Dated: 21st September, 2023 **Homework # 1**
Submitted by: Shahram Najam Syed

Q1.1.1:

Ans: For our filter bank, we have been instructed to use 3 broad categories of spatial filters (in the context of image processing) for feature extraction, which are as follows:

1. Gaussian Filter
2. Laplacian of a Gaussian
3. Gaussian first derivative filters (directional along x-axis and y-axis)

Gaussian filter acts as a low pass filter, which is effective in reducing noise and perturbation in the image by filtering high frequency components within the image. This particular property of the Gaussian filter helps generate cleaner and more visually coherent signals (which are images in our case).

Laplacian of a Gaussian, often abbreviated as LoG, is a cascaded Gaussian filter and Laplacian operator. The cascaded Gaussian filter and laplacian operator first enhances the global structures within the image by filtering out noise and small artifacts, and then laplacian operators help capture the fast changing intensities within the image.

The gaussian derivative along the x-direction is a directional filter that detects and localizes vertical edges, lines, and features through rapid intensity change along the horizontal direction.

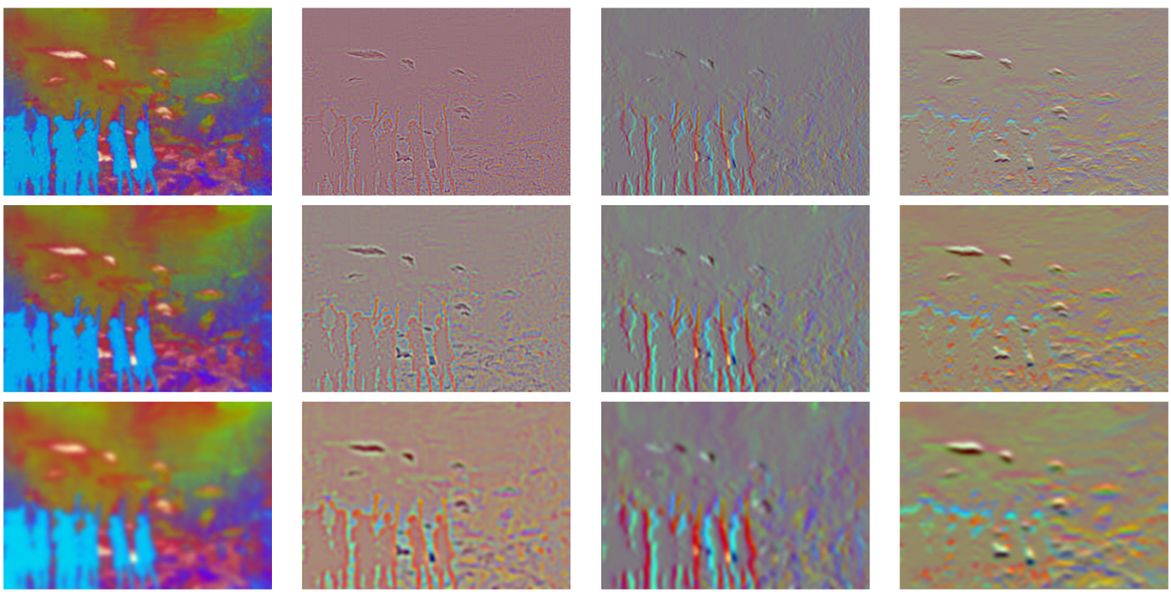
The gaussian derivative along the y-direction is a directional filter that detects and localizes horizontal edges, lines, and features through rapid intensity change along the vertical direction.

Reason for using multiple scales:
The images within the SUN dataset have a myriad of scene images, ranging from images taken up close to images taken from afar. In order to capture features of different aspects, scales, and varied intensity changes along different directions, multiple scales for the filter bank is used to achieve generalized feature extraction during the classification.

