

HW1

Due	Apr 23 by 10am	Points	100	Submitting	a file upload	Available	Apr 8 at 12am - Apr 30 at 11:59pm	23 days
-----	----------------	--------	-----	------------	---------------	-----------	-----------------------------------	---------



This assignment was locked Apr 30 at 11:59pm.

HW1 is about:

1. Detection of keypoints in images,
2. Computation of deep features representing the detected keypoints,

Keypoint detection



Download the following set of 10 images [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) . In each image, detect 200 highest-response:

- SIFT keypoints by using the OpenCV Python library: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html  (https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html)
- Harris corners with subpixel accuracy by using the OpenCV Python library: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html#harris-corners  (https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html#harris-corners)


Output: Two tensors of (i,j) coordinates of 200 keypoints detected in the 10 images of [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) . Each tensor size is (10 x 200 x 2). Save the two tensors in the corresponding two files with names: **SIFT.pth** and **Harris.pth**

point description

For an image patch x centered at every keypoint that you detected in images from [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) , compute a 128-dimensional deep feature $f(x) \in \mathbb{R}^{128}$ using a CNN. For this, we have prepared a skeleton code for you that you will need to complete in order to achieve the required task. The skeleton code consists of two parts.

1) Patch extraction: The first skeleton code can be found at [keypoint_detector.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86407302/download?download_frd=1) . It takes as input an image and (i,j) coordinates of 200 keypoints detected in the image, and then extracts 32x32 patches x around each keypoint. If the input images are color images the skeleton code will map them into the corresponding grayscale images, so the extracted patched will have the size (1 x 32 x 32). The code also stacks all patches extracted from the 10 images of [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) in a tensor of size (10 x 200 x 1 x 32 x 32). This code is complete, and you do not need to modify it.

2) Computation of keypoint descriptors: The second skeleton code can be found at [keypoint_descriptor.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86463316/download?download_frd=1) . An image patch x (e.g., extracted around a keypoint) is input to a CNN for computing the corresponding deep feature $f(x) \in \mathbb{R}^{128}$ of the patch. This code is not complete, and you will need to modify it as follows.

1. Design new CNN2 and CNN3, starting from the provided initial CNN1 architecture in [keypoint_descriptor.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86463316/download?download_frd=1) .
2. Train all three networks CNN1, CNN2, and CNN3 on the provided training dataset.
3. Using CNN1, CNN2, and CNN3, compute the corresponding three sets of deep features of all your keypoints detected in [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) .

Output: Deep features of 200 keypoints detected in the 10 images of [images.zip](#)  (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) . Save the deep features in the corresponding files: **SIFT_features_CNN1.pth**, **SIFT_features_CNN2.pth**, **SIFT_features_CNN3.pth**, **Harris_features_CNN1.pth**, **Harris_features_CNN2.pth**, and **Harris_features_CNN3.pth**.

The following sections describe each task in more detail.

Designing the New CNNs

Design the new CNN2 and CNN3 by augmenting the CNN1 architecture provided in the skeleton code. **CNN2 should have at least two new additional convolutional layers relative to CNN1**, and all other layers the same as in CNN1. Also, **CNN3 should have at least two new additional convolutional layers relative to CNN2**. For each new convolutional layer that you add to CNN1, you would need to manually specify: size of the kernel, number of channels, stride, and other required hyper-parameters. You should **empirically find optimal hyper-parameters** on the training dataset so that you achieve a minimum loss in training.

Training the CNNs

The training set that you will use for training your CNNs consists of image patches, which can be found at: <http://phototour.cs.washington.edu/patches/>  (<http://phototour.cs.washington.edu/patches/>). You do not need to download this training set, since the skeleton code will automatically download it for you to your root folder on the Pelican server. The original patches are 64x64 pixels, and the skeleton code resizes them to 32x32 pixels.

