob import glob
from PIL import Image
```

The script will be using following variables:

1. **query_string**: String to be searched.
2. **limit**: (optional, default is 100) Number of images to download.
3. **output_dir**: (optional, default is 'dataset') Name of output dir.
4. **adult_filter_off**: (optional, default is True) Enable of disable adult filteration.
5. **force_replace**: (optional, default is False) Delete folder if present and start a fresh download.
6. **timeout**: (optional, default is 60) timeout for connection in seconds.
7. **filter**: (optional, default is "") filter, choose from [line, photo, clipart, gif, transparent]
8. **verbose**: (optional, default is True) Enable downloaded message.

For the initial run, I will limit image lookup to 100 to see how much original data can be gathered.

Run 1. azerbaijani carpets script

In [ ]:
```
from bing_image_downloader import downloader

downloader.download('azerbaijani carpets', limit=100,
                    output_dir='azerbaijani_carpets', adult_filter_off=T
rue,
                    force_replace=False, timeout=60, filter="photo", ver
bose=True)
```

1. Analyzing results of the first run.

After the first batch of the gathered images, I manually checked each shot to ensure no duplicates and removed irrelevant photos, as shown below. Out of 100 downloaded images, many of them are irrelevant. Hence, I will be removing them manually from the dataset. Some photos from the Faig Ahmed collection are not helpful for my purposes as they are not traditional and can distort the later process. Hence, I will be removing them too.

In [4]:
```
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

# Placeholder for images
images = []
for img_path in glob.glob('azerbaijani_carpets/azerbaijani carpets/*.pn
g'):
    images.append(plt.imread(img_path, 0))

# Showing the orignal images
plt.figure(figsize=(5 * 20, 10 * 10))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)


for i, image in enumerate(images):
    plt.subplot(10, 10, i+1).set_yticklabels([])
    plt.subplot(10, 10, i+1).set_xticklabels([])
    plt.imshow(image)
```

In [6]:
```
# the leftover images after manual removal
# Placeholder for images
images_left = []
for img_path in glob.glob('azerbaijani_carpets/azerbaijani carpets/*.pn
g'):
    images_left.append(plt.imread(img_path, 0))

# checking how many images are left after cleaning
len(images_left)
```

Out[6]: 22

In [7]:

```
# Showing the leftover images
plt.figure(figsize=(5 * 5, 5 * 9))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)


for i, image in enumerate(images_left):
    plt.subplot(5, 5, i+1).set_yticklabels([])
    plt.subplot(5, 5, i+1).set_xticklabels([])
    plt.imshow(image)
```



















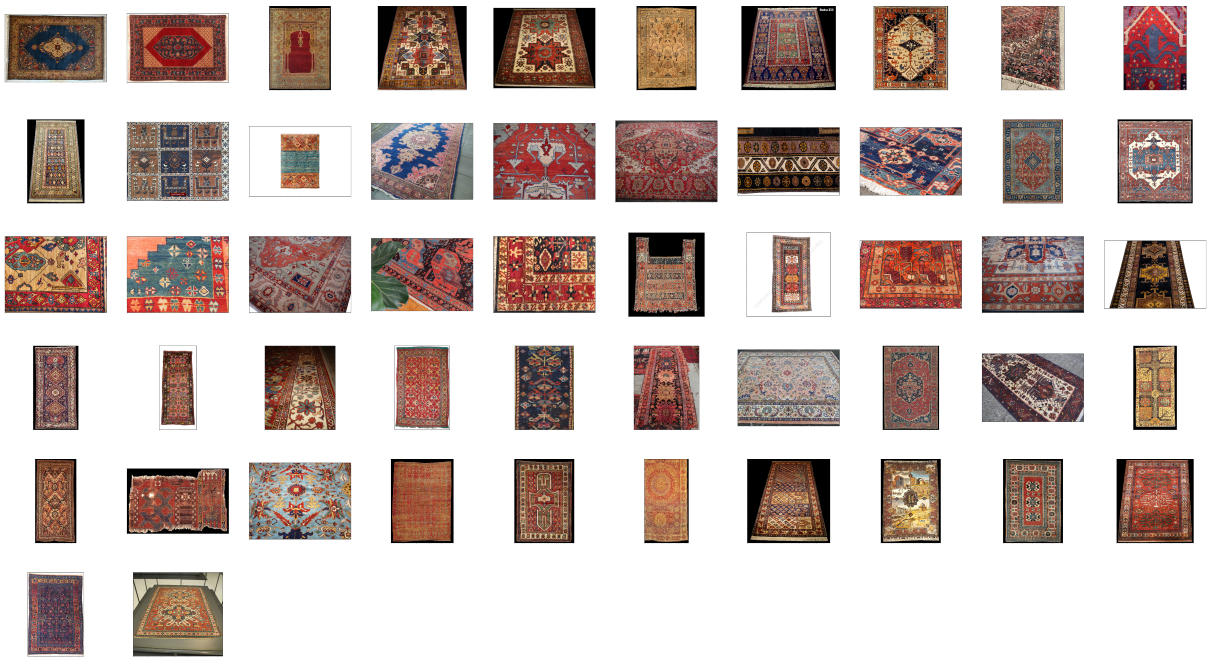



As we can see out of 100 scrapped images, we are left only with 22 and need to look for more.

Run 2. azerbaijani rugs script

In [ ]:
```
# the output here is just status update, so for better readability reaso
ns, I have shorted output
from bing_image_downloader import downloader