Q1.1.2:

Ans: Executing the main.py file yielded the following result using the default hyperparameters of:

- filter scales: [1, 2, 4];
- K: 10;
- alpha: 25;:



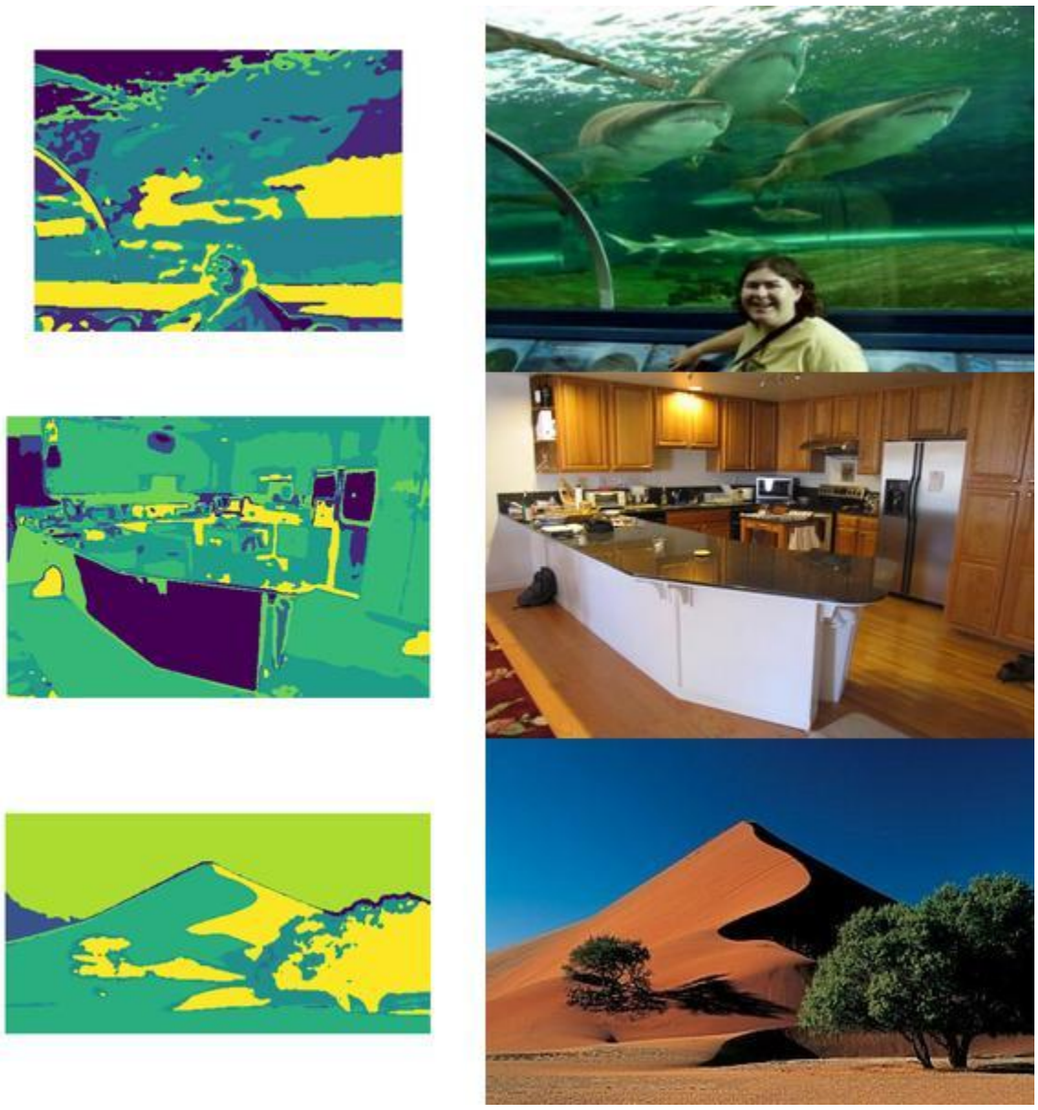
Q 1.2:

Ans: Coded in python, submitted in the zip.

Q 1.3:

Ans: Following is the collage of the images juxtaposed with their respective wordmaps. Since we were given flexibility to choose images of our own choice, hence I handpicked contextually and feature-rich images in the form of an aquarium (with edges and contours of the aquarium and sea life), kitchen (structured features of varying scales on the counter top), and a desert (sand & dune texture and desert vegetation). For these subset of images, the wordmap generated from the K-means clustering in an effective way, delineating the various objects within the scene. For the aquarium example, the sea life i.e. the sharks are somewhat accurately clustered into a single color map (with some error with the background), similarly it accurately maps the sand floor and the water. Similarly, in the kitchen scene, the cabinet's contours and edges are effectively grouped (but not quite all). And finally the desert scene effectively captures the scene, encoding the contours of the dune/mound and delineating the sand from the desert vegetation. Using the default parameters, an accurate delineation of the background and the foreground can be seen, but the delineation between the respective foreground elements/objects is not that accurate. The results visualized below are from the default hyperparameters of:

- filter scales: [1, 2];
- K: 10;
- alpha: 25;



Q 2.1:

Ans: Coded in python, submitted in the zip.

Q 2.2:

Ans: Coded in python, submitted in the zip.

Q 2.3:

Ans: Coded in python, submitted in the zip.

Q 2.4:

Ans: Coded in python, submitted in the zip.

Q 2.5:

Ans: The accuracy achieved using the following default parameters was just 47.75%.

Default hyperparameters:

- filter scales: [1, 2];
- K: 10;
- alpha: 25
- L: 1;

```
[[29.  1.  0.  2.  1.  2. 10.  2.]
 [ 1. 26.  8.  3.  3.  0.  1.  2.]
 [ 2.  5. 20.  1.  5.  6.  2.  6.]
 [ 0.  7.  1. 27. 12.  0.  2.  0.]
 [ 3.  6.  5. 12. 18.  3.  6.  4.]
 [ 1.  0.  2.  2.  5. 31.  7.  6.]
 [ 8.  5.  7.  3.  5.  5. 18.  8.]
 [ 6.  0.  7.  0.  1.  3.  4. 22.]]
0.4775
```

Q 2.6:

Ans: Referencing the confusion matrix in question 2.5, the classes misclassified the most are:

- Waterfall (class accuracy of 36%)
- Laundromat (class accuracy of 36%)

The confusion matrix further tells us that the waterfall is mostly misclassified as an “aquarium”, while the laundromat is mostly misclassified as a “kitchen”. Intuitively this does make sense since both the structure in conjugate scenes i.e. a waterfall & an aquarium, and a laundromat & a kitchen may seem structurally similar. Despite the fact that to human cognition we are able to accurately differentiate the scenes, but for such a basic scene classification system it is near impossible for us to featurize the finer (local structure) and coarser (global structure) of the scene. Hence the reason why, according to my intuitive understanding, the recognition system misclassified the most in these two classes.

Dated: 21st September, 2023 **Homework # 1**
Submitted by: Shahram Najam Syed

Q 3.1:

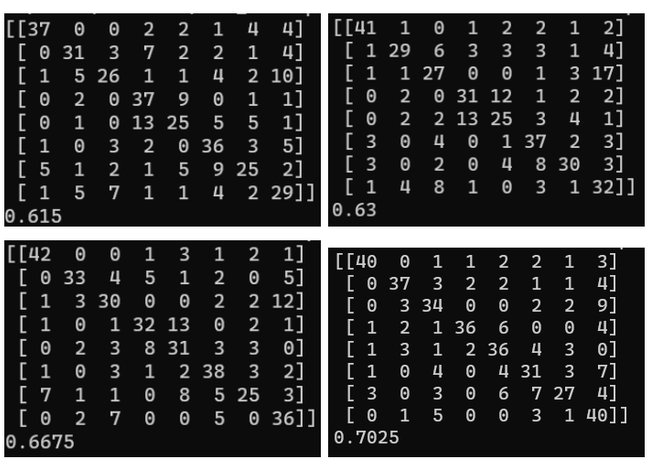
Ans: Following is the table presenting the selected set of experiments performed:

Experiment #	Feature Scales	No. of words (K cluster centroids)	Alpha	No. of layers in the SPM	Accuracy
Default parameters	[1,2]	10	25	1	47.75%
Exp # 1	[1,2,4]	10	25	1	51.5%
Exp # 2	[1,2]	50	25	1	49%
Exp # 3	[1,2]	10	50	1	53.75%
Exp # 4	[1,2,4]	50	50	2	61.5%
Exp # 5	[1,2,4,6]	75	75	3	63%
Exp # 6	[1,2,4,6,8]	100	100	3	66.75%
Exp # 7	[1,2,4,6,8,10]	100	100	4	70.25%

The above table shows an ablation table of how the varying hyperparameters effect the accuracy of the recognition system. From experiments # 1 - 4, only a single hyperparameter was varied stochastically to gauge the effects of the change on the recognition system accuracy. Following were my observations while playing with the parameters:

- **Feature scales:** Increasing the feature scales by a factor of 2 gradually alleviated the misclassification discussed in question 2.6 above. Intuitively I believe this is because of the fact that increasing the feature scales helps capturing coarser or global structures within the image, thus circumventing the misclassification observed with a basic default hyperparameter system.
- **K:** "K" is defined as the "# of words" in the opts.py description. Increasing the K increased the accuracy, but after reaching the value of hundred it plateaued and then further increasing it beyond 150 started decreasing the system accuracy. This might be because of overfitting due to K being larger than the scene complexity of the images.
- **Alpha:** "Alpha" is defined as the subset of pixels of the image sampled randomly. Variation of this had a tradeoff between the runtime complexity and the accuracy of the system. The optimal alpha value that I was able to use (without my system crashing) was 100.
- **L:** "L" is defined as the "No. of layers in the Spatial Pyramid Matching + 1". Increasing the number of layers yielded an increase in the accuracy, but after L=4, the accuracy started to slump down drastically (starting from L=5). The increasing behavior was because of the better understanding of the finer features within the scenes.

Exp # 4-7 signifies the compound effect of the hyperparameters when changed from their default values. There are some experiments that led to the accuracy dropping down to a mere ~16%, hence those experiments have been filtered out. Following are the screenshots of the confusion matrix and accuracy as calculated by sklearn.metrics.accuracy_score and sklearn.metrics.confusion_matrix:

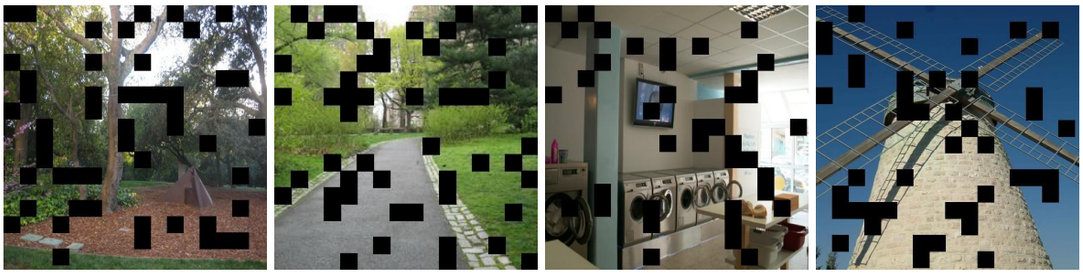


Confusion matrix and accuracy of exp # 4 (top-left), exp # 5(top right), exp#6 (bottom left), and exp#7 (bottom right)

Dated: 21st September, 2023 **Homework # 1**
Submitted by: Shahram Najam Syed

Q 3.2:

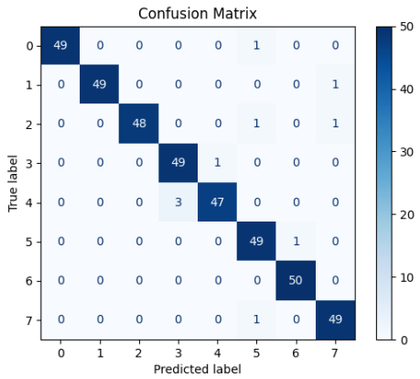
Ans: Data augmentation is a key aspect of any ML model, allowing to learn more from less, generalize better, and stand resilient against overfitting. The route I decided to take to improve my performance was motivated from a paper titled Random Erasing Data Augmentation by Zhong et.al. **[https://arxiv.org/abs/1708.04896].** But rather than randomly selecting the width and height of patches, I uniformly distributed each image into a grid of 16x16 (i.e. 256 patches) and applied random chance of blacking out the patches (with a weighted probability of only 20% to get blacked out). I randomly selected 25 images from the 8 classes we have, applied this augmentation and introduced it as additional training data. The examples of generated augmented images are:



And the maximum accuracy I was able to achieve was 80.5%. Increasing the number of blacked out patches decreased the accuracy severely, hence the 20%weighted probability is something I arrived upon through trial and error. The accuracy seems to also drop randomly for the random set of images sampled for this augmentation.

```
[[47  0  0  2  0  1  0  0]
 [ 1 41  5  0  0  2  1  0]
 [ 0  0 42  3  1  1  1  2]
 [ 5  3  0 37  0  4  0  1]
 [ 0  7  0  0 41  2  0  0]
 [ 0  2  9  0  0 35  3  1]
 [ 0  2  0  0  1  2 40  5]
 [ 0  0  9  0  0  2  0 39]]
0.805
```

Curious as to what a low-end convolutional neural architecture might have, and just by using mobilnetV2 (on my old GTX-1050 Ti machine), I got an astounding 97.5% accuracy on the test set. Following is the screenshot of the terminal along with the confusion matrix plotted on matlab.




```
Anaconda Prompt (Anaconda

../data\windmill/sun_bdtchkepcotagfds.jpg
../data\windmill/sun_botdebtxrpuybwmd.jpg
../data\windmill/sun_bqdtzxdxbmcuhzmb.jpg
../data\windmill/sun_bjtnzdtbdwmpvnfb.jpg
../data\windmill/sun_bdrjeendbkbiupuf.jpg
../data\windmill/sun_bxzsglbilsxviyio.jpg
../data\windmill/sun_buuzfnreqicjglpt.jpg
../data\windmill/sun_bphrlepchubmcwso.jpg
../data\windmill/sun_binbxnwxgeywaqwi.jpg
../data\windmill/sun_bfsrnfiwzdfujgpx.jpg
../data\windmill/sun_bzeplylllrvbae.jpg
../data\windmill/sun_bvfsbpllguhfsfsa.jpg
[[47  0  0  2  0  1  0  0]
 [ 1 41  5  0  0  2  1  0]
 [ 0  0 42  3  1  1  1  2]
 [ 5  3  0 37  0  4  0  1]
 [ 0  7  0  0 41  2  0  0]
 [ 0  2  9  0  0 35  3  1]
 [ 0  2  0  0  1  2 40  5]
 [ 0  0  9  0  0  2  0 39]]
0.805

(hw1) C:\Users\DELL\Desktop\Homework Folder\16-720\HW1\code>cd ../../

(hw1) C:\Users\DELL\Desktop\Homework Folder\16-720>cd CNN_test

(hw1) C:\Users\DELL\Desktop\Homework Folder\16-720\CNN_test>conda activate pt_env

(pt_env) C:\Users\DELL\Desktop\Homework Folder\16-720\CNN_test>python test.py
C:\Users\DELL\.conda\envs\pt_env\lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and ma
y be removed in the future, please use 'weights' instead.
  warnings.warn(
C:\Users\DELL\.conda\envs\pt_env\lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights'
 are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing `weights=None`.
  warnings.warn(msg)
Accuracy on the test dataset: 97.50%
```

References:

[1] Rafael, C. Gonzalez, and E. Woods Richard. "Digital Image Processing (-Global Edition)." (2018).
[2] Nixon, M.S., and Aguado, A.S. “Feature Extraction and Image Processing .” (2002).
[3] https://numpy.org/doc/stable/reference/generated/numpy.bincount.html
[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html
[5] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_laplace.html
[6] https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html
[7] https://www.geeksforgeeks.org/multiprocessing-python-set-1/
[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition (CVPR), 2006 IEEE Conference on, volume 2, pages 2169–2178, 2006.
[9] https://www.quora.com/How-should-we-understand-the-Spatial-Pyramid-Matching
[10] https://slazebni.cs.illinois.edu/publications/pyramid_chapter.pdf
[11] Zhong, Zhun, et al. "Random erasing data augmentation." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 07. 2020.