The skeleton code organizes the training image patches into 100,000 triplets: (anchor, positive, negative), denoted as $(x, x+, x-)$. Thus, the data loader for forming the training triplets is already implemented for you. An anchor patch x and positive patch $x+$ **represent the same detail** of the scene but captured in two different images, and hence they should have very similar feature descriptors. An anchor patch x and negative patch $x-$ **represent two different details** from two different scenes, and hence they should not have similar feature descriptors. Therefore, for training our CNNs, we seek to minimize the following triplet loss function for every triplet :

$$L = \max(0, d(x, x+) - d(x, x-) + m)$$

where $d()$ is a distance between deep features of the corresponding patches computed by the CNN, specified as:

$$d(x, x') = ||f(x) - f(x')||^2$$

and $m > 0$ is a margin set to $m = 0.1$.

You do not need to implement this loss function, since it is already provided in the skeleton code.

Training of a CNN is conducted by iterating over many epochs. **Your task will be to select a suitable: learning rate, size of the mini-batch, and the number of epochs for training. The number of training epochs must be greater than 60.**

On the Pelican server, for a mini-batch size of 1024, 60 epochs of training will take about 1.5 hours, and the GPU memory usage will be about 6.5GB. **Make sure you plan your time accordingly** for running the experiments, as sometimes the Pelican server may not have enough resources to support these experiments.

Output: For each CNN, four plots of the training triplet loss over more than 60 epochs, for two different values of mini-batch size and two different learning rates, where one of these four plots should have the optimal hyper-parameters that you have empirically found.

Validation of Training of the CNNs

The skeleton code also forms a validation set of 100,000 triplets (x , $x+$, $x-$) from image patches which can be found at: <http://phototour.cs.washington.edu/patches/> [.\(http://phototour.cs.washington.edu/patches/\)](http://phototour.cs.washington.edu/patches/). The training and validation sets do not share the same triplets. Again, the data loader for forming the validation triplets is already implemented for you. For each CNN, you will need to use the validation set to select the best trained model and save its parameters. As the criterion, you will use the triplet loss estimated over the entire validation dataset, and select the training epoch for which you get the minimum triplet loss on the validation dataset.

Output: Saved architectures and parameters of the best model for CNN1, CNN2, and CNN3. For each CNN, four plots of the validation triplet loss over more than 60 epochs, for two different values of mini-batch size and two different learning rates, where one of these four plots should have the optimal hyper-parameters that you have empirically found.


Computing Deep Features

After training CNN1, CNN2, and CNN3 on the training dataset and selecting the best models for CNN1, CNN2, and CNN, you will run the provided skeleton code for extracting patches around (i, j) locations of keypoints detected in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) , and then pass these patches to your best models of CNN1, CNN2, and CNN3 for computing the corresponding deep features. The skeleton code will save the resulting deep features in: ***_features_CNN*.pth** (e.g., for SIFT and CNN1 the file name will be **SIFT_features_CNN1.pth**).

Output: Six files of deep features ***_features_CNN*.pth**, produced by the trained CNN1, CNN2, CNN3, of your SIFT and Harris keypoints detected in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) .

What To Turn In?

A compressed folder **name.tar.gz** (please use the Linux command **tar** for compression) that contains the following files:

-  30 points) A pdf showing the eight plots of the training and validation triplet losses for each CNN over at least 60 epochs for two values of mini-batch size and two learning rates (a total of 24 plots for CNN1, CNN2, and CNN3). For each CNN, clearly mark the plots with the optimal hyper-parameters that you have empirically found, and the epoch for which you got the best model.
2. (5 points) **SIFT.pth** and **Harris.pth**, as the (10 x 200 x 2) tensor of (i, j) coordinates of 200 keypoints detected in the 10 images of [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) .
3. (5 points) **SIFT_patches.pth** and **Harris_patches.pth** consisting of the extracted patches from the 10 images of [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) for SIFT and Harris keypoints, where each file is a tensor of size (10 x 200 x 1 x 32 x 32). Importantly, the patches in these files must be in the same order as (i, j) coordinates in the corresponding **SIFT.pth** and **Harris.pth** files.
4. (5 points) **CNN1.py**, **CNN2.py**, and **CNN3.py** with the saved architecture and hyper parameters of your best models.
5. (5 points) **CNN1.pth**, **CNN2.pth**, and **CNN3.pth** with saved parameters of the best model for CNN1, CNN2, and CNN3.
6. (5 points) Six files of deep features ***_features_CNN*.pth**, produced by the best model of CNN1, CNN2, CNN3, of your SIFT and Harris keypoints detected in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) . Importantly, the deep features in these files must be in the same order as (i, j) coordinates in the corresponding **SIFT.pth** and **Harris.pth** files.