downloader.download('azerbaijani rugs', limit=100,
                    output_dir='azerbaijani_carpets', adult_filter_off=T
rue,
                    force_replace=False, timeout=5, filter="photo", verb
ose=True)
```

In [8]:
```
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

# Placeholder for images
images = []
for img_path in glob.glob('azerbaijani_carpets/azerbaijani rugs/*.png'):
    images.append(plt.imread(img_path, 0))

# Showing the orignal images
plt.figure(figsize=(5 * 20, 10 * 10))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)


for i, image in enumerate(images):
    plt.subplot(10, 10, i+1).set_yticklabels([])
    plt.subplot(10, 10, i+1).set_xticklabels([])
    plt.imshow(image)
```



2. Doing similar analysis and cleaning as with previous batch.

In [9]:
```
# the leftover images after manual removal
# Placeholder for images
images_left = []
for img_path in glob.glob('azerbaijani_carpets/azerbaijani rugs/*.png'):
    images_left.append(plt.imread(img_path, 0))

# checking how many images are left after cleaning
len(images_left)
```

Out[9]: 52

In [10]:
```
# Showing the leftover images
# plt.figure(figsize=(5 * 11, 5 * 9))
# plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.3
5)


# for i, image in enumerate(images_left):
#     plt.subplot(5, 11, i+1).set_yticklabels([])
#     plt.subplot(5, 11, i+1).set_xticklabels([])
#     plt.imshow(image)
```

From this batch, we received 52 images.

Run 3. Azerbaijan Carpets script

In [ ]:
```
from bing_image_downloader import downloader

downloader.download('Azerbaijan Carpets', limit=100,
                    output_dir='azerbaijani_carpets', adult_filter_off=T
rue,
                    force_replace=False, timeout=60, filter="photo", ver
bose=True)
```

In [11]:
```
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

# Placeholder for images
images = []
for img_path in glob.glob('azerbaijani_carpets/Azerbaijan Carpets/*.png'
):
    images.append(plt.imread(img_path, 0))

# Showing the orignal images
plt.figure(figsize=(5 * 20, 10 * 10))
plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)


for i, image in enumerate(images):
    plt.subplot(10, 10, i+1).set_yticklabels([])
    plt.subplot(10, 10, i+1).set_xticklabels([])
    plt.imshow(image)
```

In [12]:
```
# the leftover images after manual removal
# Placeholder for images
images_left = []
for img_path in glob.glob('azerbaijani_carpets/Azerbaijan Carpets/*.png'
):
    images_left.append(plt.imread(img_path, 0))

# checking how many images are left after cleaning
len(images_left)
```

Out[12]: 12

This search gave us 12 images. In total in 3 batches, I gathered 86 images.

Reflection on the current approach of data gathering.

Based on this approach, scrapping based on keywords gathering carpet images will be a tedious and lengthy process. My previous search has shown that there is n publicly available carpet database; however, I can find websites displaying Azerbiajni carpets and scrape them directly than scrapping the wider net. After a bit of research, I have found a website (http://www.azerbaijanrugs.com/guide/index.htm) hosting more than 2000 Azerbaijani and broader Caucasus/Persian region carpets categorized by carpet schools and regions. As manual download of 2000+ images will take "forever", I decided to write a script to automate the download process of the photos from this website directly. I used this course (https://www.codecademy.com/learn/learn-web-scraping?g_network=g&g_device=c&g_adid=525668108565&g_keyword=python%20beautifulsoup%20tutorial&g_acctid=2039-7011&g_adtype=search&g_campaign=US+Language%3A+Pro+-+Exact&g_keywordid=kwd-652592749664&g_campaignid=10030170700&g_adgroupid=102526217178&utm_id=t_kwd-652592749664:ag_102526217178:cp_10030170700:n_g:d_c&utm_term=python%20beautifulsoup%20tutorial&ut%20Exact&utm_source=google&utm_medium=paid-search&utm_content=525668108565&hsa_acc=2430397011&hsa_cam=10030170700&hsa_grp=102526217178&652592749664&hsa_kw=python%20beautifulsoup%20tutorial&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gclid=CyT2xoCFQMQAvD_BwE) to learn basics of how to scrap website and put together the algorithm.

We can see that this approach is way more fruitful as we got just from one category 186 unique carpet images. Hence, by categorically scrapping this website, I can get 2000+ original high-quality carpet images which would be enough for the training of the GAN models with some additional techniques.

In [ ]: