ADVANCED IMAGE PROCESSING

ASSIGNMENT 1

NAME- JYOTISH RANJAN

M. Tech AI,2023

1. **PCA-SIFT** **(20 Marks)**

   - Compute the PCA-SIFT feature descriptors for the images given here (You can ignore the second step).
   - Modify the images by (a) Scaling, (b) Rotation (c) Gaussian blur and obtain the keypoints for these images.
   - Analyse the keypoints detected qualitatively and quantitatively (number of keypoints detected in each case).

CODE IMPLEMENTATION:

CODE link

Step 1: Scale Space Extrema Detection

1. **Determine the number of images needed for this octave.**
2. **Blur the original image multiple times to create a set of blurred images for this octave.**
3. **Compute the difference between consecutive blurred images to obtain the difference of Gaussians.**
4. **Find extremes (key points) using 3 consecutive differences of Gaussians store it.**
5. **Stack three layers (`dog_lower`, `dog_current`, `dog_upper`) vertically to form `stacked_layers`.**
6. **Identify local maxima and minima within the 3x3x3 neighbourhood of each pixel in `stacked_layers`.**
7. **Iterate over each pixel in the image (excluding borders) to find key points.**
8. **Check if the pixel is a local maximum or minimum in its neighborhood and if its absolute value exceeds a certain threshold (here, 2).**
9. **If conditions are met, add the pixel coordinates `(i, j)` and scale_octave information `(k, octave_no)` to the list `final_keypoints`.**

Step 2: Orientation Assignment

1. **Loop through each unique scale and octave where key points are detected.**
2. **Retrieve the locations of key points for the current scale and octave.**
3. **Access the Gaussian image corresponding to the current scale and octave from the Gaussian image pyramid.**
4. **Compute the gradients in the x and y directions using the Sobel operator on the Gaussian image to get dx and dy.**

5. **Calculate the gradient magnitude and orientation using the computed dx and dy.**
6. **Determine the radius of the neighbourhood around the current key point based on its scale.**
7. **Iterate over the neighbourhood around each key point.**
8. **For each pixel in the neighbourhood, compute the orientation bin based on its gradient orientation.**
9. **Increment the corresponding bin in the histogram with the magnitude of the gradient at that pixel.**
10. **Find the bin with the maximum count in the histogram.**
11. **Determine the dominant orientation angle using the index of the maximum bin.**
12. **Append the orientation angle and magnitude of the current key point to its entry in the final key points list.**

<u>Step 3: Extract PCA Descriptors:</u>

1. **Loop through key points: Iterate over each key point.**
2. **Extract patch(41X41): Retrieve the patch from the Gaussian image around the key point.**
3. **Rotate patch: Rotate the patch based on the key point's orientation.**
4. **Calculate gradients: Compute the gradients (dx and dy) of the patch.**
5. **Extract central square from gradients(39X39): Extract a central square region from both dx and dy gradients.**
6. **Flatten and concatenate gradients (3042): Flatten the 2D arrays of dx and dy gradients into 1D arrays, then concatenate them.**
7. **Normalize the descriptor: Normalize the concatenated descriptor.**
8. **Append normalized descriptor: Add the normalized descriptor to the numpy array.**
9. **Perform PCA (64 dimensions): Initialize PCA with desired number of components.**
10. **Fit and Transform PCA to descriptors: Fit the PCA model to the descriptors.**

<u>Qualitative Study:</u>

- Rotation on both images has no effect on number of key points detected which clearly makes PCA-SIFT **rotation invariant**.
- Gaussian Blur tends to reduce the number of key points since it smooths out the image, making distinct feature less pronounces. This leads to fewer key points detected.
- Scaling Down reduces the number of key points in the image, as features become less distinct at smaller scales.
- Conversely, Scaling Up the image might increase the number of key points.
- As we can see number as rate of key points detected per octave decreases due to continuous down sampling of image.
- After applying PCA, dimension of descriptor reduces to 64.

Note – This implementation of PCA SIFT is not perfect but an effort to replicate basic steps of PCA-SIFT.

Quantitative Results:

Building:

Approx Number of Key points Detected:

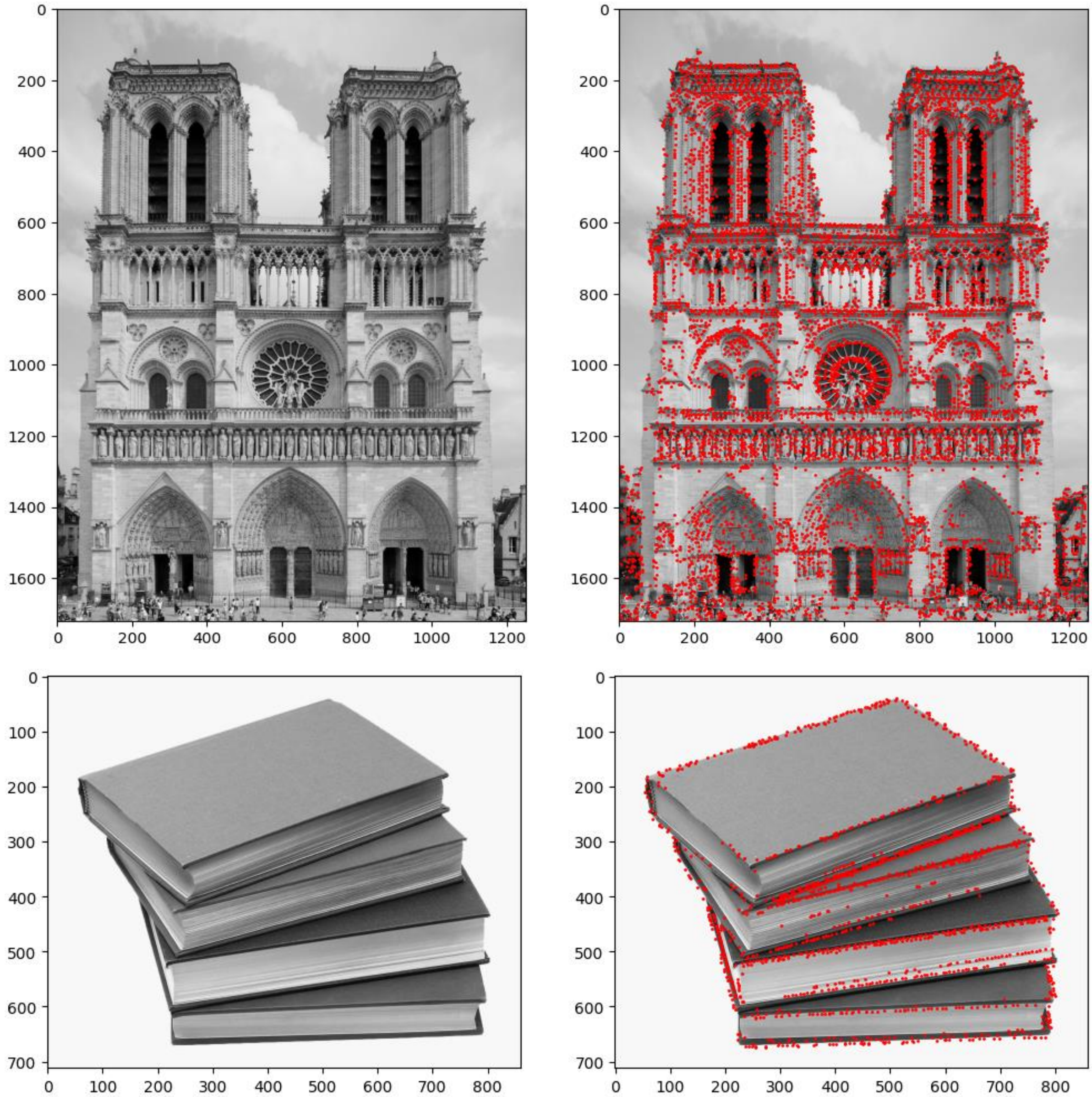
No. Of Octaves → Modified Building	1	4	Max no. of octaves possible
Original	5847	7815	7864
Scaled Up	12015	16283	16463
Scaled Down	1580	2154	2166
Gaussian Blur	6331	8361	8410
Rotation	5847	7815	7864

Books:

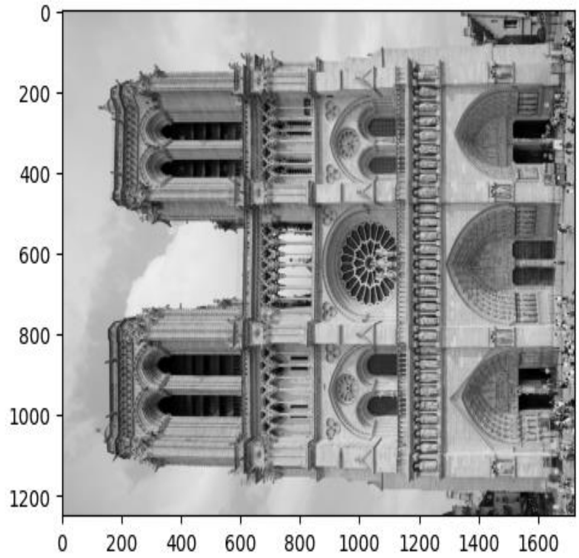
Approx Number of Key points and Descriptors Detected:

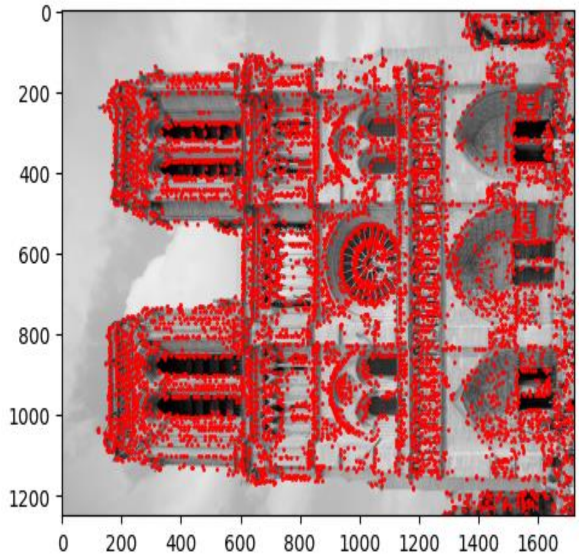
No. Of Octaves → Modified Books	1	4	Max no. of octaves possible
Original	1039	1872	1926
Scaled Up	1097	2617	2841
Scaled Down	287	534	535
Gaussian Blur	910	1578	1811
Rotation(rotation 90)	1039	1872	1926

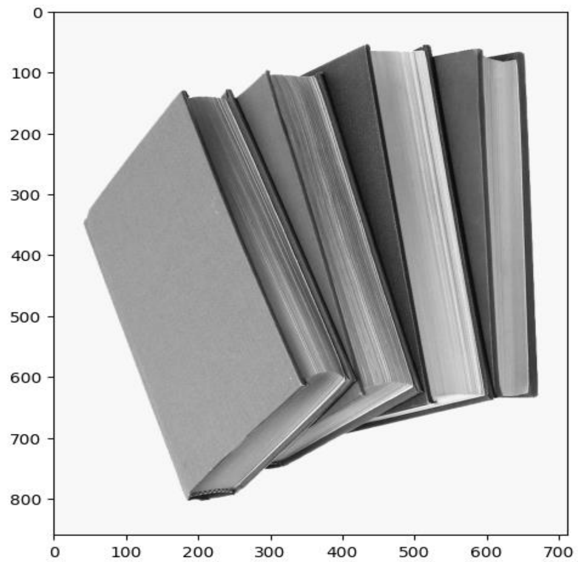
1. Original Images

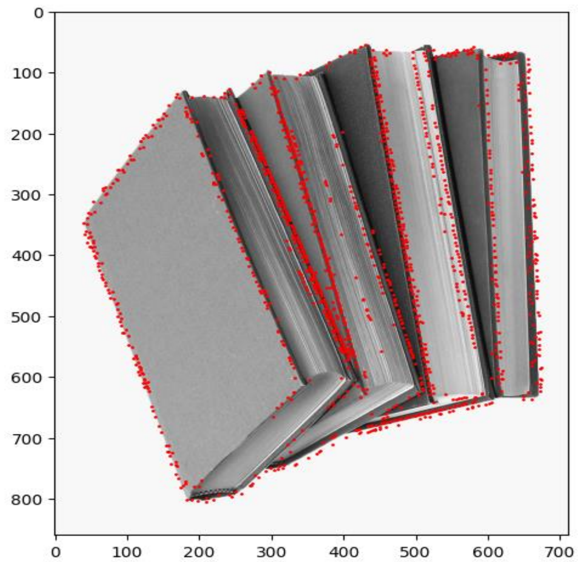


2. Rotated Building

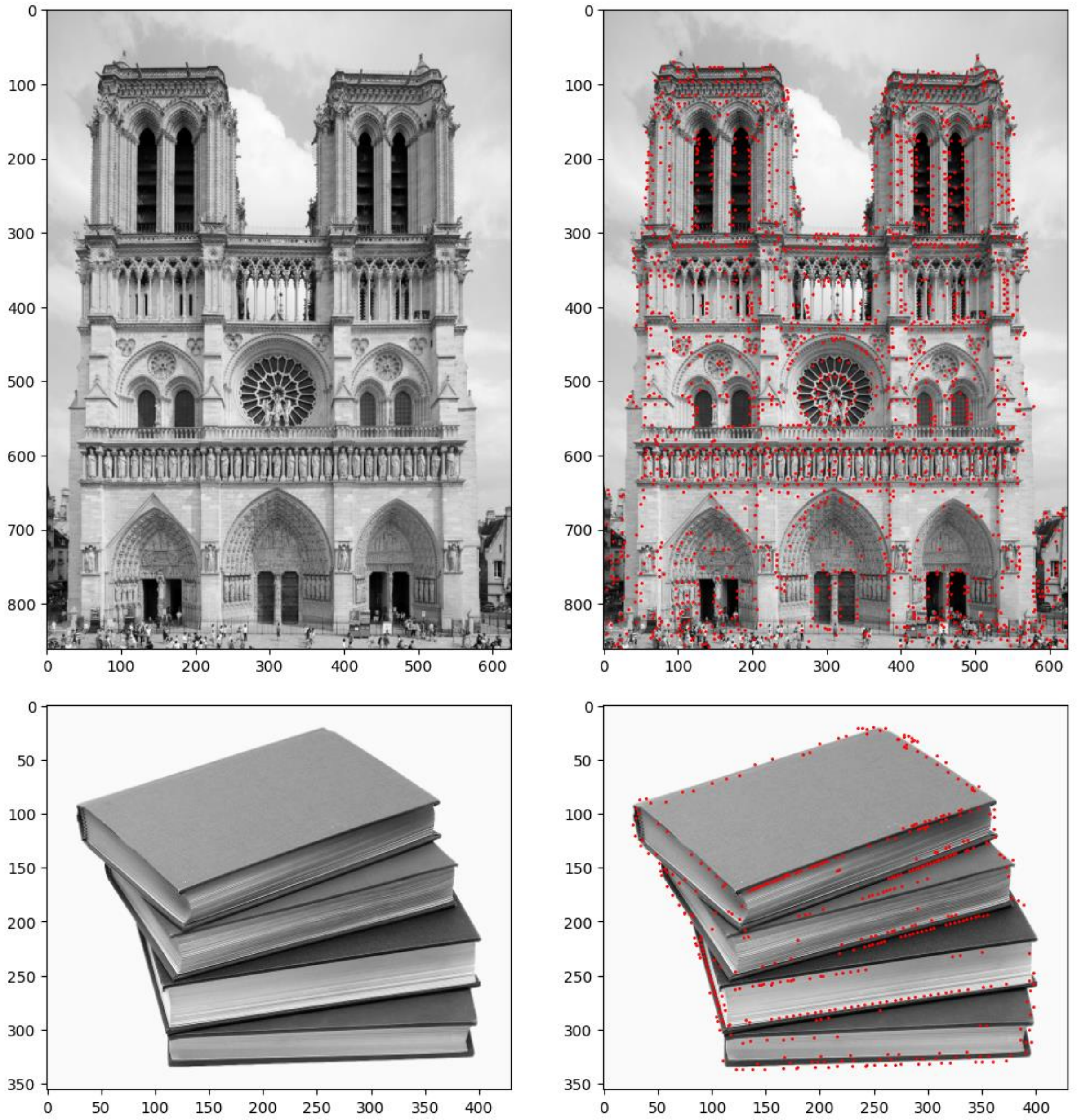




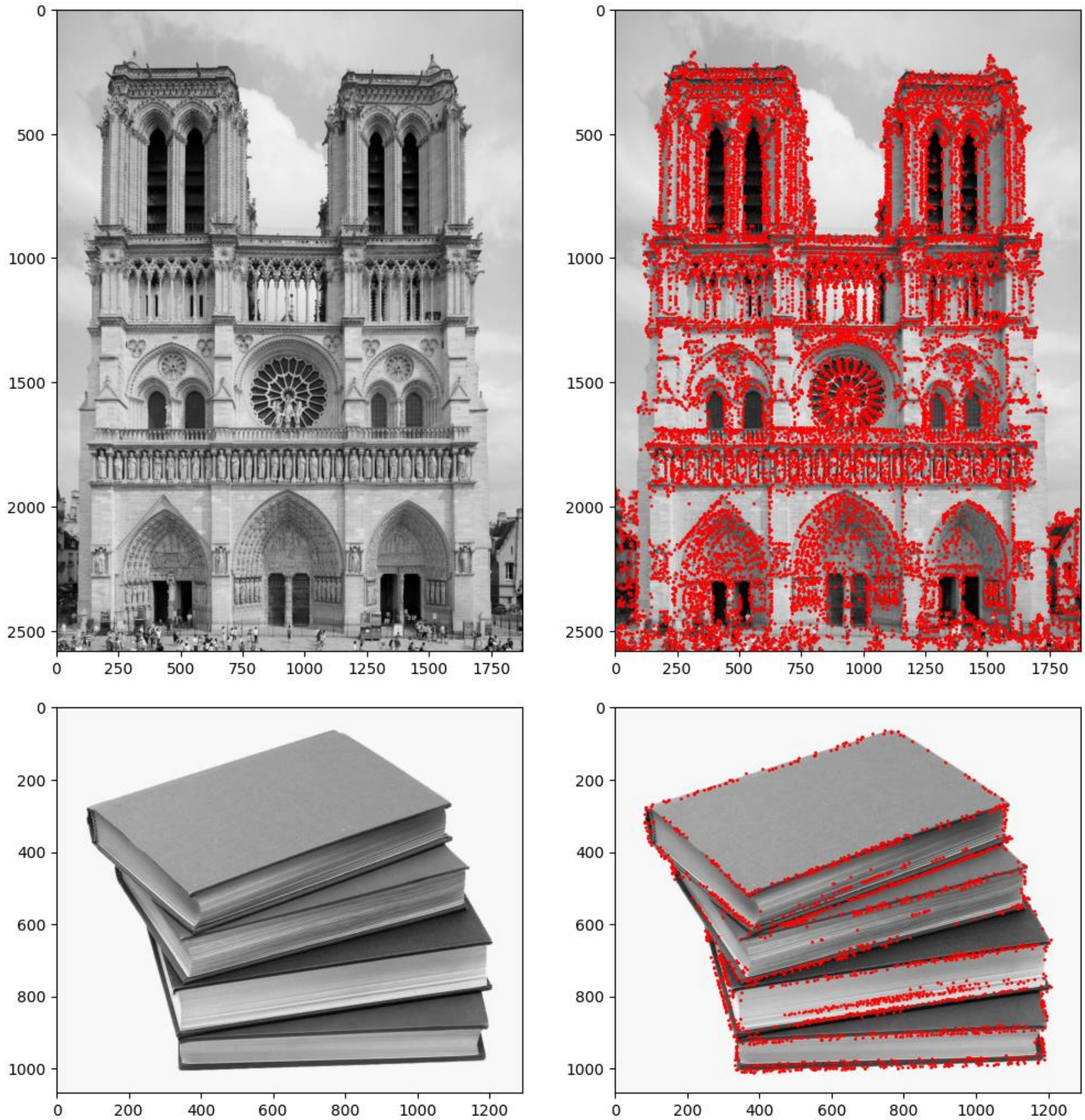




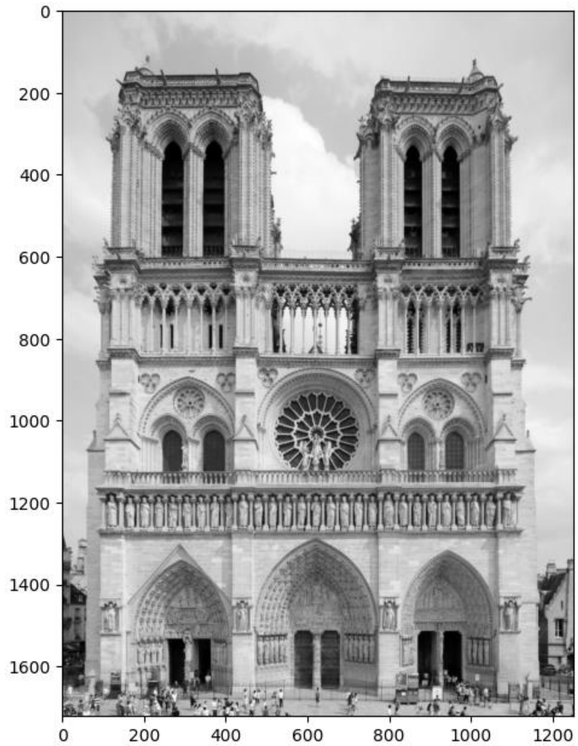
3. Scaled Down

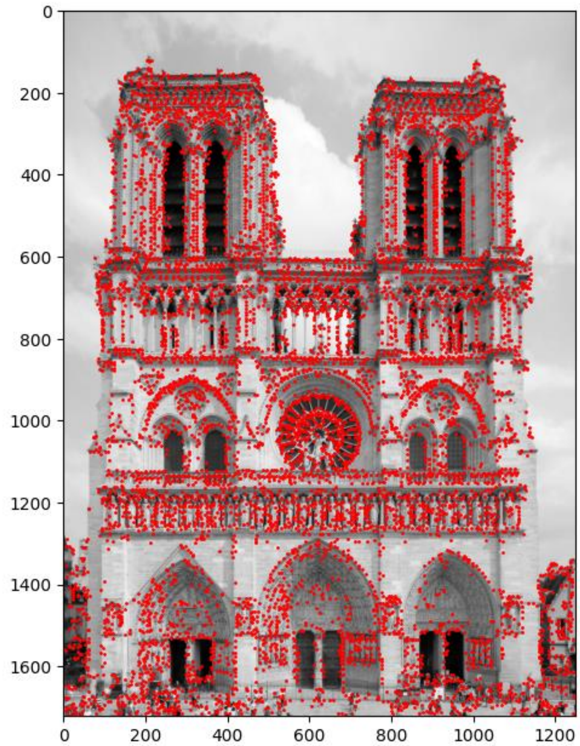


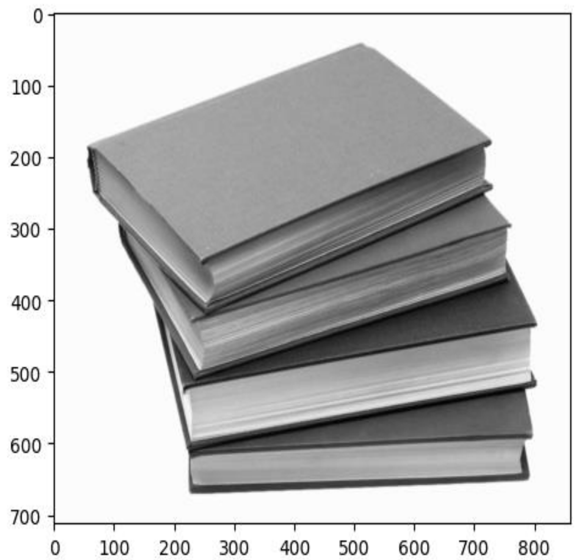
4. Scaled Up

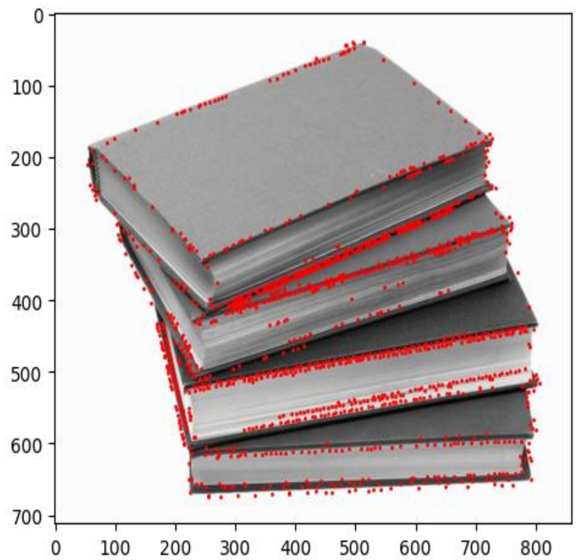

0
500
1000
1500
2000
2500
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
0 200 400 600 800 1000 1200

5. Gaussian Blurring









2. Image Classification (20 Marks)

- Build a custom CNN with conv, sigmoid, pooling and fc layers. Train the network for the CIFAR-10 dataset given [here] using the training data and report the performance on test data. Change the non linearity to ReLU, train and test the model. Compare the results obtained using these two models.