Please double-check that you can uncompress your folder with the Linux command: **tar -xzf**

Submit your compressed folder on Canvas before 10am on April 23, 2021. The penalty for late homework submission is a 5% of the maximum score, for every hour after the deadline.

Grading

In addition to the above listed points, we will also score the accuracy of your keypoint detection and description, as well as your CNN training.

First, we will compute a Euclidean distance between (i, j) coordinates of your keypoints and (i, j) coordinates of our keypoints for the 10 images of [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) . Since we will also use the same OpenCV Python library for keypoint detection, the Euclidean distance between our keypoints and yours should be zero. We will rank all students based on this Euclidean distance and give partial credit for non-zero distances according to this ranking.

Second, for evaluating your keypoint description, we will use your files **SIFT_patches.pth**, **Harris_patches.pth**, **SIFT_features_CNN1.pth**, **Harris_features_CNN1.pth**. For each image in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) , we will compare your deep features $f_{i,j}$ produced by your best model for CNN1 for keypoints that you detected at coordinates (i, j) with our deep features $f_{i,j}^*$ produced by our best model for the same keypoints. For this comparison, we will compute the MSE loss:

$$L_{\text{MSE}} = \frac{1}{10 \cdot 2 \cdot 200} \sum_{\text{image}} \sum_{\text{SIFT, Harris}} \sum_{(i,j)} \|f_{i,j} - f_{i,j}^*\|^2$$

We will rank all students based on this MSE loss and give partial credit for large loss values according to this ranking.

Third, we will form a new test set of triplets (x , $x+$, $x-$) from image patches which can be found at: <http://phototour.cs.washington.edu/patches/> [.\(http://phototour.cs.washington.edu/patches/\)](http://phototour.cs.washington.edu/patches/). You do not have access to this test set. We will compute the triplet loss of your best models for CNN1, CNN2, and CNN3 on this test set. We will rank all students based on this test triple loss, and give partial credit for large loss values according to this ranking.

Rubric

- 30 points = If you submit everything from the above turn-in list,
- 10 points = If (i, j) coordinates of your keypoints detected in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) are accurately computed, in terms of the Euclidean distance from ours
 - Based on the aforementioned Euclidean distance, students will be ranked and split into 3 groups. The group with top-ranked results will get 10 points. The group with bottom-ranked results will get 3 points. The group with middle-ranked results will get 6 points.
- 20 points = If your deep features of keypoints detected in [images.zip](#) [↓](#) (https://canvas.oregonstate.edu/courses/1811612/files/86219050/download?download_frd=1) are good, in terms of the sufficiently small MSE loss L_{MSE}
 - Based on the aforementioned MSE loss, students will be ranked and split into 3 groups. The group with top-ranked results will get 20 points. The group with bottom-ranked results will get 6 points. The group with middle-ranked results will get 12 points.
- 40 points = If your triplet loss is sufficiently small on the test triplets formed from patches in <http://phototour.cs.washington.edu/patches/> [.\(http://phototour.cs.washington.edu/patches/\)](http://phototour.cs.washington.edu/patches/).
 - Based on the aforementioned triplet loss estimated on our test triplets, students will be ranked and split into 3 groups. The group with top-ranked results will get 40 points. The group with bottom-ranked results will get 13 points. The group with middle-ranked results will get 26 points.