- Evaluate the accuracy on the additional test set given. here.

CODE IMPLEMENTATION:

CNN link

1. Data Loading and Preprocessing which includes reshaping, concatenating of data (training, validation and testing data).
2. Checking number of classes(labels).
3. Creating Sigmoid and ReLu CNN models (Pre-Trained Model in code link)
4. Model Fitting using training data.
5. Predicting on testing data and calculating confusion matrix and classification report.

Classification Matrix On Additional Testing Data:

- **Sigmoid:**

```
Classification Report for model_sigmoid:
              precision    recall  f1-score   support

           0       0.55      0.66      0.60      1000
           1       0.60      0.73      0.66      1000
           2       0.47      0.36      0.41      1000
           3       0.41      0.28      0.33      1000
           4       0.55      0.32      0.40      1000
           5       0.43      0.52      0.47      1000
           6       0.57      0.63      0.60      1000
           7       0.55      0.62      0.58      1000
           8       0.66      0.57      0.62      1000
           9       0.50      0.61      0.55      1000

    accuracy                           0.53     10000
   macro avg       0.53      0.53      0.52     10000
weighted avg       0.53      0.53      0.52     10000
```

- **ReLu:**

```
Classification Report for model_relu:
              precision    recall  f1-score   support

           0       0.70      0.59      0.64      1000
           1       0.69      0.81      0.74      1000
           2       0.59      0.24      0.34      1000
           3       0.57      0.23      0.33      1000
           4       0.51      0.47      0.49      1000
           5       0.55      0.55      0.55      1000
           6       0.40      0.92      0.55      1000
           7       0.70      0.66      0.68      1000
           8       0.77      0.66      0.71      1000
           9       0.67      0.73      0.70      1000

    accuracy                           0.59     10000
   macro avg       0.61      0.59      0.57     10000
weighted avg       0.61      0.59      0.57     10000
```

Comparative Study:

1. Training Performance On Validation Data:

   Sigmoid: Accuracy - 53%

   ReLu: Accuracy – 67%

2. Testing On Additional Test Data:

   Sigmoid: Accuracy - 53%

   ReLu: Accuracy – 59%

3. ReLu outperforms sigmoid in most classes in terms of precision, recall and f1 score.
4. F1 score: comparatively Sigmoid performs well for class 2 and 6 only.
5. Recall : Comparatively Sigmoid performs well in class 0,2 and 3 only.
6. Precision: Comparatively Sigmoid performs well in class 6 only.